



Simulateur pour WinRelais

1.1 04/04/23



INGE=REA

Révision 1

Caractéristiques de ce document

Logiciel	WRsimulateur
Version	1.1 portée par WinRelais EDU 2.4.6 (2.4f) STUDIO
Date	4 avril 2023
Auteur	POPEK Frédéric / Auteur de WRsimulateur
Editeur	INGE=REA
Licence	Libre de droits

SOMMAIRE

Objectifs du simulateur	3
➤ Limitations de la version 1.1	3
Bibliothèque simulable	5
➤ Accessibilité	5
➤ Créer un symbole (fonction avancée)	6
➤ Modes de dessin par défaut	10
Composants simulables	11
➤ Paramétrage	11
➤ Signalétique	12
➤ Blocs de configuration généraux	13
Synoptiques et animation	14
➤ Préambule	14
➤ Comment décrire un objet physique ou cinématique	14
➤ Comment décrire une liaison glissière	17
➤ Un cas particulier de glissière : le vérin	19
➤ Liaison glissière et parenté	19
➤ Comment décrire un convoyeur	20
➤ Sous-ensembles synoptiques	21
Intégrer un variateur de vitesse	22
Utilisation du grapheur	24
Intégrer un automate programmable	26
➤ Préambule	26
➤ Exemple ascenseur	26
➤ Principe de construction d'un objet API	31

Objectifs du simulateur

L'objectif du simulateur est de permettre aux utilisateurs de simuler des circuits électriques, électrotechniques et pneumatiques pour un usage pédagogique et de présentation en avant-projet.

Le simulateur permet notamment la simulation d'installations de type SAP¹ en utilisant un automate programmable virtuel connecté en MODBUS/TCP-IP, couplé à une partie opérative animée.

Le simulateur intègre un modèle de simulation mixte analogique/numérique qui permet la simulation simultanée

- de circuits électriques/électrotechniques standards,
- de circuits pneumatiques,
- d'objets complexes (variateurs de vitesse, relais de sécurité, etc.)
- d'automates programmables,
- de parties opératives simples

Limitations de la version 1.1 du simulateur

Modèle numérique

- Les charges mécaniques appliquées aux actionneurs (vérins, moteurs) sont fixes. Elles correspondent au point de fonctionnement nominal des actionneurs.
- Les transitoires des courants et des tensions ne sont pas simulées.
- La fréquence d'acquisition des appareils de mesure est limitée à 50 Hz (1 mesure toutes les 20 ms).
- Les mesures sont données en valeur efficace ou moyenne uniquement.

Edition des schémas

- Il n'existe pas pour l'instant de 'moulinette' pour convertir des schémas antérieurs en schémas simulables.
- Il n'est pas possible de mixer des symboles simulables et non simulables dans un folio.
- L'épaisseur des traits pour les conducteurs est fixe.
- Seule la police de caractères 'Arial' est supportée.
- L'alignement de texte supporté est 'gauche' uniquement.
- Les caractères gras et italiques sont transformés en caractères normaux.
- Les cadres et les cartouches de folio ne sont pas dessinés dans le simulateur.
- Une seule taille de symbole est utilisable : la taille 'normale'.

¹ SAP : Système Automatisé de Production

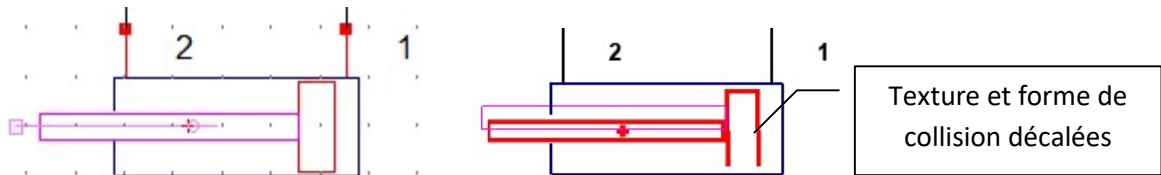
Edition des symboles dans WinSymbole

Certaines fonctionnalités graphiques proposées par WinSymbole ne sont pas prises en compte par le simulateur :

- Pas de courbe de Bézier, ni de texte, cercle non rempli uniquement,
- épaisseur des contours fixe,
- Pas de graphique SVG,
- Pas d'image d'arrière plan
- Pas de mini-dessin.

Anomalie d'édition d'un symbole

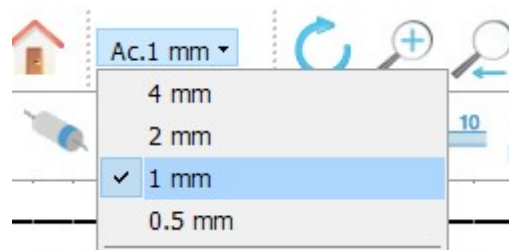
- Il peut arriver qu'un décalage de la texture et de la forme de collision survienne suite à la rotation d'un symbole dans WinRelais. Il est alors nécessaire de recentrer l'origine du symbole dans WinSymbole. (Voir [page 17](#) sur la question du barycentre)



- Il est préférable de mener la rotation plutôt dans WinSymbole, d'ajuster avec soin le barycentre puis de sauver le symbole pour réutilisation.

IMPORTANT

Pour permettre la pose des composants simulables sur les folios en toute sérénité, il est fortement recommandé de définir le pas d'accrochage de la grille à 1 mm dans WinRelais et WinSymbole

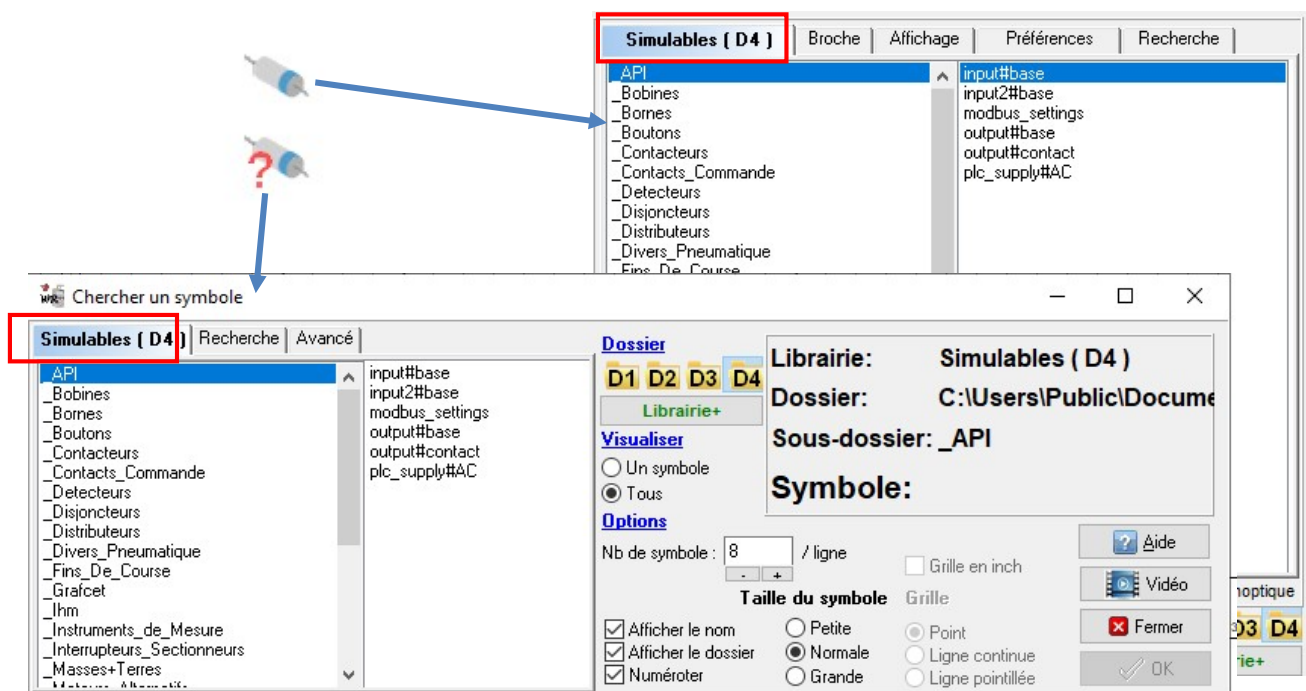
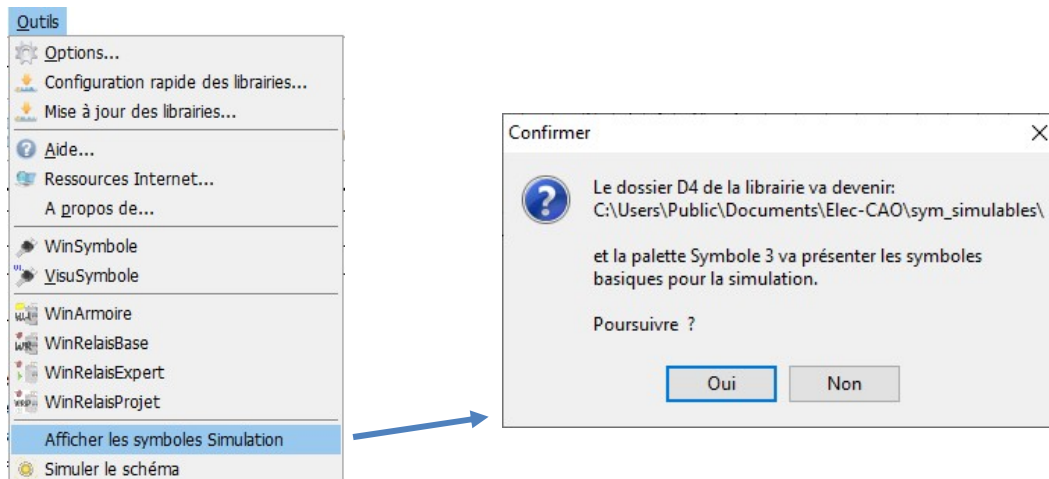


Bibliothèque simulable

Accessibilité

La bibliothèque des composants simulables est disponible dans le dossier D4 préparé par WinRelais.

Pour faciliter l'utilisation de cette bibliothèque, WinRelais propose la commande suivante :



Deux façons d'accéder aux symboles simulables

Créer un symbole à partir d'un symbole existant (fonction avancée)

Il est possible de décliner un nouveau symbole à partir d'un symbole existant pour adapter son dessin et la disposition des bornes selon les besoins. Le nouveau symbole héritera de toute la logique de simulation du symbole père.

Le visuel du symbole peut-être défini avec WinSymbole ou avec une image (texture) au format png.

Contraintes :

- Le nom des broches ne doit pas être modifié.
- Le nombre de broches ne doit pas changer.

Exemple 1 : On souhaite créer un symbole d'ampèremètre de forme carrée.

1. Placer le symbole 'ampere metre' sur le folio.
2. Aller dans [Modifier un symbole\Avancé\Exécuter WinSymbole](#).
3. Modifier le symbole à sa convenance dans WinSymbole. Enregistrer la modification (Ctrl-S).
4. [Accepter les modifications](#).
5. Sauver le symbole dans la bibliothèque '_InstrumentsDeMesure' en le nommant 'ampere metre#carre'.

Attention : ne pas utiliser de caractères accentués pour nommer les symboles

Il est très important de garder le même nom de départ, 'ampere metre' dans l'exemple, puis d'ajouter le caractère '#' suivi du nom de son choix.

Le symbole '#' indique au simulateur que le nouveau symbole 'carre' est une déclinaison du symbole 'ampere metre'.

Placer un symbole: A2, mesure= efficace

1

2

3

4

5

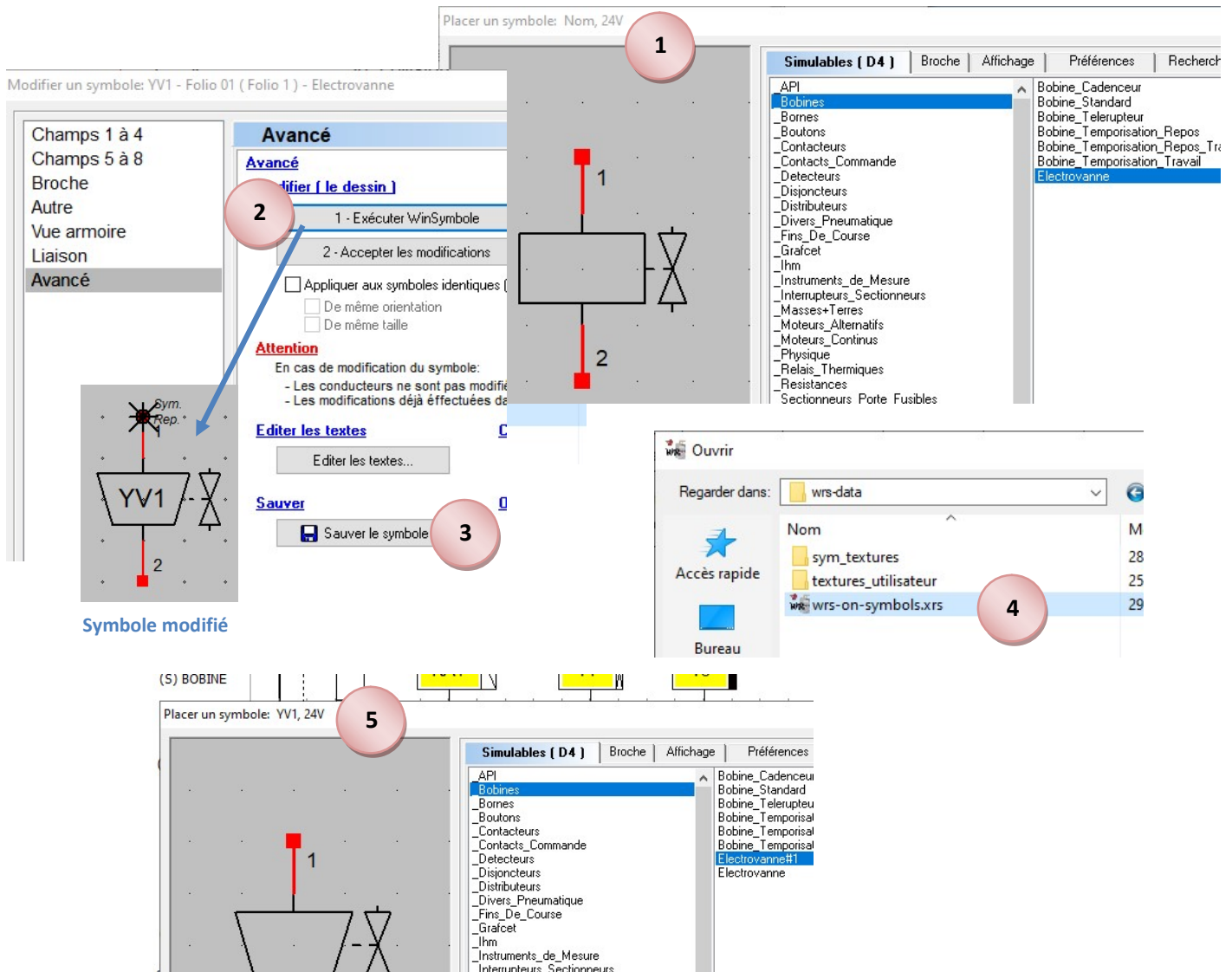
Symbole modifié dans WinSymbole

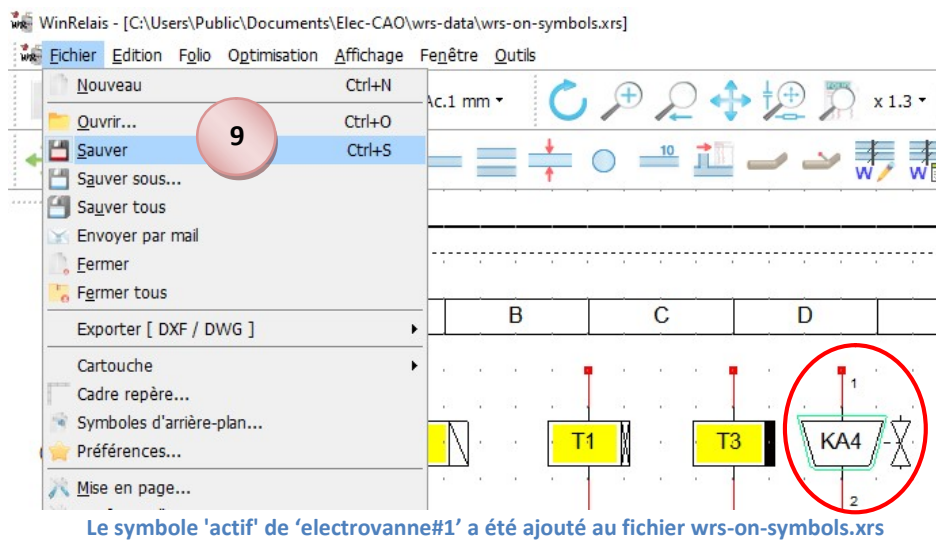
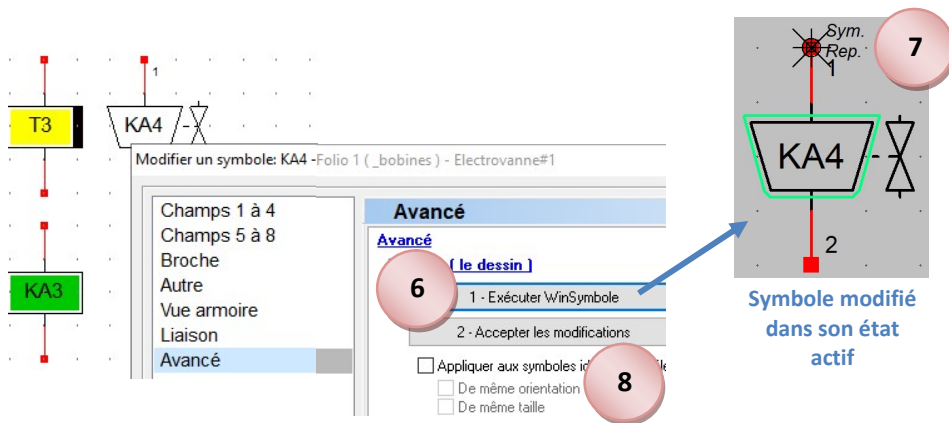
Exemple 2 : On souhaite ajouter un symbole d'électrovanne à la famille '**_Bobines**' qu'on appellera '**electrovanne #1**'.

L'opération est plus longue que dans l'exemple précédent car il faut créer le dessin du symbole quand il est actif. Le dessin des symboles actifs sont rangés dans le fichier :

C:\Users\Public\Documents\Elec-CAO\wrs-data\wrs-on-symbols.xrs

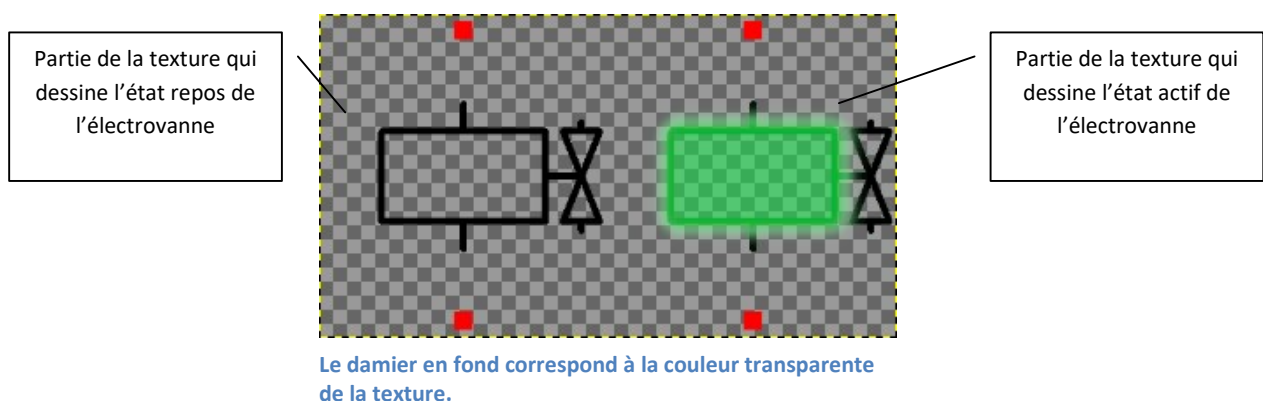
1. Placer le symbole '**electrovanne**' de la famille '**_Bobines**' sur le folio.
2. Modifier le symbole comme dit aux points 2.3 et 4 de l'exemple 1.
3. Sauver le symbole dans la bibliothèque '**_Bobines**' en le nommant '**electrovanne #1**'.
4. Ouvrir le schéma '**wrs-on-symbols.xrs**' dans WinRelais. Sélectionner de préférence le folio '**_bobines**'.
5. Placer le symbole fraîchement enregistré '**electrovanne #1**' sur le folio.
6. Aller dans **Modifier un symbole\Avancé\Exécuter WinSymbole**.
7. Modifier le symbole à sa convenance dans WinSymbole. Enregistrer la modification (Ctrl-S).
8. **Accepter les modifications**.
9. Enregistrer le schéma '**wrs-on-symbols.xrs**'.



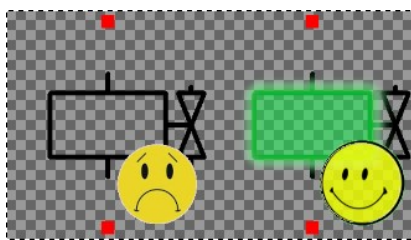
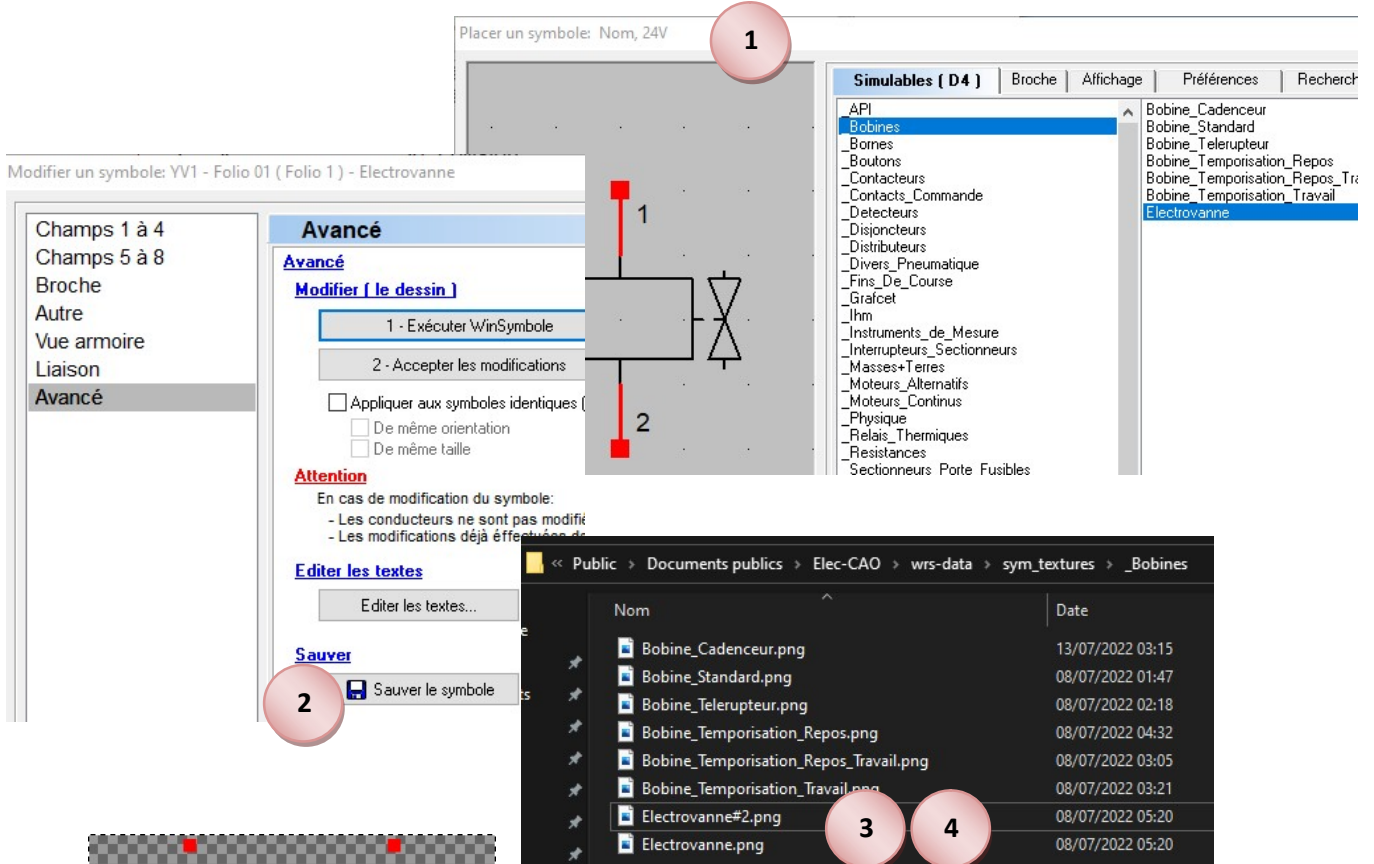


Exemple 3 : Utilisation d'une image au format png pour créer le visuel d'une electrovanne qui sera nommée '**electrovanne #2**'

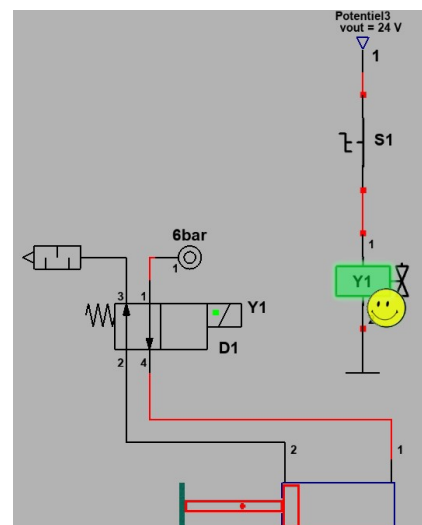
On peut trouver que les possibilités de dessin de WinSymbole ne sont pas suffisantes pour créer le visuel des symboles. Il est alors possible d'utiliser une texture qui sera éditée dans un logiciel à part. Le visuel de l'objet 'electrovanne' initial manipulé précédemment est construit avec cette texture :



1. Placer le symbole 'electrovanne' de la famille '_Bobines' sur le folio.
2. Sauver le symbole dans la bibliothèque '_Bobines' en le nommant 'electrovanne #2'.
3. Accéder à la texture 'electrovanne.png' présente dans le dossier :
C:\Users\Public\Documents\Elec-CAO\wrs-data\sym_textures_Bobines\Electrovanne.png
4. Copier dans le même dossier la texture renommée en 'electrovanne #2.png'
5. Editer la texture 'electrovanne #2.png' dans un logiciel adéquat, (GIMP par exemple) selon les besoins, puis enregistrer. (commande 'Ecraser electrovanne#2.png' avec GIMP)



Modification de la texture avec le logiciel GIMP



Voilà à quoi ressemble la nouvelle électrovanne dans la simulation

Mode de dessin par défaut des familles de symboles

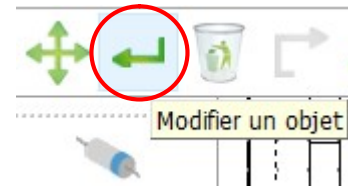
Famille	Mode de dessin par défaut
_API	WinSymbole
_Bobines	Texture
_Bornes	Texture
_Boutons	Texture
_Contacteurs	WinSymbole
_Contacts_Commande	Texture
_Detecteurs	WinSymbole/Texture*
_Disjoncteurs	WinSymbole
_Divers_Pneumatiques	WinSymbole
_Fins_De_Course	Texture
_Grafcet	WinSymbole
_Intruments_de_Mesure	WinSymbole
_Interrupteurs_Sectionneurs	WinSymbole
_Masses+Terres	WinSymbole
_Moteurs_Alternatifs	WinSymbole
_Moteurs_Continus	WinSymbole
_Relais_Thermiques	WinSymbole
_Resistances	WinSymbole
_Sectionneurs_Porte_Fusibles	WinSymbole
_Signalisation	Texture
_Sources	WinSymbole/Texture*
_Transformateurs	WinSymbole

* Certains symboles de la famille de composants utilisent le mode WinSymbole et d'autres le mode texture. Il est nécessaire d'ouvrir le dossier :
C:\Users\Public\Documents\Elec-CAO\wrs-data\sym_textures pour connaître le mode de dessin d'un composant précis.

- Un symbole qui possède le mode de dessin par défaut 'WinSymbole' peut être décliné en un autre symbole uniquement avec WinSymbole.
- Un symbole qui possède le mode de dessin par défaut 'Texture' peut être décliné en un autre symbole avec une texture modifiée, ou avec une description WinSymbole.

Composants simulables

Paramétrage : disponible dans 'Modifier un objet' dans WinRelais



Les champs d'édition des paramètres sont accessibles à partir de l'icône 'Modifier un objet'

- Relais, relais temporisé, cadenceur
 - **v** = tension de la bobine [12 à 230 V],
 - **toff** = retard à la désactivation [0 à 30 s],
 - **ton** = retard à l'activation [0 à 30 s].
- Contacts temporisés
 - **toff** = retard à la désactivation [0 à 30 s],
 - **ton** = retard à l'activation [0 à 30 s].
- Signalisation
 - **v** = tension de la bobine [12 à 230 V],
 - **couleur** = couleur du voyant [rouge, vert, bleu, orange, blanc].
- Disjoncteur et disjoncteur DDR
 - **In** = calibre [0.5 à 100 A],
 - **retard** = retard intentionnel [0 à 10000 ms], permet la sélectivité des protections,
 - **courbe** = courbe de déclenchement [B, C, D],
 - **idn** = sensibilité du différentiel [10 à 10000 mA],
 - **retard_diff** = retard intentionnel du DDR [0 à 10000 ms], permet la sélectivité différentielle.
- Fusible
 - **In** = calibre fusible [0.5 à 100 A],
 - **retard** = retard intentionnel en ms (pratique pour ajuster la simulation),
 - **type** = type de cartouche [aM, gG].
- Relais thermique
 - **ith** = seuil thermique [0.1 à 100 A]
- Moteur alternatif
 - **Pu** = puissance utile moteur [90 W à 22 kW], (exemples : 125 W, 10 kW)
 - **nm** = vitesse nominale moteur [740 à 3000 rpm]
 - **cos** = cos(phi) moteur [0.5 à 1]
 - **rendement** = rendement moteur [0.5 à 1]
 - **Venroulement** = tension nominale aux bornes d'un enroulement [127 à 400 V]
 - **couplage** = couplage moteur [étoile, triangle]

- Moteur continu
 - **Pu** = puissance utile moteur [90 W à 22 kW], (exemples : 200 W, 10 kW)
 - **nm** = vitesse nominale moteur [740 à 3000 rpm],
 - **Vinduit** = tension d'induit [12 à 320 V],
 - **rendement** = rendement moteur [0.5 à 1].
- Source, alimentation stabilisée, transformateur
 - **pression** = pression air comprimé [0 à 10 bar],
 - **Vin** = tension d'entrée [12 à 400 V],
 - **Vout** = tension de sortie [12 à 400 V].
- Résistance de puissance
 - **pn** = puissance nominale [10 W à 22 kW], (exemples : 200 W, 1.5 kW)
 - **Venroulement** = tension nominale aux bornes d'un enroulement [230 ou 400 V].
- Potentiomètre
 - **valeur** = résistance du potentiomètre [1e-3 à 10e6 ohms], (exemples : 10 ohms, 10 k ohms, 10M ohms),
 - **alpha** = position curseur [0 à 100 %].
- Résistance simple
 - **valeur** = résistance [1e-3 à 10e6 ohms], (exemples : 10 ohms, 10k ohms, 10 M)
- Instruments de mesure
 - **type** = type de mesure [moyenne, efficace]
 - **afficher ici** permet de positionner précisément la valeur de la mesure sur le folio.

Signalétique :



protection électrique armée (disjoncteur et fusible)



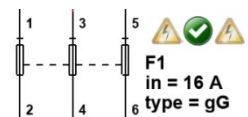
ouverture protection électrique sur surcharge ou court-circuit (disjoncteur et fusible)



ouverture protection sur courant de fuite (DDR)



indicateur de l'état individuel de cartouches fusibles. Exemple sur un porte-fusible tripolaire avec deux cartouches fusibles fondues.

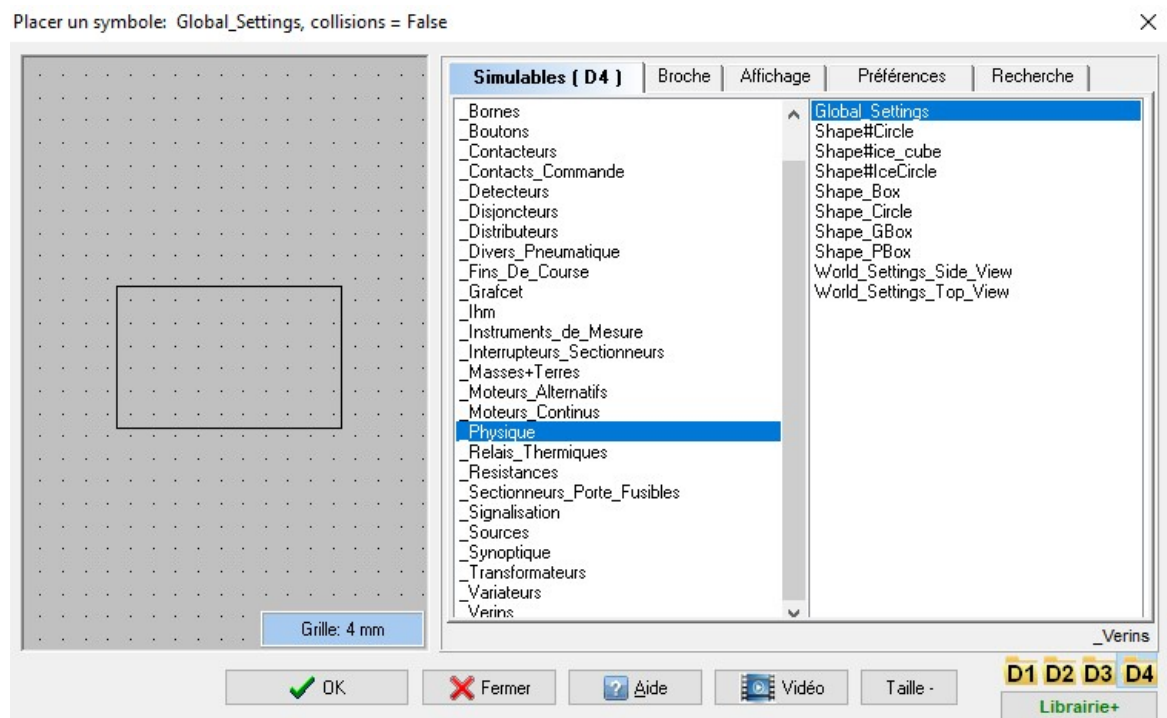


récepteur sous-alimenté



récepteur suralimenté

Blocs de configuration généraux



Accès aux blocs de configuration

Ce bloc ajuste certains détails d'affichage du folio courant, il est essentiellement utilisé à des fins de mise au point :

```
visuel folio
collisions = False
sensors_collisions = True
wr_equips = False
wrs_equips = True
terminals_nums = True
```

- **collisions** = (False ou True) : visualise les formes de collision des objets physiques (physiques et cinématiques),
- **sensors_collisions** = (False ou True) : visualise les formes de collision des détecteurs et des fins de course,
- **wrs_equips** = ((False ou True) : visualise les numéros des équipotentiels définis par le simulateur,

- **wr_equips** = ((False ou True) : visualise les numéros des équipotentiels définis dans WinRelais,
- **terminals_nums** = ((False ou True) : visualise les numéros de bornes des appareils.

```
Top View
gravity = 0
damping = 0

Side View
gravity = 9.81
damping = 1
```

Les blocs '**World_Settings**' définissent le comportement du moteur physique 2D. Il est possible d'ajuster le moteur en :

- vue de dessus : la gravité n'est pas prise en compte,
- vue de côté : la gravité est prise en compte.

On ne peut placer qu'un seul bloc '**World_Settings**' par folio. Il est possible d'avoir des vues différentes d'un folio à l'autre, par exemple :

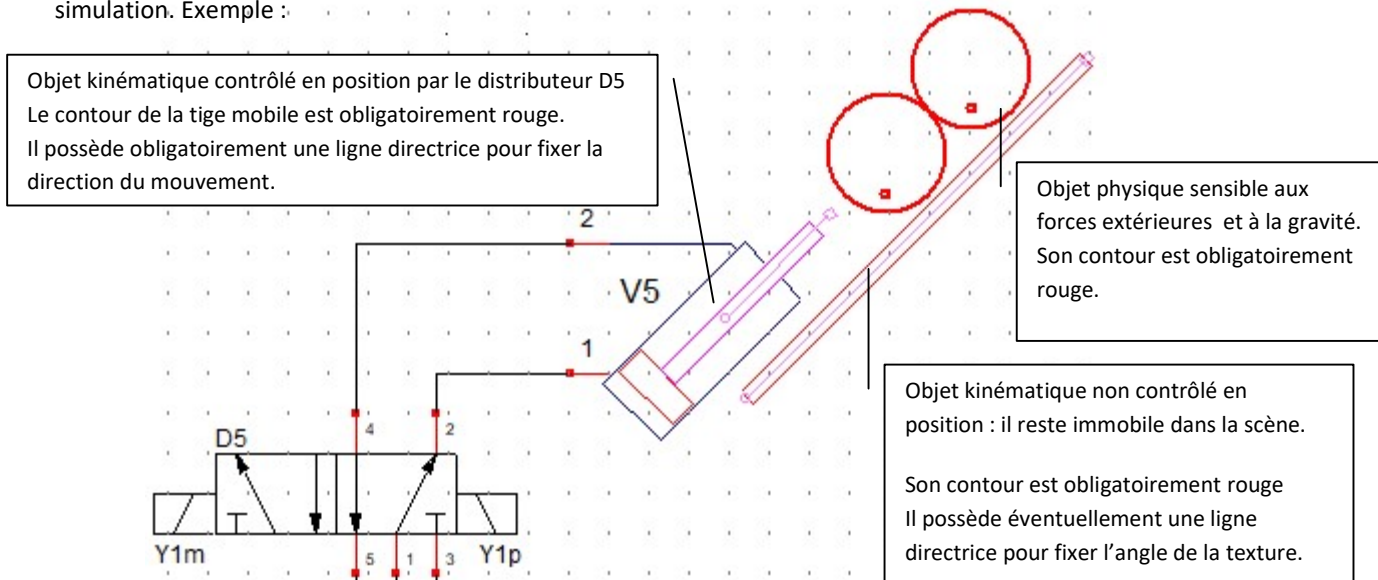
- un folio avec un espace physique décrit en vue de dessus,
- le folio suivant avec un espace physique en vue de côté.

Le coefficient de '**damping**', compris entre 0 et 1, simule la friction de l'air qui s'exerce sur un mobile.

Synoptique et animation

Préambule

Le simulateur offre la possibilité de créer des petites parties opératives contrôlables avec les actionneurs de l'installation. Il intègre un moteur physique 2D pour apporter un certain réalisme à la simulation. Exemple :



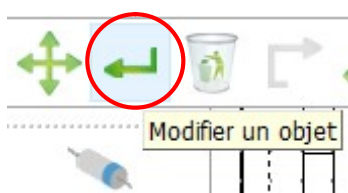
Cet exemple montre deux bidons cylindriques vus de côté posés sur une rampe inclinée. Lorsque la tige du vérin sort, les deux bidons sont remontés vers le haut de la rampe.

Lorsque la tige du vérin rentre, les deux bidons roulent (ou glissent selon le coefficient de frottement appliqué) vers le bas de la rampe en restant finalement bloqués en position par le nez du vérin.

Deux types d'objets sont utilisés par le moteur physique :

- les objets physiques qui sont soumis aux forces extérieures et à la gravité. Ils ont notamment une masse en kilogrammes
- les objets cinématiques ont leurs positions dans la scène contrôlés directement par le programme. Ils sont insensibles aux forces extérieures et donc non soumis aux lois de la physique. Les objets physiques peuvent entrer en collision avec les objets cinématiques.

Comment décrire un objet physique ou cinématique



Les champs d'édition des paramètres sont accessibles à partir de l'icône 'Modifier un objet'

Champs 1 à 4	Champs 5 à 8
Champs 1 à 4	Champs 5 à 8
Nom: bidon1	Champ 5: friction = 2
Valeur: pas_de_rotation = False	Champ 6: nom_texture = quebec
Champ 3: dynamique = True	Champ 7: id_collision = *
Champ 4: masse = 1 kg	Champ 8: []
Symbole: Ω Σ Ψ	
Information	
Dossier: _Physique	
Fichier: Shape#Circle.xsy	
ID: -1	

dynamique = True -> l'objet est de type 'physique', False -> l'objet est de type 'kinématique'

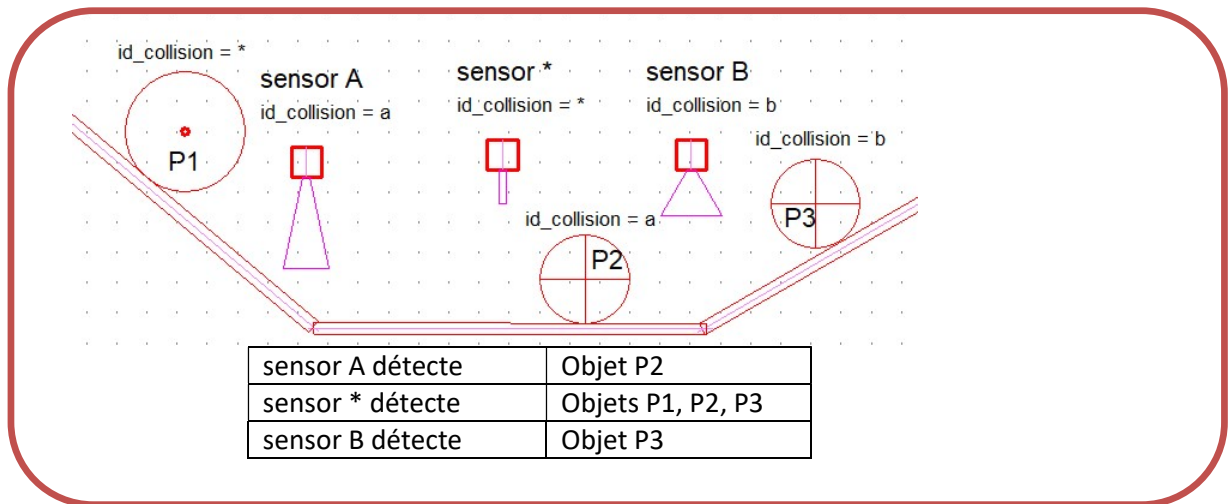
masse = masse en kg de l'objet physique [1 à 100 kg]

friction = coefficient de friction [0 à 100] appliqué aux objets physiques

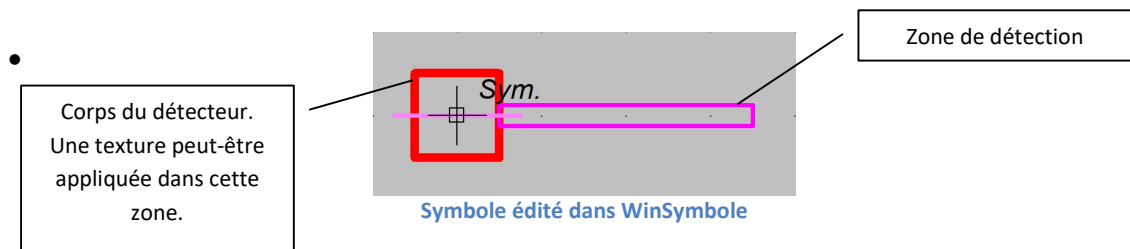
pas_de_rotation = applique une masse et une valeur d'inertie considérable à l'objet physique.

id_collision = identificateur de collision, permet de filtrer les collisions par groupes d'objets. L'identificateur vaut '*' par défaut. Essentiellement utilisé pour la détection d'objets par les détecteurs et les fins de course.

- un **id_collision** permet de rendre sensible un détecteur/fin de course à un groupe d'objets précis.
Par exemple :

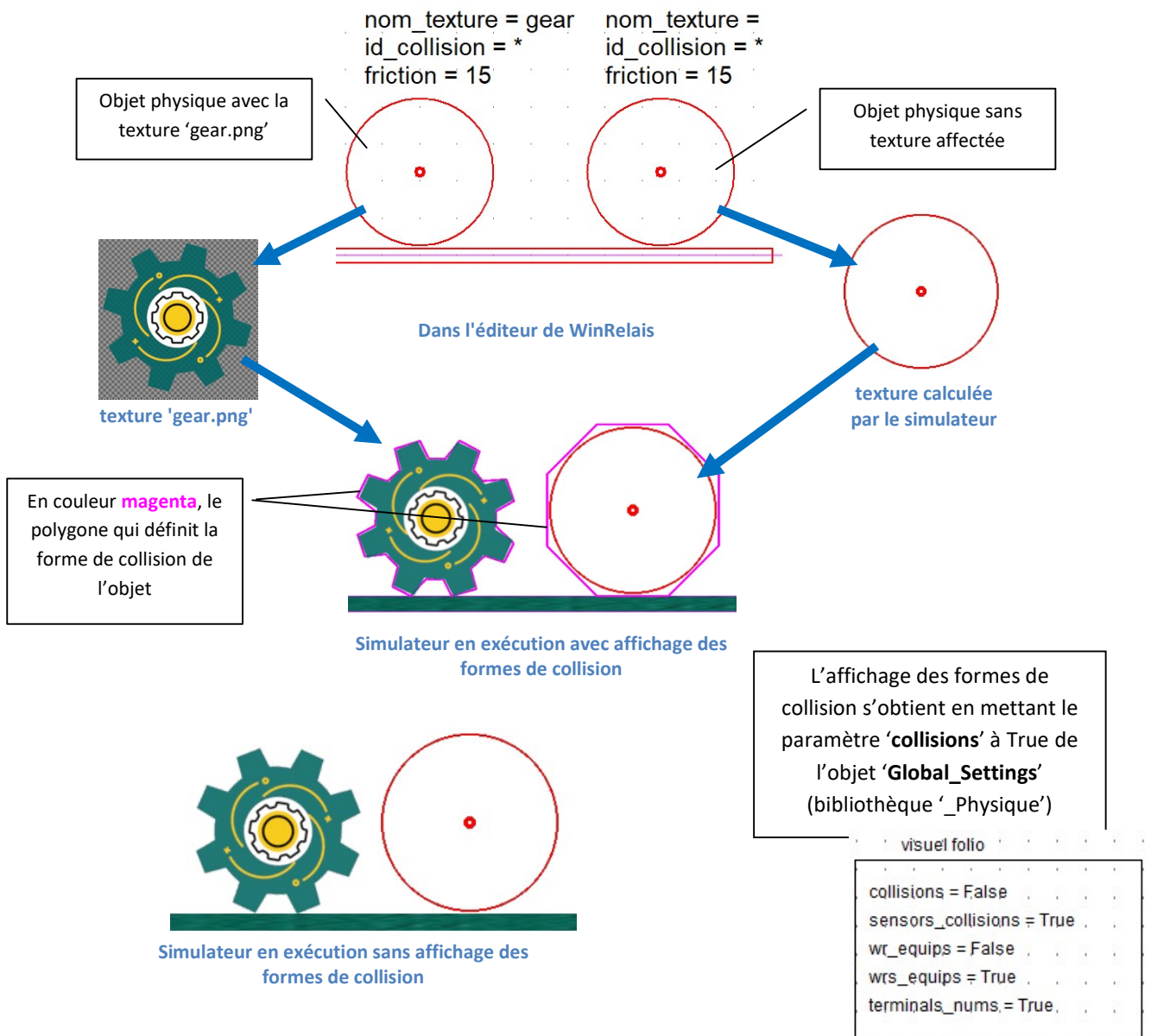


- Un détecteur/fin de course est un objet kinématique particulier disponible dans la bibliothèque **_Synoptique**
- Un détecteur/fin de course qui a **id_collision = *** détecte tous les objets qui entre dans sa zone de détection.
- La zone de détection d'un détecteur/fin de course est définie par un contour de couleur **magenta** (255,0,255) . Par exemple :

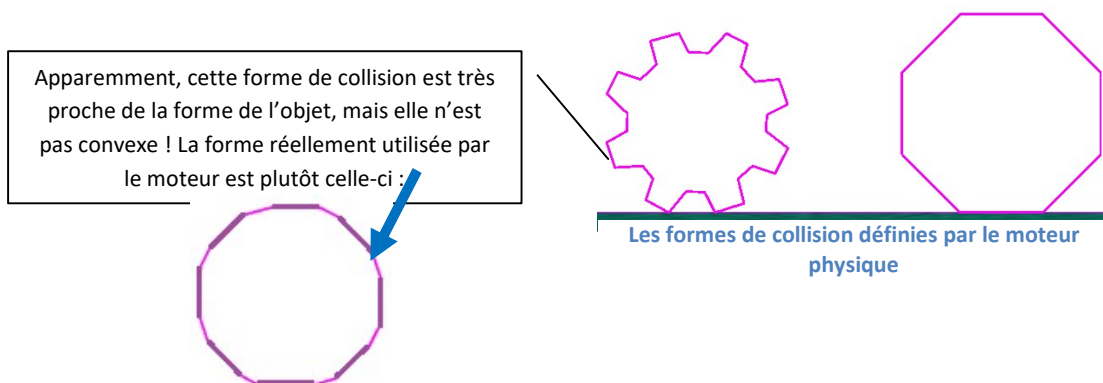


nom_texture = nom de la texture appliquée à l'objet

- Si un nom de texture est défini, l'application recherche la texture (au format **png** seulement) dans le dossier C:\Users\Public\Documents\Elec-CAO\wrs-data\textures_utilisateur.
- Si le nom de la texture n'est pas défini ou si le fichier texture n'est pas trouvé, une texture calculée par le programme sera plaquée sur l'objet. Par exemple :



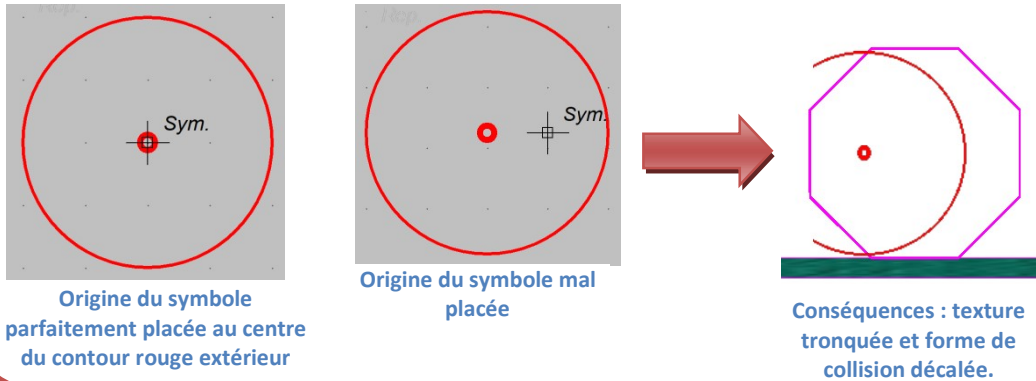
La silhouette de la texture permet de définir automatiquement la forme de collision de l'objet. Le moteur physique utilise la partie transparente de la texture pour délimiter la forme de collision sous la forme d'un polygone **convexe** comme montré ci-dessous :



Quand la texture n'est pas définie, le moteur utilise le bord extérieur coloré en **rouge** de la forme affichée dans WinRelais pour construire une texture.

- Contrainte de dessin dans **WinSymbole**

Il faut placer l'origine du symbole au barycentre du contour **rouge** extérieur pour garantir le centrage de la texture calculée :



Comment décrire une liaison glissière (slider) ?

Les liaisons glissières sont disponibles dans la bibliothèque **_Synoptique** et sont préfixées avec le mot clef **slider** :



Extrait de la bibliothèque des symboles simulables rangée en D4

- Description des paramètres disponibles :

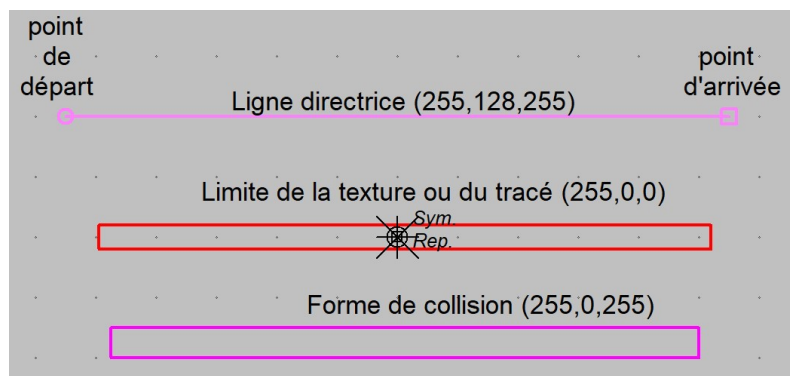
Champs 1 à 4	
Champs 1 à 4	
Nom	table1
Valeur	contrôle_direction = M1
Champ 3	contrôle_vitesse = M1
Champ 4	vitesse = 40 %
Champs 5 à 8	
Champs 5 à 8	
Champ 5	friction = 50
Champ 6	id_collision = *
Champ 7	parent =
Champ 8	nom_texture =

contrôle_direction = désigne l'objet moteur utilisé pour contrôler le mobile en direction,
contrôle_vitesse = désigne l'objet moteur utilisé pour contrôler le mobile en vitesse, en général le même moteur est utilisé pour la direction et la vitesse,
vitesse = fixe la vitesse de déplacement du mobile en pourcentage de la vitesse du moteur (transcrite en vitesse linéaire). [0 à 100%],
friction = fixe la friction de la surface du mobile [0 à 100],
id_collision = identificateur de collision, permet de filtrer les collisions par groupes d'objets. L'identificateur vaut ***** par défaut. Essentiellement utilisé pour la détection d'objets par les détecteurs et les fins de course.
parent = désigne un parent qui ne peut-être qu'une autre liaison glissière,
nom_texture = nom de la texture éventuelle qui peut-être appliquée au mobile.

- Éléments constitutifs d'une liaison glissière éditables dans WinSymbole :
Ces éléments sont différenciés dans WinSymbole grâce à des couleurs précises :
 - **(255,128,255)** identifie la ligne directrice et les limites de déplacement (symboles carré et cercle),
 - **(255,0,0)** identifie le contour extérieur du mobile. Ce contour doit être un polygone convexe unique,
 - **(255,0,255)** identifie la forme de collision optionnelle qui doit être un polygone convexe unique. Si ce polygone n'est pas défini, le moteur physique calcule la forme de collision à partir du contour extérieur rouge.

L'origine du symbole doit être soigneusement :

- placée au barycentre du contour extérieur rouge,
- sur la ligne directrice.



Les différentes primitives nécessaires pour décrire une liaison glissière dans WinSymbole

Exemples

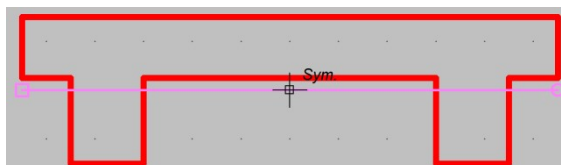
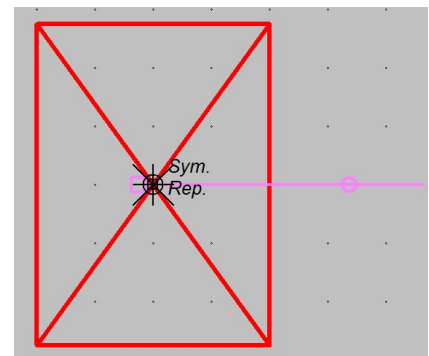
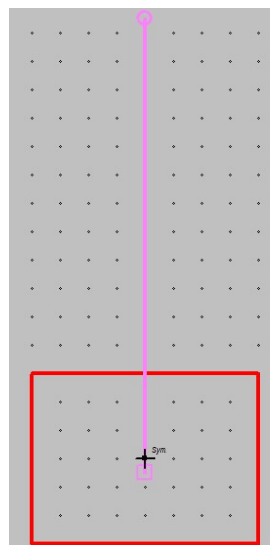


Table mobile. Voir '3 - demo_CyclePendulaire'



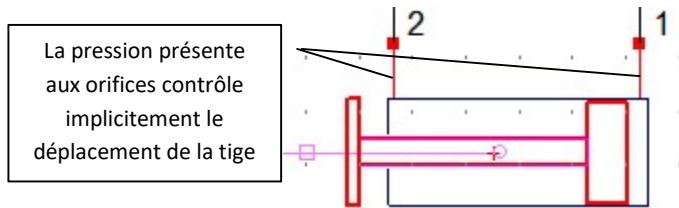
Porte cabine. Voir '1 - demo_Ascenseur'

Cabine. Voir '1 - demo_Ascenseur'



- On peut placer le mobile à une position de départ précise selon les besoins. Il faut vérifier que l'origine du mobile est bien placée sur la ligne directrice et entre le point de départ et d'arrivée.
- On peut orienter la ligne directrice d'un certain angle selon les besoins avec l'outil rotation de WinSymbole. Attention au placement du centre de rotation !

Un cas particulier de glissière : le vérin



Champs 1 à 4

Champs 1 à 4

Nom: V2

Valeur: pression= 6 (0-10bar)

Champ 3: vitesse = 60 (0-100%)

Champ 4: []

pression = valeur de la pression de l'air comprimé [0 à 10 bar]
vitesse = coefficient de vitesse de déplacement de la tige [0 à 100%]. La vitesse de déplacement est également liée à la pression.

Champs 5 à 8

Champs 5 à 8

Champ 5: id_collision = *

Champ 6: nom_texture =

Champ 7: []

Champ 8: []

id_collision = identificateur de collision, permet de filtrer les collisions par groupes d'objets. L'identificateur vaut '*' par défaut. Essentiellement utilisé pour la détection d'objets par les détecteurs et les fins de course. Voir les détails à la [page 15](#).

nom_texture = nom de la texture éventuellement appliquée à la tige du vérin. Voir les contraintes associées aux textures à la [page 15](#).

Champs accessibles à partir de la commande 'Modifier un objet'

Liaison glissière et parenté

Tous les objets kinématiques peuvent être attachés à une liaison glissière. Ceci est extrêmement pratique pour créer des formes de collision non convexes, par exemple :

Le nez de la tige est un objet kinématique attaché à la tige du vérin grâce au paramètre parent :

Champs 1 à 4

Nom: nez_pousseur

Valeur: parent = V2

Champ 3: id_collision = *

Champ 4: nom_texture = pousseur

Symbole: Ω Σ Φ

Information

Dossier: _Synoptique
 Fichier: kinematic#pousseur.xsy
 ID: -1

Champs 1 à 4

Nom: V2

Valeur: pression= 6 (0-10bar)

Champ 3: vitesse = 60 (0-100%)

Champ 4: []

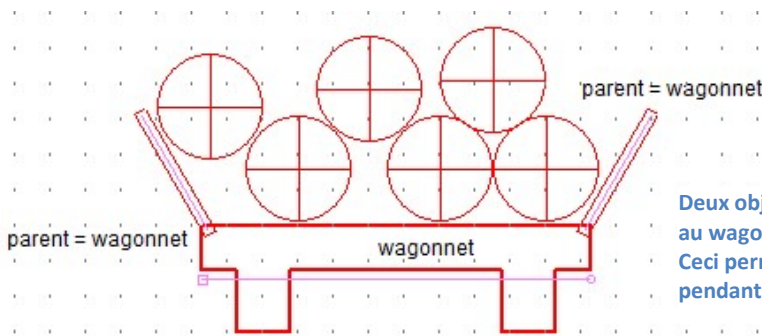
Symbole: Ω Σ Φ

Information

Dossier: _Verins
 Fichier: VerinDE#long.xsy
 ID: -1

Forme de collision résultante. La parenté permet de combiner les formes de collision pour n'en former qu'une.

Il est possible d'attacher plusieurs objets kinématiques :



Deux objets kinématiques sont attachés au wagonnet pour construire ses flancs. Ceci permettra de 'contenir' les six bidons pendant la simulation.

Comment décrire un convoyeur

Les convoyeurs sont disponibles dans la bibliothèque **_Synoptique** et sont préfixées avec le mot clef **conveyor** :



Champs 1 à 4	
Champs 1 à 4	
Nom	tapis0
Valeur	controle_vitesse = M1
Champ 3	controle_direction =
Champ 4	vitesse = 100 %
Champs 5 à 8	
Champs 5 à 8	
Champ 5	nom_texture = convoyeu
Champ 6	parent =
Champ 7	id_collision = *
Champ 8	

controle_direction = désigne l'objet moteur utilisé pour contrôler la direction du convoyeur,

controle_vitesse = désigne l'objet moteur utilisé pour contrôler la vitesse du convoyeur, en général le même moteur est utilisé pour la direction et la vitesse,

vitesse = fixe la vitesse du convoyeur en pourcentage de la vitesse du moteur (transcrite en vitesse linéaire). [0 à 100%],

friction = fixe la friction du convoyeur [0 à 100],

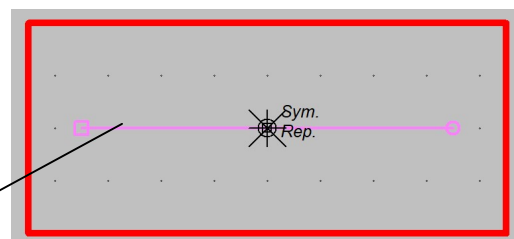
id_collision = identificateur de collision, permet de filtrer les collisions par groupes d'objets. L'identificateur vaut '*' par défaut.

Essentiellement utilisé pour la détection d'objets par les détecteurs et les fins de course. Voir les détails à la [page 15](#).

parent = désigne un parent qui ne peut-être qu'une liaison glissière,

nom_texture = nom de la texture éventuelle qui peut-être appliquée au convoyeur. Voir les contraintes associées aux textures à la [page 15](#).

La ligne directrice fixe la direction qui sera prise par les objets physiques posés sur le convoyeur



Convoyeur en vue de dessus édité dans WinSymbole. La texture appliquée ne sera visible qu'à la simulation

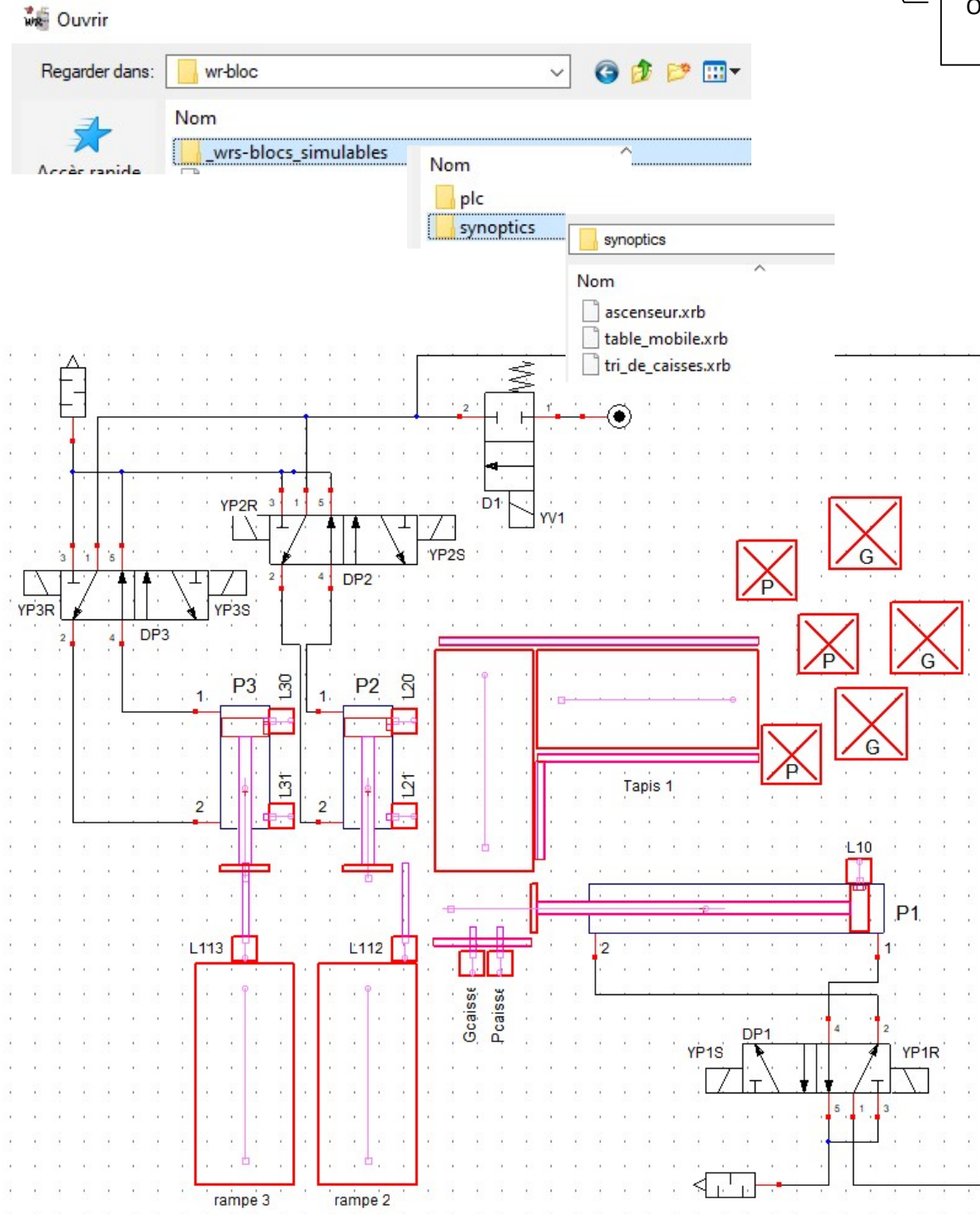
Sous-ensembles synoptiques

On peut sauver et récupérer des synoptiques complets grâce aux commandes de blocs de WinRelais.



Sauver un bloc

Ouvrir un bloc



Tout est rassemblé dans le même bloc : pré-actionneurs, actionneurs, capteurs, éléments de partie opérative et la matière d'œuvre (les caisses à transiter)

Variateur de vitesse

WRsimulateur permet d'insérer des variateurs de vitesse dans les folios. La bibliothèque comprend actuellement (version 1.1) des objets variateurs monophasés et triphasés type ATV31 de SchneiderElectric.

L'exemple **18-demo_variateurATV31** illustre une mise en œuvre d'un variateur monophasé pour contrôler la vitesse de déplacement d'une table mobile.

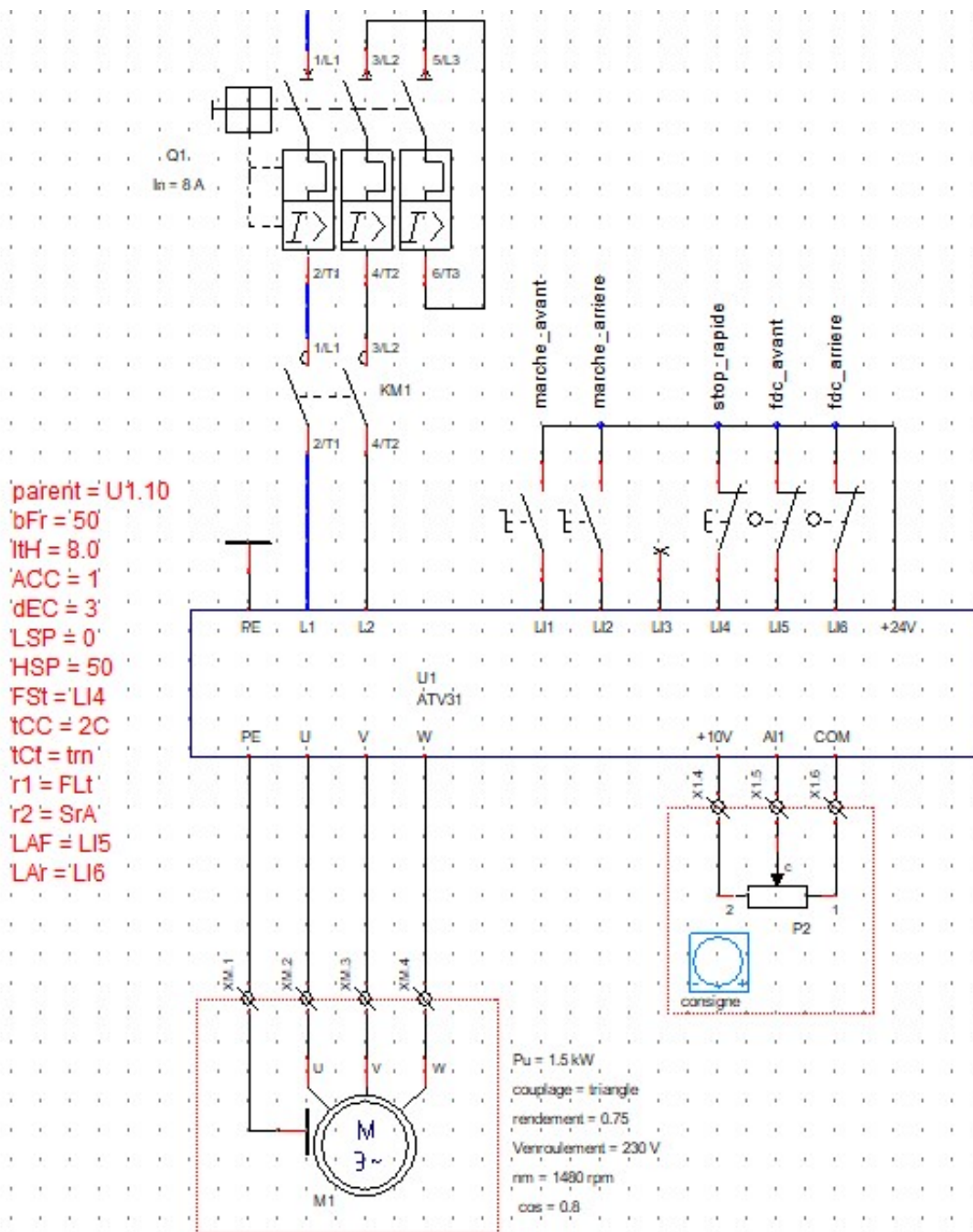
Une liste éditable des paramètres essentiels de configuration est attachée au variateur. Les numéros de page mentionnés plus bas font référence au document de SchneiderElectric ATV31 Guide de programmation du 05/10/2009:

ATV31_programming_manual_FR_1624588_04.pdf

Liste des paramètres ATV31 implantés dans le simulateur

parent = U1.10A A placer en début de liste pour préciser que cette liste est attachée au variateur U1.
La valeur 10A fixe le courant nominal du variateur. (valeur par défaut 63 A si non défini)

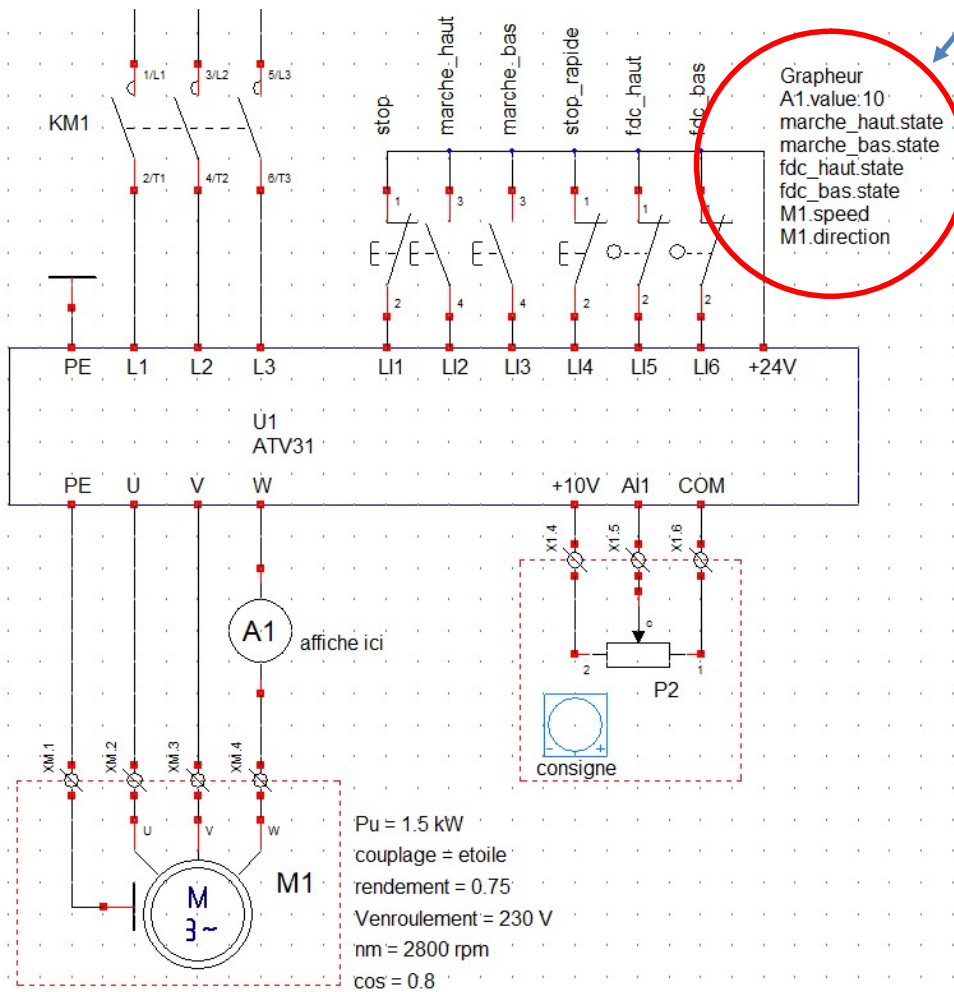
bFr = 50	Fréquence standard moteur (p.23) [50,60]
ItH = 10.0	Protection thermique du moteur (p.19) (0.2 à 1.5.In), In = courant nominal variateur Il faut régler itH au courant nominal du moteur entraîné.
CLI = 15.0	Limitation de courant (p.xx) (0.25 à 1.5.In)
ACC = 3.0	Temps de la rampe de d'accélération (p.19) 0.1s à 3276s
dEC = 3.0	Temps de la rampe de décélération (p.19) 0.1s à 3276s
LSP = 0	Petite vitesse (p.19) 0 Hz à HSP
HSP = 50	Grande vitesse (p.19) LSP à bFR
PS2 = nO	2 vitesses présélectionnées (p.50) [nO,LI3,LI4,LI5,LI6]
PS4 = nO	4 vitesses présélectionnées (p.50) [nO,LI3,LI4,LI5,LI6]
PS8 = nO	8 vitesses présélectionnées (p.50) [nO,LI3,LI4,LI5,LI6]
SP2 = 10	2 ^e Vitesse présélectionnée (p.51) 0 Hz à HSP
SP3 = 15	3 ^e Vitesse présélectionnée (p.51) 0 Hz à HSP
SP4 = 20	4 ^e Vitesse présélectionnée (p.51) 0 Hz à HSP
SP5 = 25	5 ^e Vitesse présélectionnée (p.51) 0 Hz à HSP
SP6 = 30	6 ^e Vitesse présélectionnée (p.51) 0 Hz à HSP
SP7 = 35	7 ^e Vitesse présélectionnée (p.51) 0 Hz à HSP
SP8 = 40	8 ^e Vitesse présélectionnée (p.51) 0 Hz à HSP
tCC = 2C	Commande 2fils / 3fils (p.27) [2C, 3C]
tCt = trn	Type de commande si 2fils choisi (p.27) [LEL, trn, PFO]
r1 = FLt	Relais r1 (p.28) [nO,FLt,rUn,FtA,FLA,CtA,SrA,tSA,APL,LI1,LI2,LI3,LI4,LI5,LI6]
r2 = nO	Relais r2 (p.29) [nO,FLt,rUn,FtA,FLA,CtA,SrA,tSA,bLC,APL,LI1,LI2,LI3,LI4,LI5,LI6]
FSt = nO	Arrêt rapide sur état logique 0 (p.45) [nO, LI1, LI2, LI3, LI4, LI5, LI6]
LAF = nO	Fin de course sens avant (p.64) [nO, LI1, LI2, LI3, LI4, LI5, LI6]
LAr = nO	Fin de course sens avant (p.64) [nO, LI1, LI2, LI3, LI4, LI5, LI6]
LAS = nSt	Type d'arrêt en fin de course (p.64) [rMP, FSt, nSt]



ATV31 avec sa table de paramètres. Mode deux fils sur changement d'état avec gestion de l'arrêt par les fins de courses fdc_avant et fdc_arriere.

Utilisation du grapheur

Le module grapheur permet de représenter l'évolution de grandeurs physiques (jusqu'à 16 maximum) indiquées dans une zone de texte. Cette zone de texte doit commencer par le mot-clef 'Grapheur'. Par exemple :



state : état actionné/non actionné des composants boutons-poussoirs et fins de course désignés.

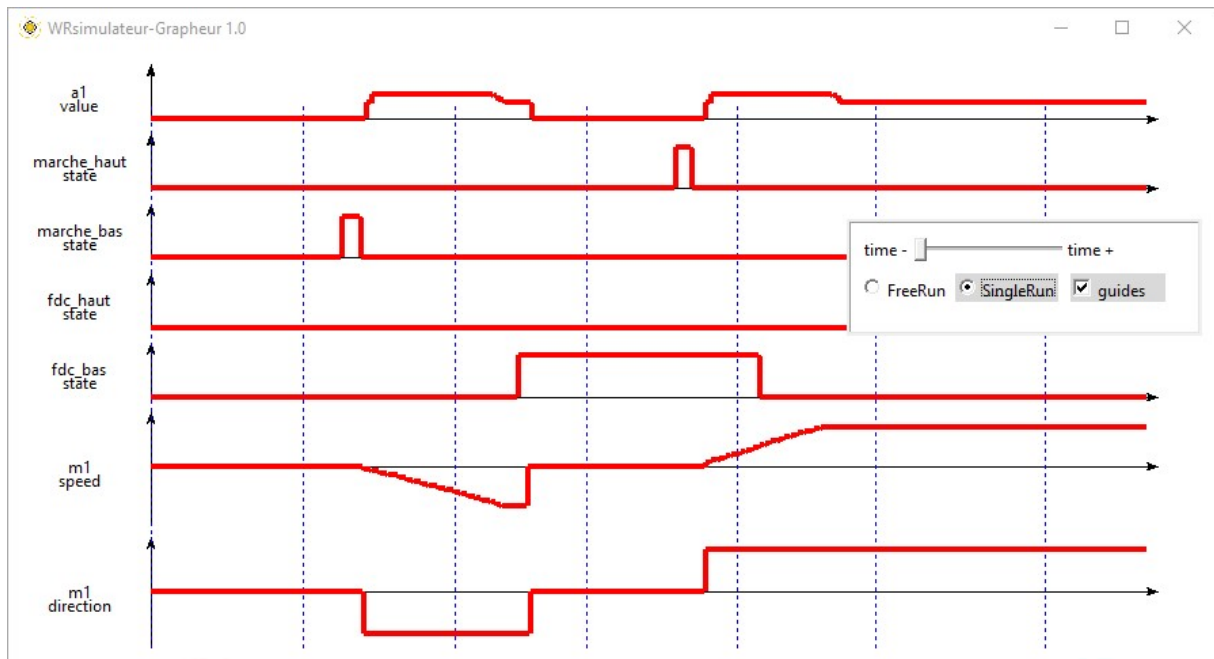
Grapheur
 A1.value.10
 marche_haut.state
 marche_bas.state
 fdc_haut.state
 fdc_bas.state
 M1.speed
 M1.direction

value : valeur mesurée par l'ampèremètre A1, calibre 10 A.

speed : Vitesse de rotation en % de $N_{nominal}$ du moteur M1.

direction : sens de rotation du moteur M1

Les données du grapheur en détail. L'ordre des lignes correspond à l'ordre d'affichage des chronogrammes.



Relevé correspondant aux réglages fixés dans la zone de texte débutant par le mot clef 'Grapheur'. L'intervalle entre deux lignes verticales en traits pointillés correspond toujours à **1 seconde**.

Intégration d'un automate programmable industriel (API)

Préambule

WRsimulateur offre la possibilité d'intégrer un automate dans vos folios. Les exemples fournis utilisent le logiciel (gratuit) EcoStruxure Basic Expert de SchneiderElectric qui permet la programmation et la simulation des automates de la famille M221.

La liaison entre WRsimulateur et l'API virtuel est réalisée selon le protocole MODBUS/TCPIP où l'API est l'application serveur, et WRsimulateur l'application cliente.

La bibliothèque **_API** du simulateur propose un objet de configuration MODBUS/TCPIP :

MODBUS 23 TCP/IP

```
port = 502
exchange_adress_table = 400
unit_ID = 111
```

- port : numéro du port utilisé pour les échanges. 502 par défaut
- exchange_adress_table : définit la première adresse des mots en mémoire API qui range les données d'E/S qui seront échangées via MODBUS requête 23. %MW400 est l'adresse du 1^{er} registre par défaut
- unit_ID : numéro d'identification de client. 111 par défaut

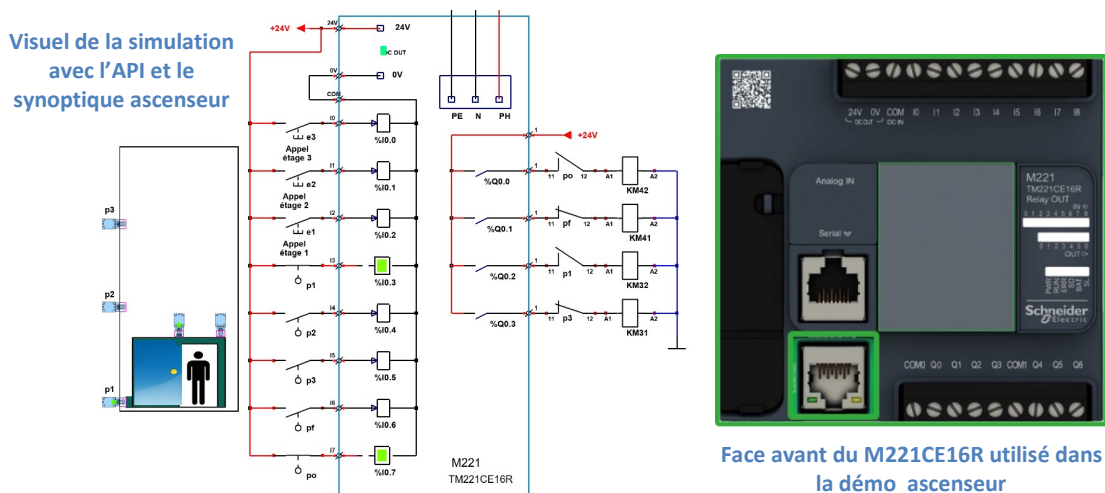
- ✓ Le client autorise pour l'instant les échanges MODBUS en requête 23 uniquement (Read/Write Multiples registers).
- ✓ La requête 23 est une fonction MODBUS/TCPIP avancée, qui permet d'optimiser singulièrement les durées d'échange client/serveur. Elle est supportée par la majorité des serveurs MODBUS des API modernes.
- ✓ Si l'objet de configuration MODBUS n'est pas placé dans votre projet, les valeurs utilisées par défaut sont : port = 502, exchange_adress_table = 400 et unit_ID = 111.

Exemple ascenseur

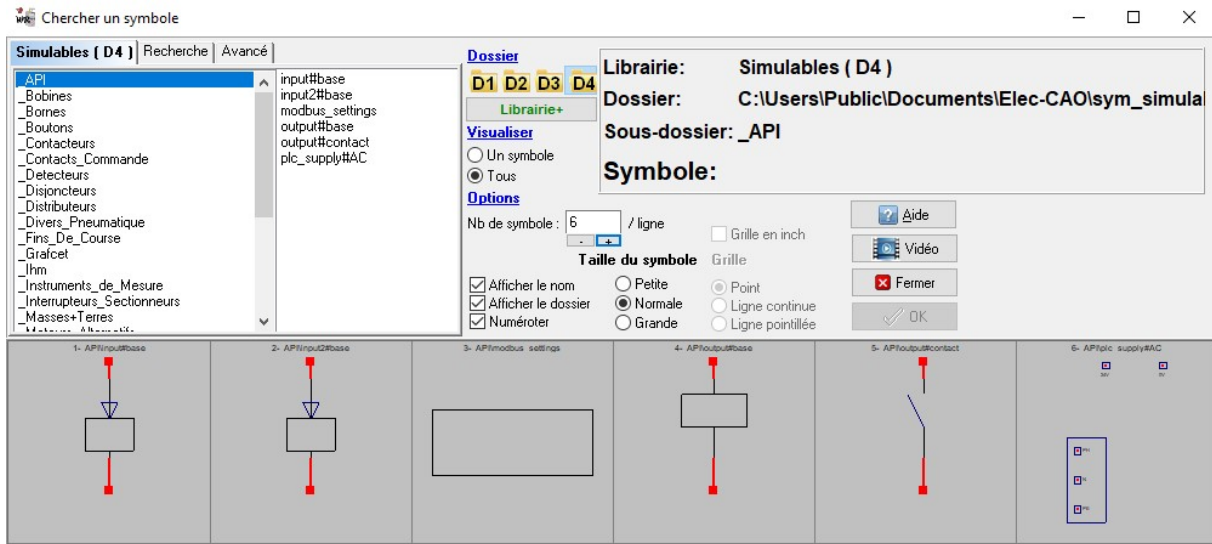
(fichiers 15-demo_ascenseur_M221.xrs et 15-demo_ascenseur_M221.smbp)

Préparation côté WRsimulateur :

L'exemple ci-dessous utilise l'API virtuel de l'outil de programmation gratuit EcoStruxure Machine Expert-Basic édité par la société Schneider Electric.



- ✓ La commande de l'ascenseur nécessite 8 entrées et 4 sorties TOR. On peut utiliser un M221 type TM221CE16R qui dispose de 9 entrées TOR et de 7 sorties à relais.
- ✓ Adressage des entrées TOR : %I0.0, %I0.1, %I0.2, %I0.3, %I0.4, %I0.5, %I0.6, %I0.7, %I0.8
- ✓ Adressage des sorties : %Q0.0, %Q0.1, %Q0.2, %Q0.3, %Q0.4, %Q0.5, %Q0.6
- ✓ Le simulateur propose deux objets qui permettent de placer des entrées et des sorties TOR sur le folio.
- ✓ L'exemple étudié implante 8 objets d'entrée et 4 objets de sortie en les identifiant avec les adresses API standards.



Détail des objets disponibles pour décrire un automate.

L'affectation des E/S retenue est la suivante :

Entrées

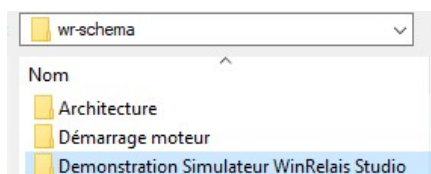
adresse	mnémonique	commentaire
%I0.0	E3	appel étage 3
%I0.1	E2	appel étage 2
%I0.2	E1	appel étage 1
%I0.3	P1	cabine à l'étage 1
%I0.4	P2	cabine à l'étage 2
%I0.5	P3	cabine à l'étage 3
%I0.6	PF	porte cabine fermée
%I0.7	PO	porte cabine ouverte

Sorties

adresse	mnémonique	commentaire
%Q0.0	OVRIR	Ouvrir porte
%Q0.1	FERMER	Fermer porte
%Q0.2	DESCENDRE	Descendre cage
%Q0.3	MONTER	Monter cage

Préparation côté EcoStruxure Machine Expert-Basic

Le programme complet [15-demo_ascenseur_M221.smb](#) est disponible dans le dossier :



WRsimulateur supporte jusqu'à :

- 64 entrées TOR répartis en deux racks de 32 voies chacun,
 - 64 sorties TOR répartis en deux racks de 32 voies chacun.
- Mappage complet avec l'adresse de la table d'échange ajustée à %MW400 :

- %MW400 à %MW403 correspondent aux registres MODBUS de sortie :

%Q0.15 %Q0.14 %Q0.13 %Q0.12 %Q0.11 %Q0.10 %Q0.9 %Q0.8 %Q0.7 %Q0.6 %Q0.5 %Q0.4 %Q0.3 %Q0.2 %Q0.1 %Q0.0																
x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	%MW400
%Q0.31 %Q0.30 %Q0.29 %Q0.28 %Q0.27 %Q0.26 %Q0.25 %Q0.24 %Q0.23 %Q0.22 %Q0.21 %Q0.20 %Q0.19 %Q0.18 %Q0.17 %Q0.16																
x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	%MW401
%Q1.15 %Q1.14 %Q1.13 %Q1.12 %Q1.11 %Q1.10 %Q1.9 %Q1.8 %Q1.7 %Q1.6 %Q1.5 %Q1.4 %Q1.3 %Q1.2 %Q1.1 %Q1.0																
x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	%MW402
%Q1.31 %Q1.30 %Q1.29 %Q1.28 %Q1.27 %Q1.26 %Q1.25 %Q1.24 %Q1.23 %Q1.22 %Q1.21 %Q1.20 %Q1.19 %Q1.18 %Q1.17 %Q1.16																
x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	%MW403

- %MW404 à % MW407 correspondent aux registres MODBUS d'entrée :

%I0.15 %I0.14 %I0.13 %I0.12 %I0.11 %I0.10 %I0.9 %I0.8 %I0.7 %I0.6 %I0.5 %I0.4 %I0.3 %I0.2 %I0.1 %I0.0																
x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	%MW404
%I0.31 %I0.30 %I0.29 %I0.28 %I0.27 %I0.26 %I0.25 %I0.24 %I0.23 %I0.22 %I0.21 %I0.20 %I0.19 %I0.18 %I0.17 %I0.16																
x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	%MW405
%I1.15 %I1.14 %I1.13 %I1.12 %I1.11 %I1.10 %I1.9 %I1.8 %I1.7 %I1.6 %I1.5 %I1.4 %I1.3 %I1.2 %I1.1 %I1.0																
x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	%MW406
%I1.31 %I1.30 %I1.29 %I1.28 %I1.27 %I1.26 %I1.25 %I1.24 %I1.23 %I1.22 %I1.21 %I1.20 %I1.19 %I1.18 %I1.17 %I1.16																
x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	%MW407

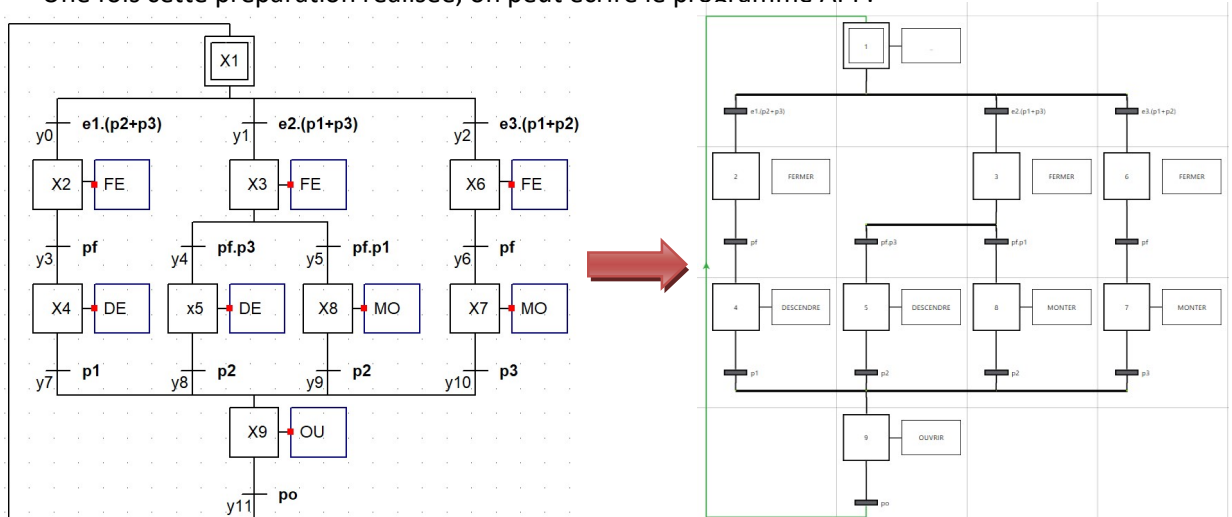
- Mise en correspondance des tableaux d'entrées/sorties avec la table d'échange ajustée sur l'adresse de départ %MW400 :

adresse	Mappage MODBUS	mnémonique	commentaire
%I0.0	%MW404 :X0	E3	appel étage 3
%I0.1	%MW404 :X1	E2	appel étage 2
%I0.2	%MW404 :X2	E1	appel étage 1
%I0.3	%MW404 :X3	P1	cabine à l'étage 1
%I0.4	%MW404 :X4	P2	cabine à l'étage 2
%I0.5	%MW404 :X5	P3	cabine à l'étage 3
%I0.6	%MW404 :X6	PF	porte cabine fermée
%I0.7	%MW404 :X7	PO	porte cabine ouverte
adresse	Mappage MODBUS	mnémonique	commentaire
%Q0.0	%MW400 :X0	OUVRIR	Ouvrir porte
%Q0.1	%MW400 :X1	FERMER	Fermer porte
%Q0.2	%MW400 :X2	DESCENDRE	Descendre cage
%Q0.3	%MW401 :X3	MONTER	Monter cage

- Dans l'onglet **Programmation -> Outils -> Liste de symboles**, on trouve l'assignation des entrées/sorties qui correspond aux tableaux précédents.

Utilisé	Adresse	Symbole	Commentaire
<input type="checkbox"/>	%MW400:X0	OUVRIR	Ouvrir porte
<input type="checkbox"/>	%MW400:X1	FERMER	Fermer porte
<input type="checkbox"/>	%MW400:X2	DESCENDRE	Descendre cage
<input type="checkbox"/>	%MW400:X3	MONTER	Monter cage
<input type="checkbox"/>	%MW404:X0	E3	appel étage 3
<input type="checkbox"/>	%MW404:X1	E2	appel étage 2
<input type="checkbox"/>	%MW404:X2	E1	appel étage 1
<input type="checkbox"/>	%MW404:X3	P1	cage à l'étage 1
<input type="checkbox"/>	%MW404:X4	P2	cage à l'étage 2
<input type="checkbox"/>	%MW404:X5	P3	cage à l'étage 3
<input type="checkbox"/>	%MW404:X6	PF	porte fermée
<input type="checkbox"/>	%MW404:X7	PO	porte ouverte

- Une fois cette préparation réalisée, on peut écrire le programme API :





Tâche maître

- 1 - GPN
 - Etape 1 - _
 - Etape 2 - FERMER
 - Etape 3 - FERMER
 - Etape 4 - DESCENDRE
 - Etape 5 - DESCENDRE
 - Etape 6 - FERMER
 - Etape 7 - MONTER
 - Etape 8 - MONTER
 - Etape 9 - OUVRIR
- Transitions
 - Tr1 - e2.(p1+p3)
 - Tr2 - e1.(p2+p3)
 - Tr3 - pf
 - Tr4 - pf.p3
 - Tr5 - e3.(p1+p2)
 - Tr6 - pf
 - Tr7 - pf.p1
 - Tr8 - p1
 - Tr9 - p2
 - Tr10 - p2
 - Tr11 - p3
 - Tr12 - po
- 2 - POST
 - Rung0 - FERMER
 - Rung1 - OUVRIR
 - Rung2 - MONTER
 - Rung3 - DESCENDRE

- On peut ensuite procéder à la mise en service de l'API en activant les boutons **Lancer le simulateur** et **Démarrer le contrôleur** dans l'onglet **Mise en service** d'EcoStruxure.
 - Il faut démarrer le contrôleur

	IN	OUT	ANA	CART 1
PWR	0	8	0	
RUN	1	1	0	
ERR	2	2	0	
STAT	3	3	1	
	4	4		
	5	5		
	6	6		
	7			

Buttons: Arrêter contrôleur, Démarrer contrôleur, Lancer le simulateur, Arrêter le simulateur

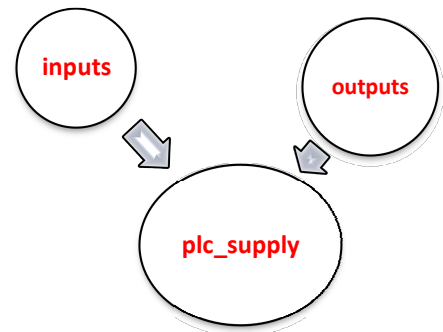
- Pensez bien à démarrer le contrôleur **AVANT** de lancer WRsimulateur sinon vous risquez d'obtenir les messages suivants et une communication MODBUS incorrecte :



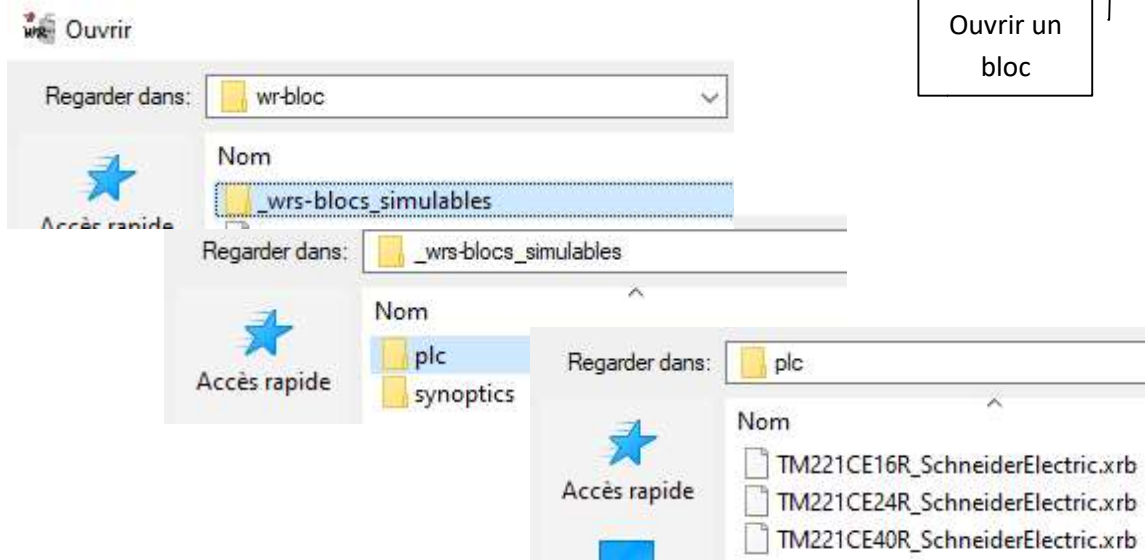
Principe de construction d'un objet API

Un automate est construit avec les objets élémentaires :

- **input#** pour les entrées TOR,
- **output#** pour les sorties TOR
- **plc_supply#** pour l'alimentation de l'API



- Toutes les E/S ont pour parent l'objet **plc_supply** afin d'asservir l'alimentation de ces E/S à l'alimentation de l'automate.
- Le serveur MODBUS est implicitement attaché à l'automate en place dans le schéma.
- On ne peut placer qu'un seul automate dans le schéma
- Des exemples d'automates sont disponibles dans le dossier wr-bloc à partir de la commande 'Ouvrir un bloc' de WinRelais



Toutes ces empreintes d'automate sont ajustables et modifiables selon les besoins

