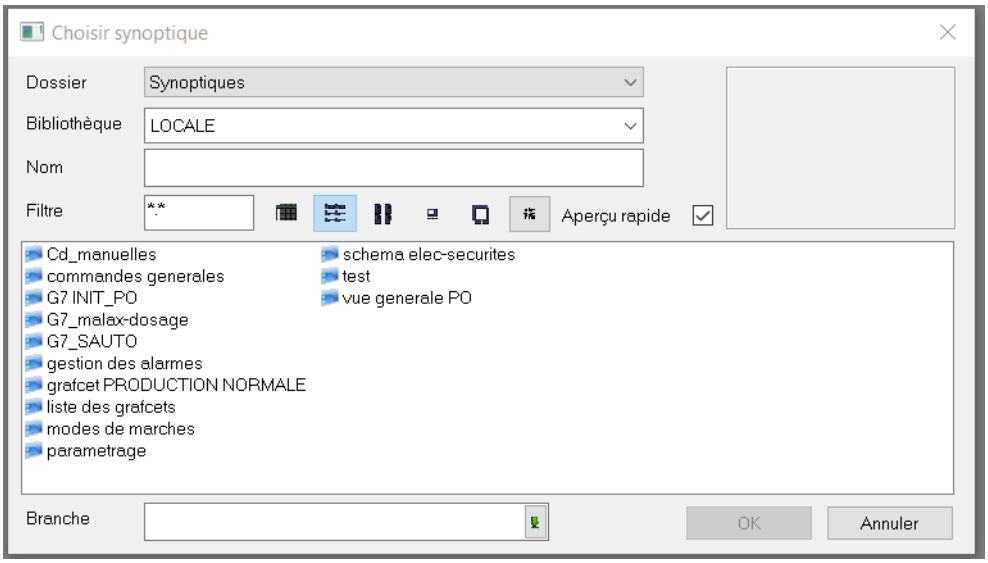
Cahier des charges pour la supervision du malaxeur

Sommaire

1 Démarche

- Sous moodle, télécharger le répertoire `Sup_MALAX_NOM1_NOM2_12_03_2020`
- Le dézipper et enregistrer ce réperoire dans : `C:\ARCInformatique\PcVue12\Usr`
  Les répertoires contenus dans ***Sup_ MALAX_ NOM1_ NOM2_ 12_ 03_ 2020*** *qui nous intéressent sont :*
  - ***B*** *: les images,*
  - ***C*** *: fichier de variables Varexp,*
  - ***W*** *: les synoptiques,*
- Lancer PCvue,
- Dans le menu déroulant *Fichier*, choisir *Ouvrir*,
- Choisir le synoptique *commandes générales*,



- Voici ce synoptique :

Renseigner vos noms à la place de *Binôme : DUPOND et DURAND.*

Changer le nom *COMMANDES MANUELLES.* C'est une erreur ! Noter à la place, par exemple, *LANCEMENT CYCLE AUTO.*

2 Unity Pro

1. Etapes

- Récupérez le programme de malaxeur du semestre dernier (corriger les éventuelles erreurs),
- Sous moodle (lien ci-dessous), vous avez la procédure pour convertir un fichier Unity Pro de l'ancienne version à celle installée en janvier 2020 (Unity Pro XL version 11.0) :
  `https://www.enib.fr/~mecatro/automatismes/S4_supervision/S4_automatismes_aideimport-exportdevariables_07-02-2020.pdf`
- Créer les variables adressées %Mi ou %MWi
- Construire vos équations logiques sous Unity Pro (plus facile que sur PC vue). L'exemple est donné dans ce poly pour **KM11_12**.
- Ajouter une section ladder *Affectation variables*
- Choisir le mode simulation **TCPIP : 127.0.0.1**, générer, transférer et vous mettre en **mode exécution** (RUN).
  `https://moodle.enib.fr/pluginfile.php/53046/mod_label/intro/Mode_simulation.pdf`

2. Ecrans d'exploitation

2.1 Intérêt

C'est beaucoup **plus convivial** et **pratique** d'utiliser des écrans d'exploitation.

2.2 Exemples sous moodle

Vous avez deux exemples sous moodle :

- un répertoire zippé **«tuto_ecran_exploitation.7z»**.
- un deuxième exemple, répertoire zippé **«Exemple2_ecran_exploitation.7z»**,
- un tutoriel relatif au deuxième exemple.

3 Mode démonstration de PCVue

Ce mode de démonstration est limité à **25 variables**.
Quand vous arriverez à la limite de 25 variables, il faut faire une sauvegarde de votre projet.

Vous supprimez ces 25 variables (préalablement soigneusement enregistrées) et vous enregistrez votre projet sous un autre nom avec 25 nouvelles variables et ainsi de suite...Toutes ces variables pourront être ensuite recollées au retour à l'Enib !

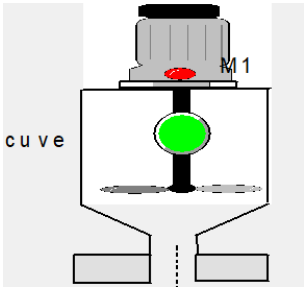
Remarque : La taille de la trame TOR ne bloque pas l'ouverture du projet et ne bloque pas non plus la mise en route de la communication (seul compte le nombre de variables "mappées" sur la trame).

4 Synoptiques

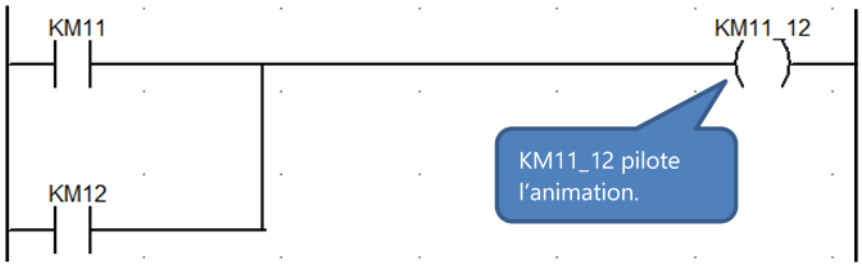
1. Partie opérative

- **Pour les fins de course du vérin (_1s1 et_1s2), et le capteur de présence du bac (pb) :**
  - voyant vert quand le capteur est activé,
  - voyant gris quand le capteur est désactivé,
- **Pour la rotation du moteur :**
  - Allumer le voyant correspondant au sens de rotation commandé.

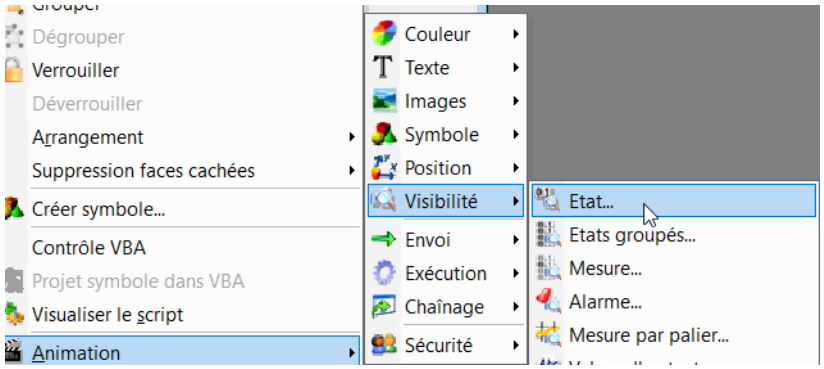
  - Allumer le voyant lorsque le moteur tourne peu importe le sens.



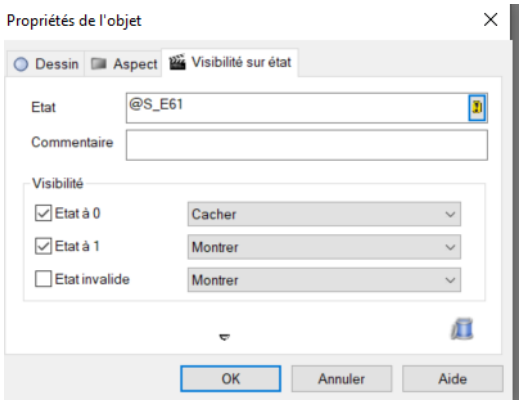
Il faudra créer un bit **KM11_12** sous Unity Pro comme proposé ci-dessous :



- **Pour les mouvements du vérin :**
  - Cette flèche  est visible lors de la sortie du vérin et disparaît quand le vérin est arrivé en fin de course.
  - Même principe pour cette flèche  .
  - ClicD sur chaque flèche, et choisir une *Animation/Visibilité/Etat.*

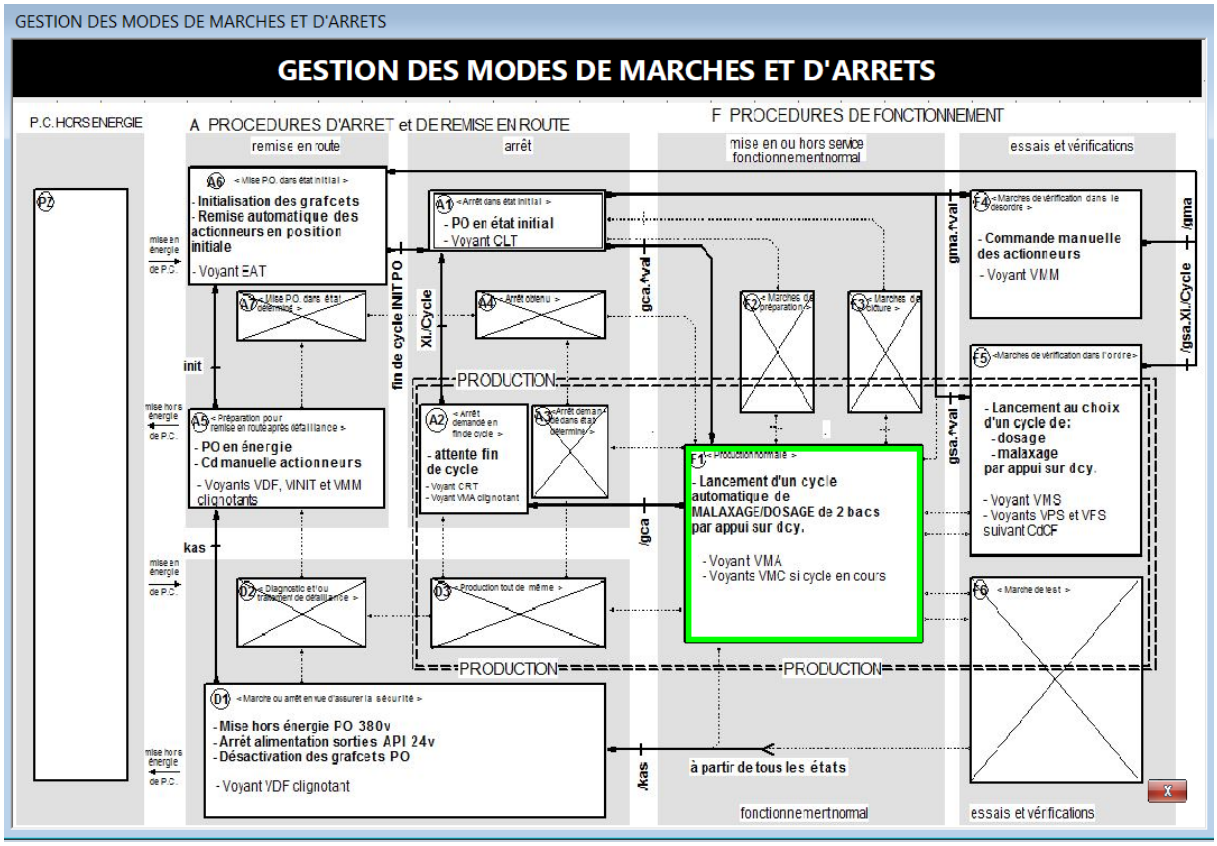


- Faire en sorte que la flèche s'affiche lors du déplacement du vérin.



2. Gemma

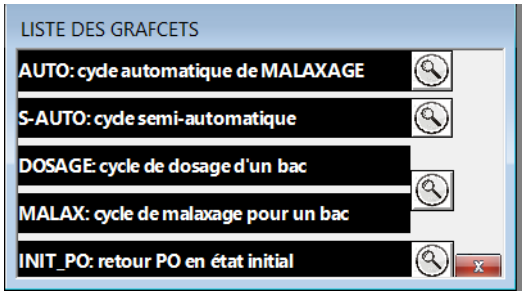
- Affichage du mode de marche **actif** par un **cadre vert** *(exemple donné pour le mode production normale)*,
- Affichage par un voyant vert de chacune des conditions d'évolution (facultatif)



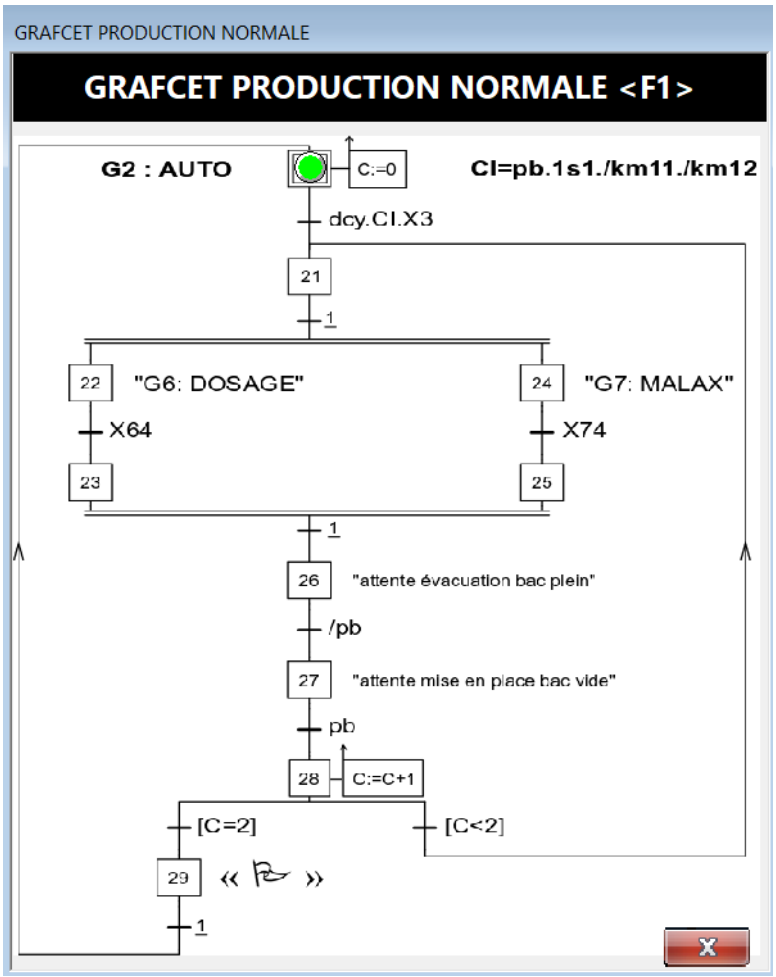
3. Grafcets

Affichage de la situation (étapes actives) des grafcets suivants :

- grafcet automatique,
- grafcet semi-automatique,
- grafcets élémentaires de dosage et de malaxage,
- grafcet d'initialisation



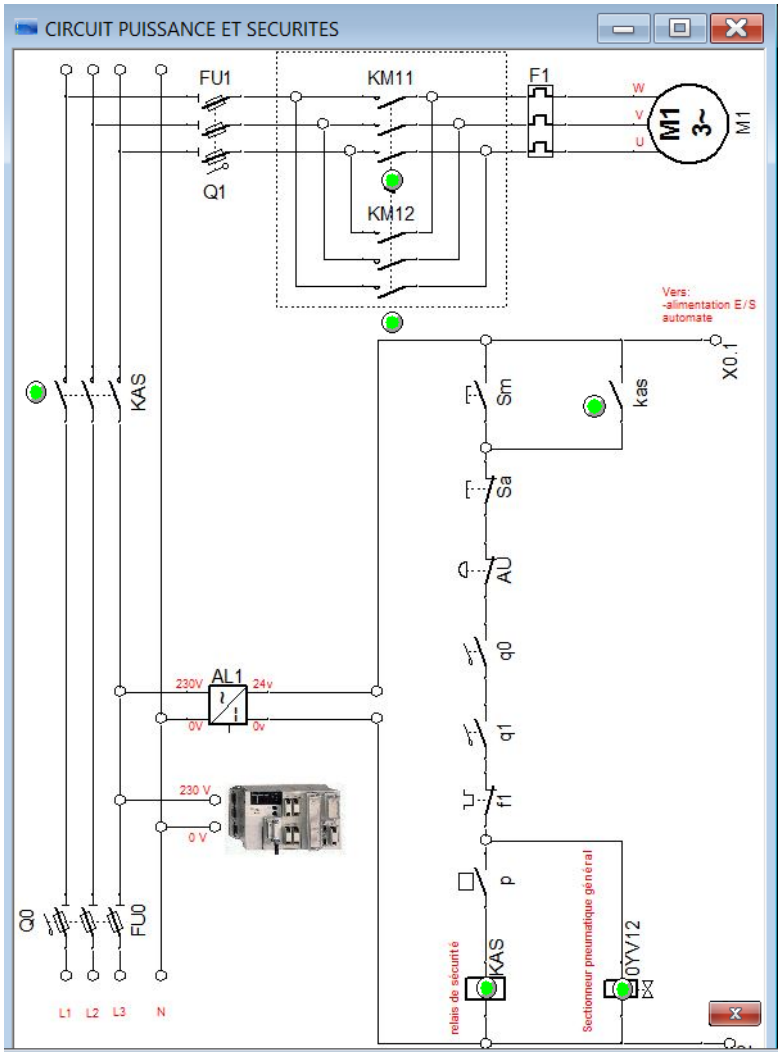
Exemple : *On est à l'étape initiale du grafcet automatique :*



4. Maintenance

Réaliser les animations des divers éléments présents su ce circuit :

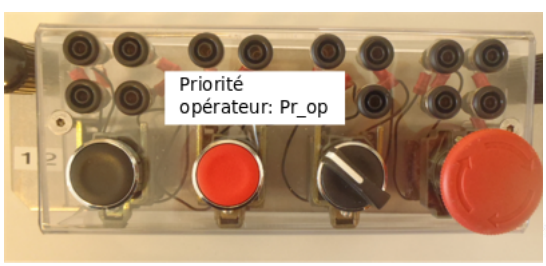
- voyant vert quand activation d'une bobine (ou électrovanne) ou état **travail** d'un contact,
- voyant gris quand non-activation d'une bobine (ou électrovanne) ou état **repos** d'un contact,

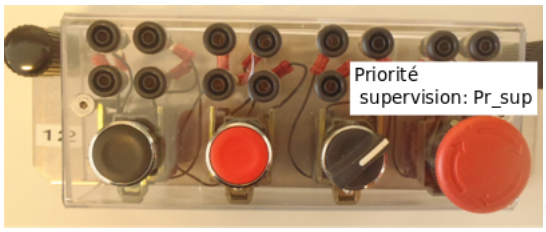


5. Lancement du cycle auto

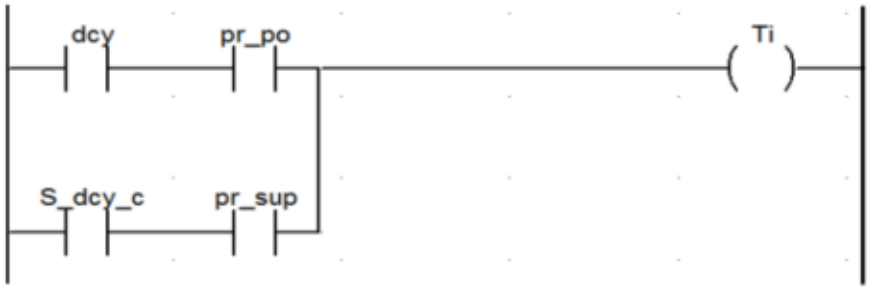
5.1 Priorité

- Le lancement d'un cycle à partir de la supervision pose un problème de sécurité vis à vis l'opérateur PO. Pour **sa sécurité**, seul l'opérateur **PO** (au poste de la machine réelle) peut modifier la sélection de priorité.
- Le **synoptique de supervision** informera quant à lui, du choix ayant été fait par l'opérateur **PO**.
- L'opérateur a besoin d'un sélecteur de priorité **à deux états** : «Pr_op» *priorité donnée à l'opérateur* ou «Pr_sup» *priorité donnée à la supervision*. Quand on place le sélecteur sur le choix «Pr_sup», le sélecteur doit se tourner vers votre PC.

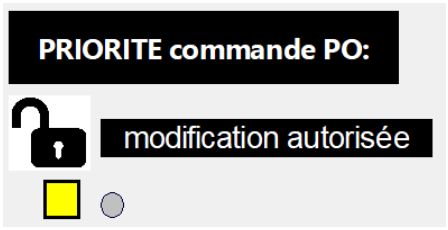




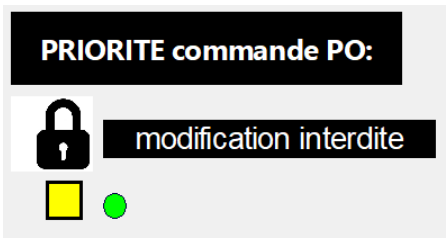
- Les variables « Pr_sup » et « Pr_po » sont à utiliser pour sécuriser le lancement des cycles. On ne doit pouvoir que lancer un cycle depuis la supervision si la priorité est donnée à la supervision et vice-versa.



- Cette configuration *cadenas ouvert* signifie que l'opérateur pourra lancer le cycle depuis la supervision.



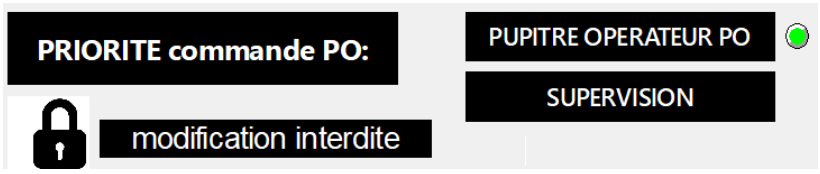
- Cette configuration *cadenas fermée* signifie que l'opérateur **ne pourra pas** lancer le cycle depuis la supervision.



Remarque : Les animations de cadenas et de changement de texte (modification autorisée à modification interdite) sont déjà faites. Appuyez sur le carré jaune pour tester.

- Les animations **doivent être modifiées** pour que le cadenas s'ouvre ou se ferme suivant la position de l'interrupteur rotatif du pupitre du banc. Le carré jaune et le voyant vert sont à enlever par la suite.
- La commande sur **dcy** depuis la supervision **ne doit pas être ambigüe**. C'est pour cela qu'il faudra une visualisation comme ci-dessous de :
  - **la priorité supervision** :

  - **la priorité PO** :



5.2 Lancement cycle

Une bibliothèque est disponible sous PC vue à l'endroit suivant :

`C:\ARCInformatique\PcVue12\Lib\SH_COMMANDS\B`

Voici un extrait des boutons que vous pouvez éventuellement choisir :


COMMANDS_PUSH_BUTTON_ROUND_GLOW_CURVED_POWER_ON_GREEN.png


COMMANDS_PUSH_BUTTON_ROUND_GLOW_POWER_ON_GREEN.png


COMMANDS_SWITCH_TOGGLE_TYPE1_LED_OFF.png


COMMANDS_SWITCH_TOGGLE_TYPE1_LED_ON.png


COMMANDS_SWITCH_TOGGLE_COLORED_GREEN.png

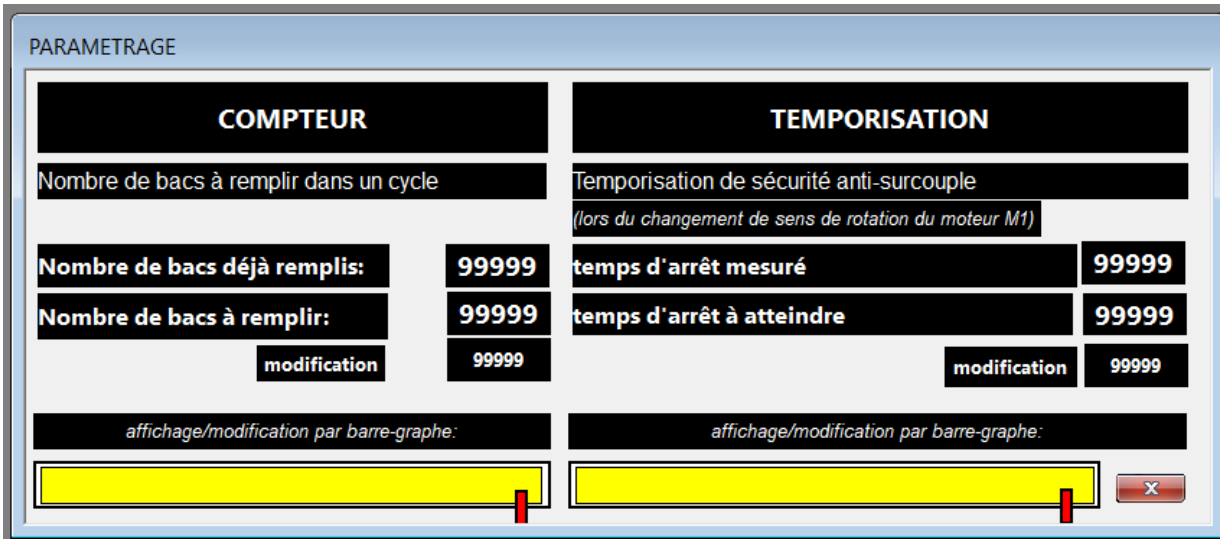

COMMANDS_SWITCH_TOGGLE_COLORED_RED.png

Il suffit de les superposer les boutons on et off et de paramétrer les animations en conséquence. Il est possible que sur votre synoptique, l'image ne soit pas présente. C'est pour cela que le mieux est d'aller chercher dans cette bibliothèque, l'image qui vous convient.

6. Paramètres

Deux paramètres seront modifiables, comme le montre l'image ci-dessous :

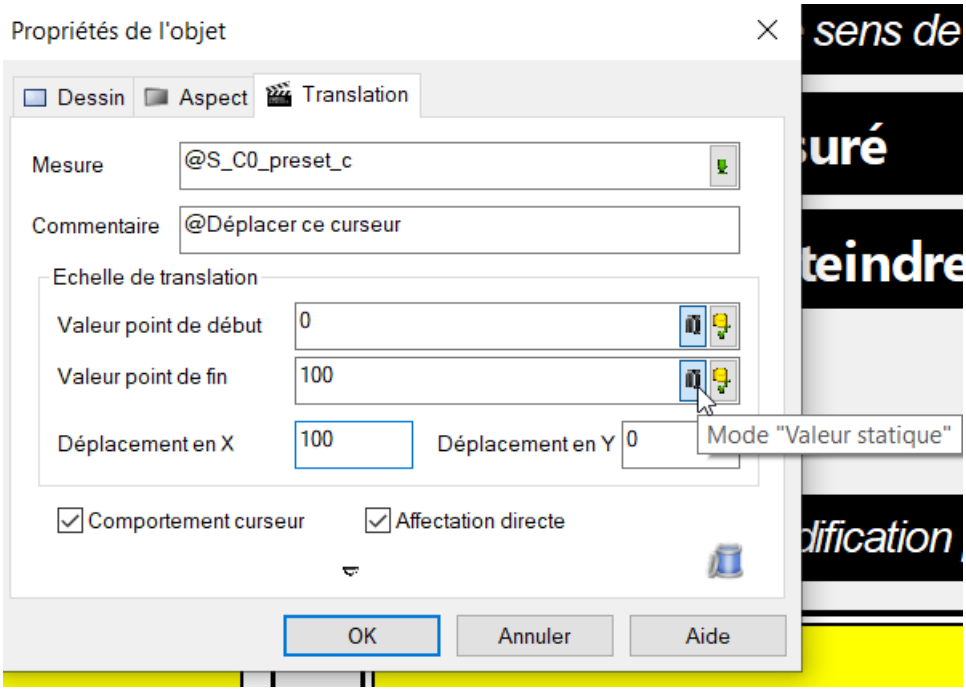
- **le compteur du nombre de bacs et la temporisation entre le changement de sens de rotation du moteur :**



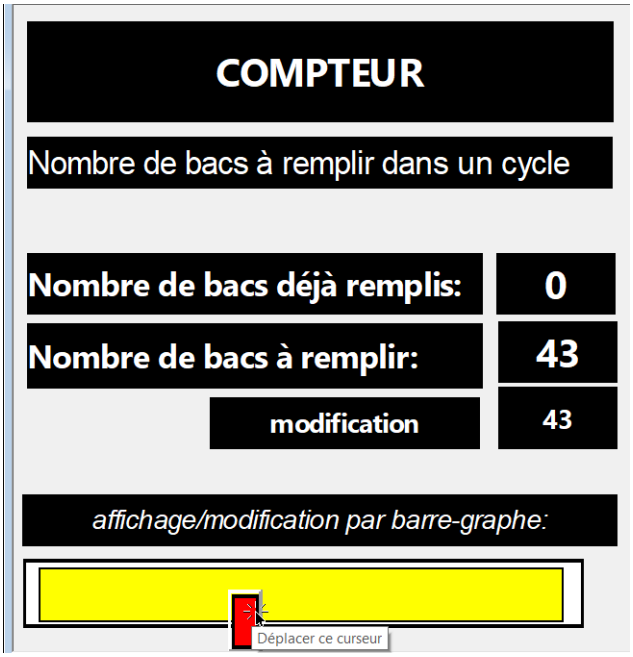
Pour réaliser le déplacement du curseur,

- **clicD** sur le rectangle rouge et choisir *Animation/Position/Translation.*

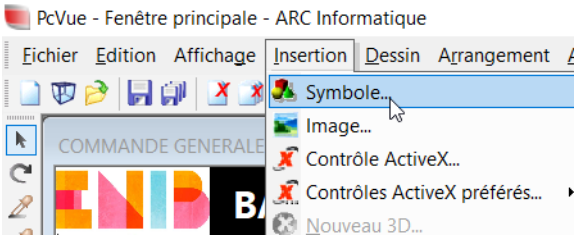
- Renseigner les champs ci-dessous.
  - *Commentaire :* :permet l'affichage dune bulle d'aide lorsque la souris passe au-dessus de la zone de contrôle.
  - *Echelle de translation* indique les valeurs des points de début et de fin (spécifie la plage à prendre en compte pour positionner l'objet graphique). Le déplacement en X et en Y représente le nombre de pixels à déplacer. Normalement, vous sélectionnez X ou Y parce que c'est une animation de translation sur un seul axe.
  - *Comportement du curseur et affectation directe.* Ces cases sont à cocher pour utiliser l'objet graphique comme un curseur.

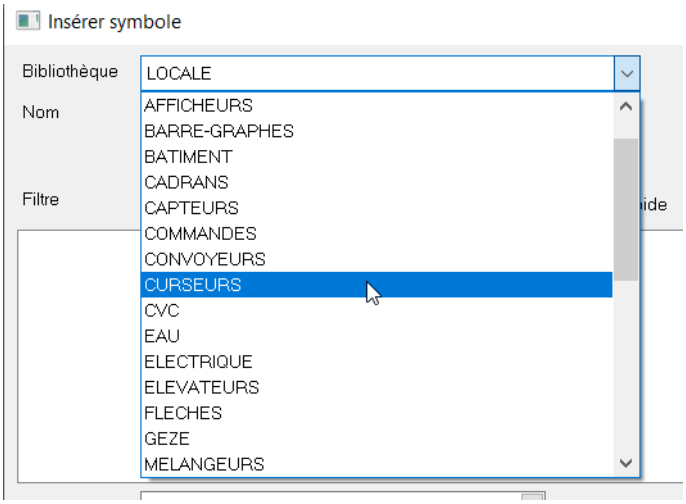


- Quand on déplace le curseur, la valeur modifiée apparaît directement dans le champ *modification* et *nombre de bacs à remplir*.

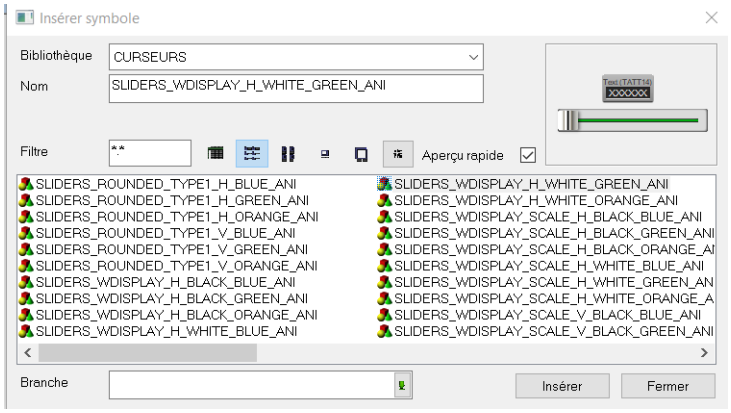


Vous pouvez choisir d'autres curseurs (ou sliders) dans **la bibliothèque de PCvue**.

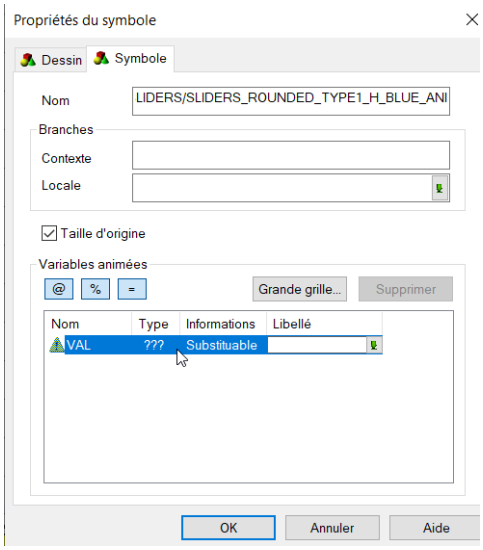


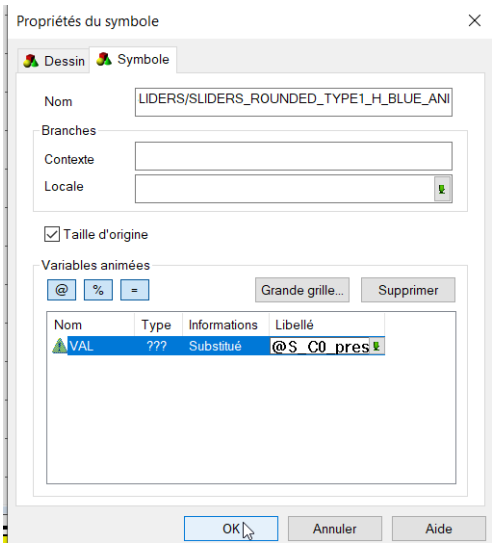


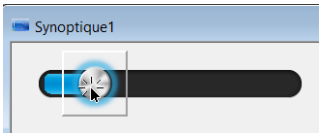
Vous avez l'embarras du choix :



C'est très rapide à configurer. Choisissons au hasard le premier,. Il n'y a que la variable à renseigner.







Il s'agit pour **ces deux paramètres** comme expliqué dans le poly (de prise en main de PCvue) :

- de faire afficher **la valeur courante**,
- de pouvoir **modifier** la valeur de présélection depuis la supervision soit :
  - *en tapant la nouvelle valeur dans le champ modification,*
  - *en bougeant le curseur rouge*
- de faire afficher **la valeur de présélection**,

7. Gestion des alarmes

7.1 A lire avant

https://moodle.enib.fr/pluginfile.php/51110/mod_label/intro/manuel_Pcvue_alarmes.pdf?time=1585570171675

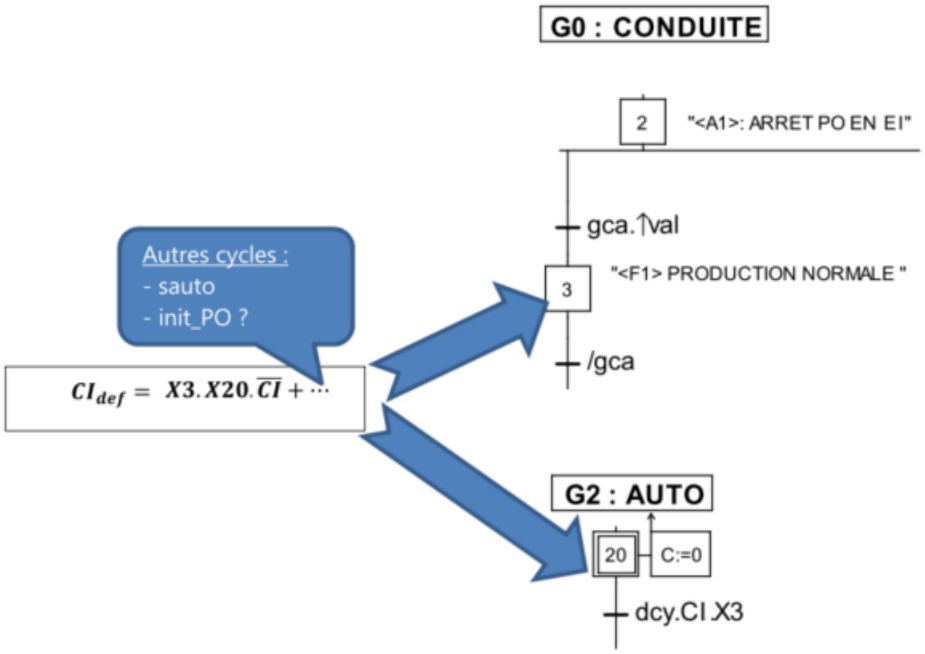
7.2 Synoptique PCvue

7.3 ALARME 1 : Traitement défaut en cours : Trt_df

- Il faut prévoir une alarme traitement défaut, **Trt_df**. Cette variable **sera à 1** dans :
  - le mode <D1> : «Marche ou arrêt en vue d'assurer la sécurité»
  - le mode <A5> : «Préparation pour remise en route après défaillance».
- Quand cette variable sera à **1**, cela voudra dire qu'il faut **traiter le défaut**.

7.4 ALARME 2 : Défaut Conditions Initiales : CI_df

- On veut par exemple, lancer le cycle auto et les conditions initiales ne sont pas présentes. C'est intéressant de savoir qu'il y a un problème de lancement de cycle à cause des conditions initiales qui ne sont pas respectées.
  On prévoira une alarme et on la nommera **CI_df**, défaut conditions initiales.
  L'exemple donné *ci-dessous* est pour le lancement du cycle automatique mode <F1> «Production Normale».

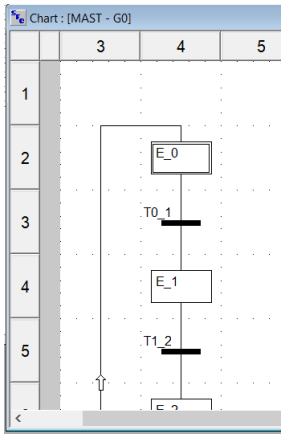


Le défaut CI représente le problème rencontré par l'opérateur lorsqu'il demande le lancement d'un cycle, et que toutes les conditions nécessaires sont validées sauf le respect des conditions initiales.

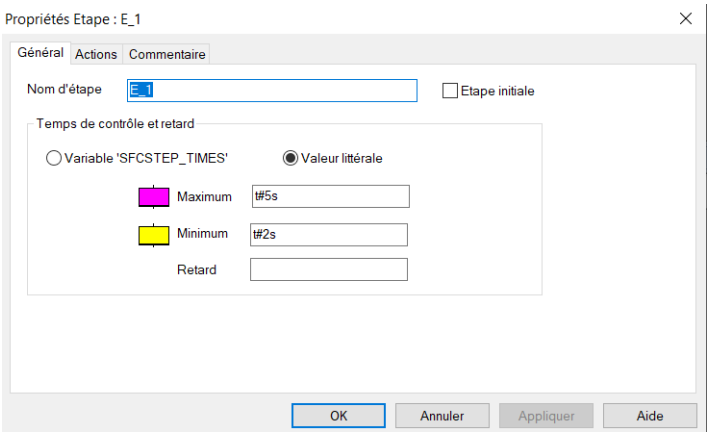
Remarques : Les conditions initiales peuvent être instables en raison de défauts capteurs ou de blocage de course sur l'actionneur, ou de fuites internes aux électrovannes...

7.5 ALARMES 3 et 4 : Temps enveloppes

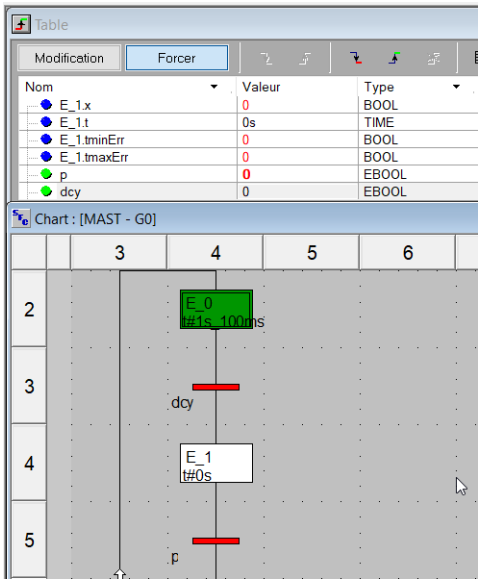
- **Explication sur les temps enveloppes sur un exemple quelconque**
  - Sous Unity Pro, clic2 sur l'étape 1.



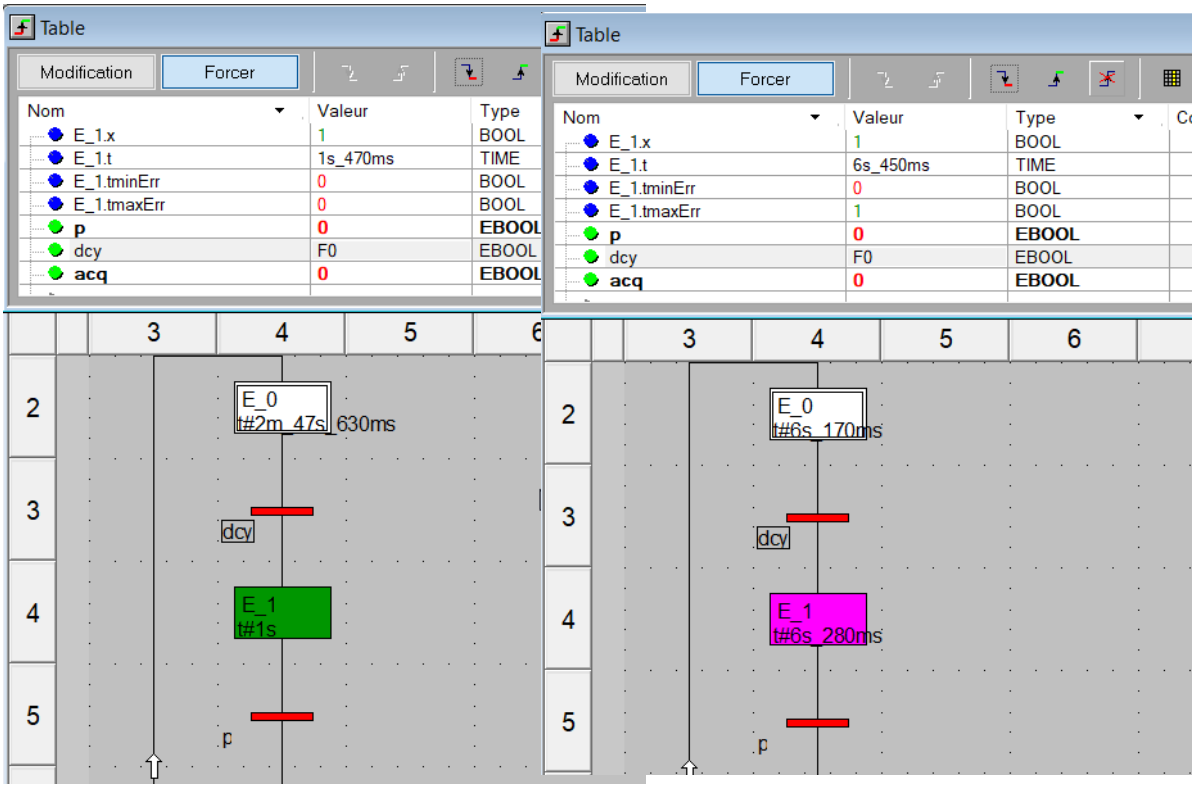
  - Vous réglez *le temps de contrôle maximum* à 5s en tapant dans le champ ci-dessous **t#5s** et *le temps de contrôle minimum* à 2s en tapant dans le champ ci-dessous **t#2s**. Ce sont des valeurs prises au hasard pour l'exemple.



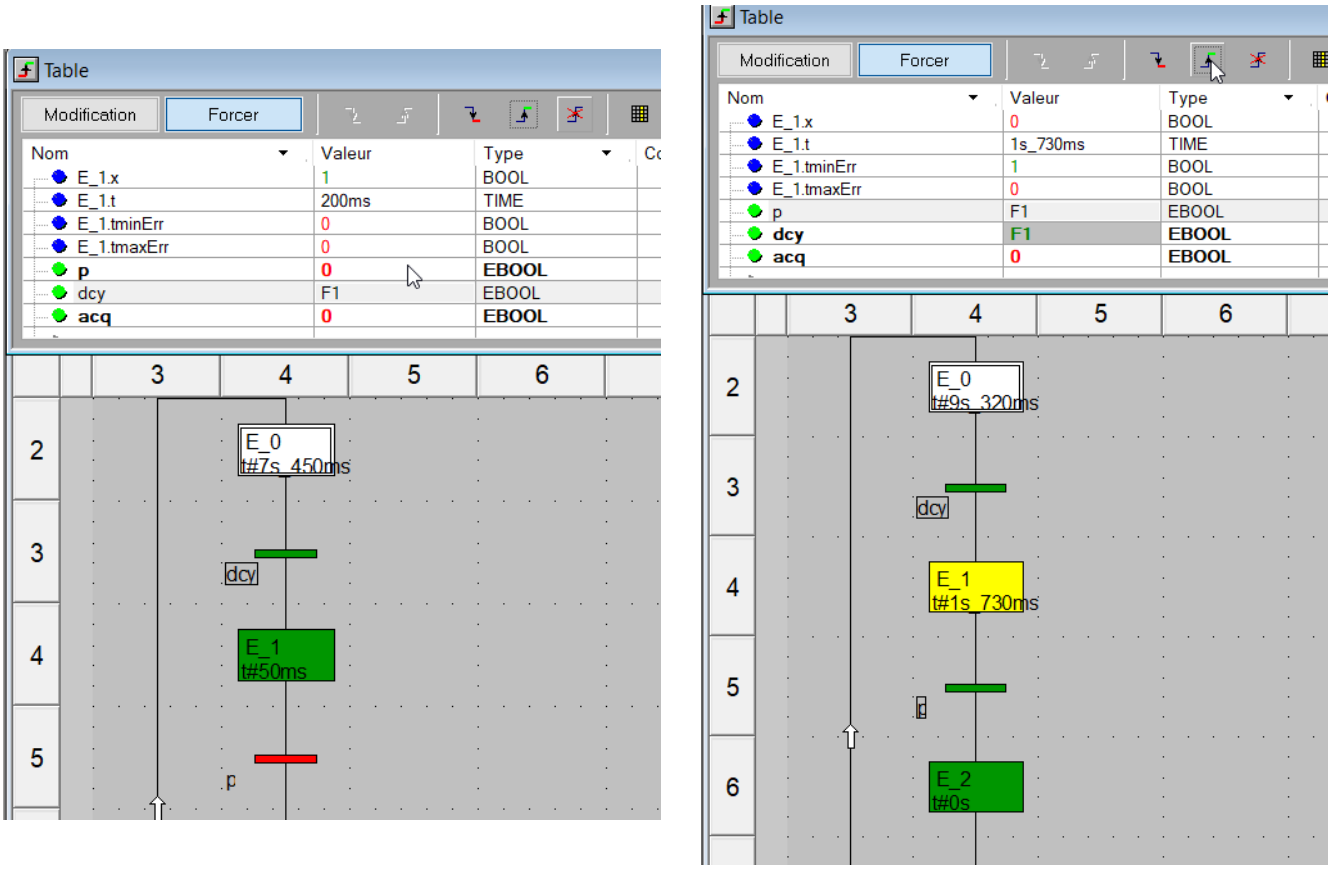
  - Dans la table, on suit en particulier, les variables **E_1.tminErr** et **E_1.tmaxErr**.



- Dans la table, on suit en particulier, les variables **E_1.tminErr** et **E_1.tmaxErr**. On remarque que si le temps d'activité de l'étape dépasse *le temps de contrôle maximum* défini à **5s**, elle passe de la couleur verte à la couleur violette et la variable booléenne **E_1.tmaxErr** passe à 1.



- Maintenant, si le temps d'activité de l'étape est inférieur au *temps de contrôle minimum* défini à **2s**, elle passe de la couleur verte à la couleur jaune et la variable booléenne **E_1.tminErr** passe à 1.



- La variable **E_1.tminErr** reste à **1** tant que *l'étape 1* n'a pas été réactivée. Lors de la réactivation de *l'étape 1*, elle passe à **0** et si le temps est maintenant supérieur à **2s**, elle reste à **0** (comportement similaire avec **E_1.tmaxErr**).

- **Cas du malaxeur**

  Dans le cas du malaxeur, étant donné qu'il y a un compteur, deux remplissages de bacs ont lieu. Il se peut qu'il y ait un dépassement du temps enveloppe pour le premier bac et pas pour le deuxième bac. Dans ce cas là, la variable **E_1.tmaxErr** sera à **0 alors qu'il y a eu un problème**.
  Cela traduit un fonctionnement instable qui mérite d'être corrigé.

  - Admettons que le vérin **met un temps trop important** pour atteindre sa fin de course. On utilisera une alarme que l'on nommera **1A2max_df**.

    Cette variable sera à élaborer à partir de la variable **E_i.tmaxErr**, *i étant le numéro de l'étape concernée*, mais attention, il ne faudra pas la prendre strictement identique.

    On établira un temps de référence (ou temps de contrôle supérieur, pour reprendre le terme d'Unity Pro) et on comparera le temps mis par le vérin pour aller jusqu'en fin de course à **ce temps de contrôle supérieur**.

  - Admettons maintenant que le vérin **met un temps trop faible** pour atteindre sa fin de course. On utilisera une alarme que l'on nommera **1A2min_df**.

    De même, si le temps mis par le vérin pour aller jusqu'en fin de course est inférieur à **ce temps de contrôle inférieur**, une alarme **1A2min_df** sera activée.