AUTOMGEN *Junior*

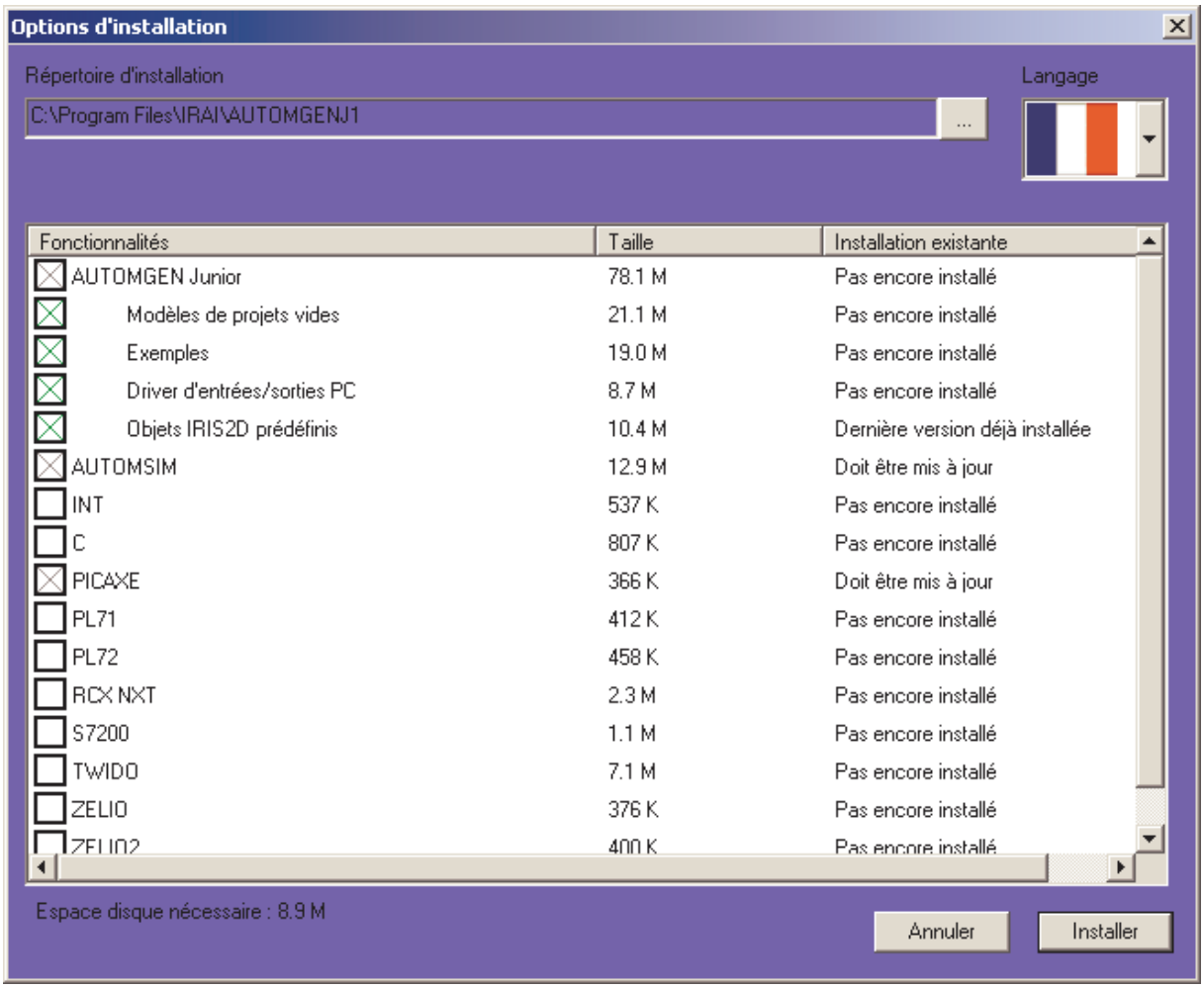
Introduction

AUTOMGEN Junior est la version d'AUTOMGEN destinée au collège. Elle est construite sur la base de l'atelier logiciel AUTOMGEN[8]. Cette version est l'évolution d'AUTOMGEN STARTER KIT vers les technologies d'automatisme disponibles dans AUTOMGEN[8], elle permet de ce fait d'accéder aux technologies de génération de code pour différentes cibles : automates programmables, cartes à micro-processeurs, robots, etc.

La première partie de ce manuel est spécifique à AUTOMGEN Junior, vous trouverez ensuite le manuel complet d'AUTOMGEN[8], pour les utilisateurs expérimentés souhaitant explorer toutes les fonctionnalités de ce produit.

Installation

Pendant l'installation d'AUTOMGEN Junior, choisissez les options que vous souhaitez installer :



Choisissez « Modèle de projet vides » si vous souhaitez ne pas installer les corrigés pour les scénarios (postes élèves). Dans le cas contraire, l'ouverture des scénarios ouvrira également un programme déjà réalisé permettant de faire fonctionner les systèmes proposés.

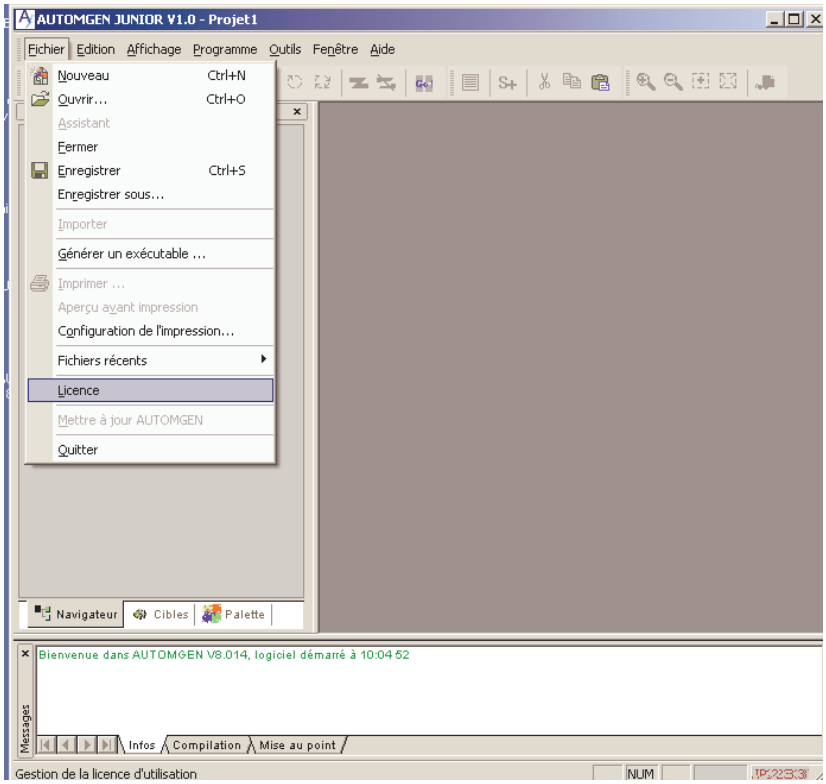
Cochez les cases « INT, C, PICAXE, PL71, PL72, RCX NXT, S7200, TWIDO, ZELIO, ZELIO2 » en fonction des systèmes cibles que vous souhaitez programmer. Vous pouvez décider de tout installer. Vous pourrez également relancer à tout moment l'installation d'AUTOMGEN Junior pour rajouter ou enlever des éléments. Notez que ces opérations n'affecteront pas une licence éventuellement installée (voir le chapitre « Enregistrer la licence » ci-après).

Enregistrer la licence

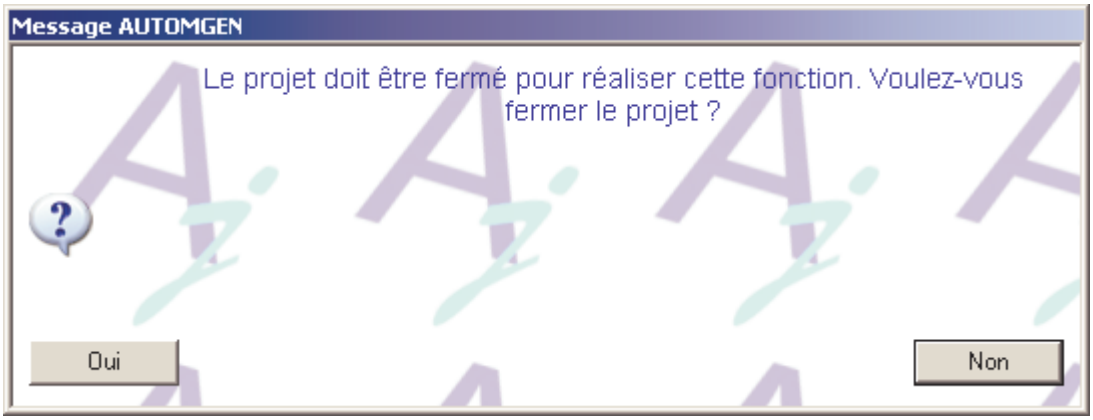
AUTOMGEN Junior fonctionne en version d'évaluation (version limitée à 30 jours d'essai) tant que vous n'avez pas enregistré la licence.

Pour enregistrer la licence, lancez AUTOMGEN Junior, la fenêtre suivante s'affiche :

Fermer cette fenêtre en cliquant sur la croix de fermeture en haut à droite…

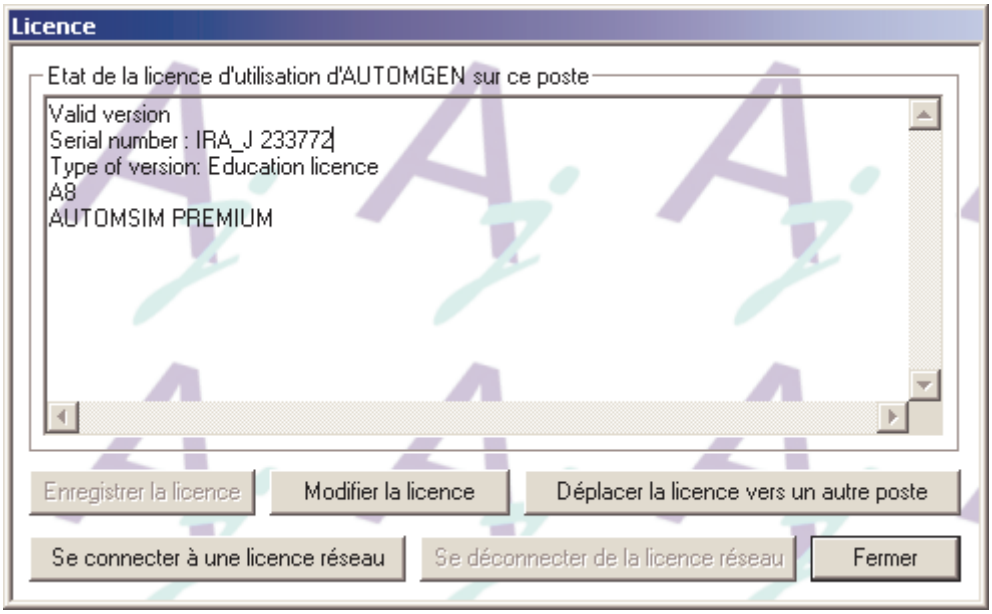


Allez ensuite dans le menu Fichier / Licence…

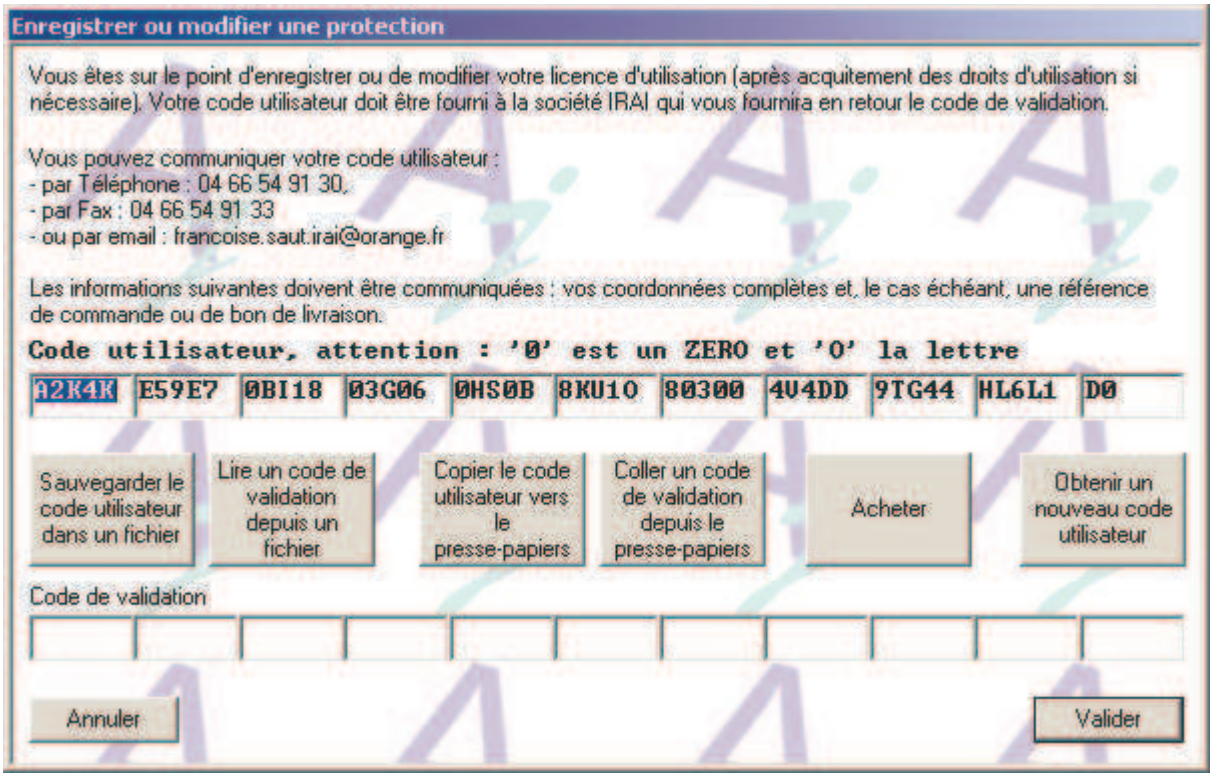


Répondez « Oui »…

Fermez de nouveau cette fenêtre…



Cliquez sur « Enregistrer la licence »…



Envoyez le code utilisateur ainsi généré par email à l'adresse francoise.saut.irai@orange.fr

Vous recevrez par email un code de validation que vous saisirez dans les zones « code de validation », puis vous cliquerez sur « Valider » pour valider la licence. Vous disposez de 20 jours entre la génération d'un code utilisateur et la saisie du code de validation.

Installation en réseau

Les fichiers d'AUTOMGEN Junior peuvent être installés sur un serveur de fichiers. Les licences peuvent également être gérées par un gestionnaire de licences réseau. Pour ceci, référez vous au chapitre correspondant dans le manuel d'AUTOMGEN[8] ci-après.

Utilisation

Au lancement d'AUTOMGEN Junior, vous pouvez choisir un des scénarios prédéfinis. Vous pouvez également fermer l'assistant de démarrage pour accéder à l'ensemble des fonctionnalités proposées par AUTOMGEN[8].

*

AUTOMGEN^{8}

www.irai.com

Environnement

Installation

Si vous installez à partir du CD-ROM d'AUTOMGEN, placez celui-ci dans votre lecteur de CD-ROM.

L'installation se lance automatiquement.

Si ce n'est pas le cas, lancez l'exécutable « Setup.exe » qui se trouve à la racine du CD-ROM.

Configuration requise

Ordinateur compatible PC équipé :
- du système d'exploitation WINDOWS 98 SE ou WINDOWS ME ou WINDOWS 2000 ou WINDOWS XP ou WINDOWS 2003 ou WINDOWS VISTA,
- 256 Mo de mémoire (suivant les systèmes d'exploitation, la mémoire requise par le système d'exploitation lui-même peut être supérieure),
- carte graphique avec au minimum une résolution de 1024 x 768 en 65536 couleurs.

Installation en réseau

AUTOMGEN peut être installé en réseau.

Exécutez la procédure d'installation sur le PC « serveur » (assurez vous d'avoir l'ensemble des droits d'accès au moment de l'installation).

Pour lancer AUTOMGEN, sur les PCs clients, créez un raccourci vers l'exécutable « autom8.exe » du répertoire d'installation d'AUTOMGEN sur le PC serveur.

Reportez-vous au chapitre « complément d'information sur l'installation d'AUTOMGEN en réseau » pour plus d'information sur l'installation d'AUTOMGEN et des licences en réseau.

Nouvelles fonctionnalités d'AUTOMGEN[8]

Choix du mode « Débutant / Expert »

Le mode « débutant » permet aux débutants d'utiliser un environnement « dépouillé » extrêmement simple à utiliser.

Intégration renforcée de la norme Grafcet 60848

Les nouveaux éléments de cette norme sont désormais accessibles dans les menus contextuels d'édition de programme.

Unicité des fichiers

Les fichiers générés par l'ensemble des mises à jour d'AUTOMGEN[8] peuvent être relus par l'ensemble des mises à jours d'AUTOMGEN[8].

Moteur physique intégré à IRIS3D

Le moteur TOKAMAK est intégré à IRIS3D. Il permet d'obtenir une simulation de parties opératives 3D d'un extrême réalisme.

Gestion d'objets 3D évolués dans IRIS3D

La sauvegarde et la relecture d'objets et de comportements permettent la gestion de bibliothèques d'objets réutilisables facilement. Des objets prédéfinis (vérins, tapis, etc…) sont proposés en standard. Une application de simulation de parties opératives 3D peut désormais être réalisée en quelques clics de souris.

Liens évolués entre les objets IRIS3D et AUTOMGEN

Des modes évolués permettent de gérer facilement les déplacements d'objets complexes entre AUTOMGEN et IRIS3D. Une variable AUTOMGEN peut par exemple donner directement la vitesse d'un objet. Le compte rendu de la position peut également être simulé à la façon d'un codeur absolu.

Objets IRIS3D texturés

Les objets texturés apportent désormais à IRIS3D un fantastique réalisme de rendu.

Drag and drop depuis IRIS3D vers les folios AUTOMGEN

Un clic droit sur les objets IRIS3D permet d'accéder à la liste des variables et de « traîner » une référence jusqu'à un folio de programmation.

Objet AUTOMSIM définissable par l'utilisateur

Les utilisateurs d'AUTOMSIM apprécieront le nouvel objet définissable par l'utilisateur qui permet de créer vos propres objets.
(voir la partie de ce manuel consacrée à AUTOMSIM)

Drag and drop depuis AUTOMSIM vers les folios AUTOMGEN

Un clic sur les objets AUTOMSIM permet de « traîner » une référence jusqu'à un folio de programmation.

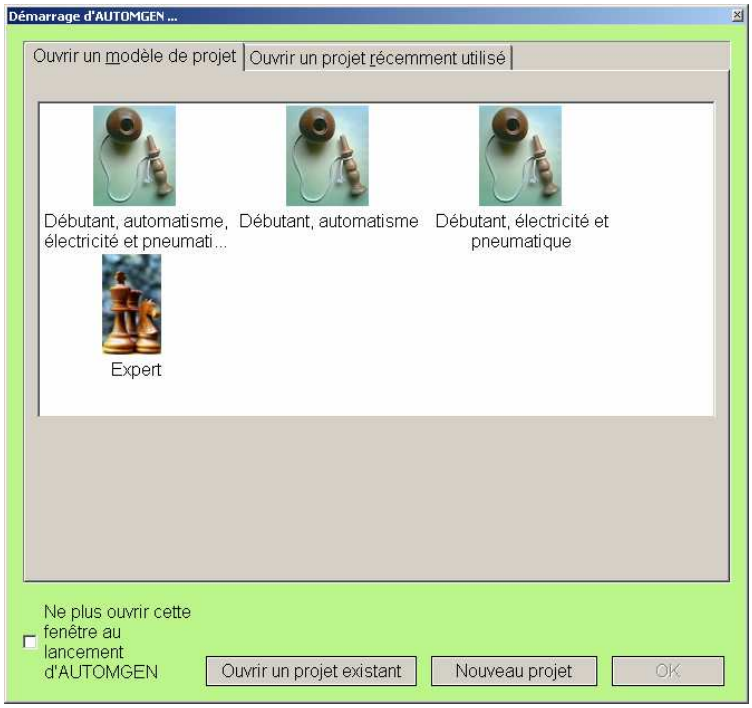
Améliorations de l'environnement

Enfin, de nombreuses améliorations de l'environnement, telles que la loupe sur la palette de dessin, les palettes simplifiées du mode "débutants" ou la personnalisation des menus rendent AUTOMGEN encore plus convivial.

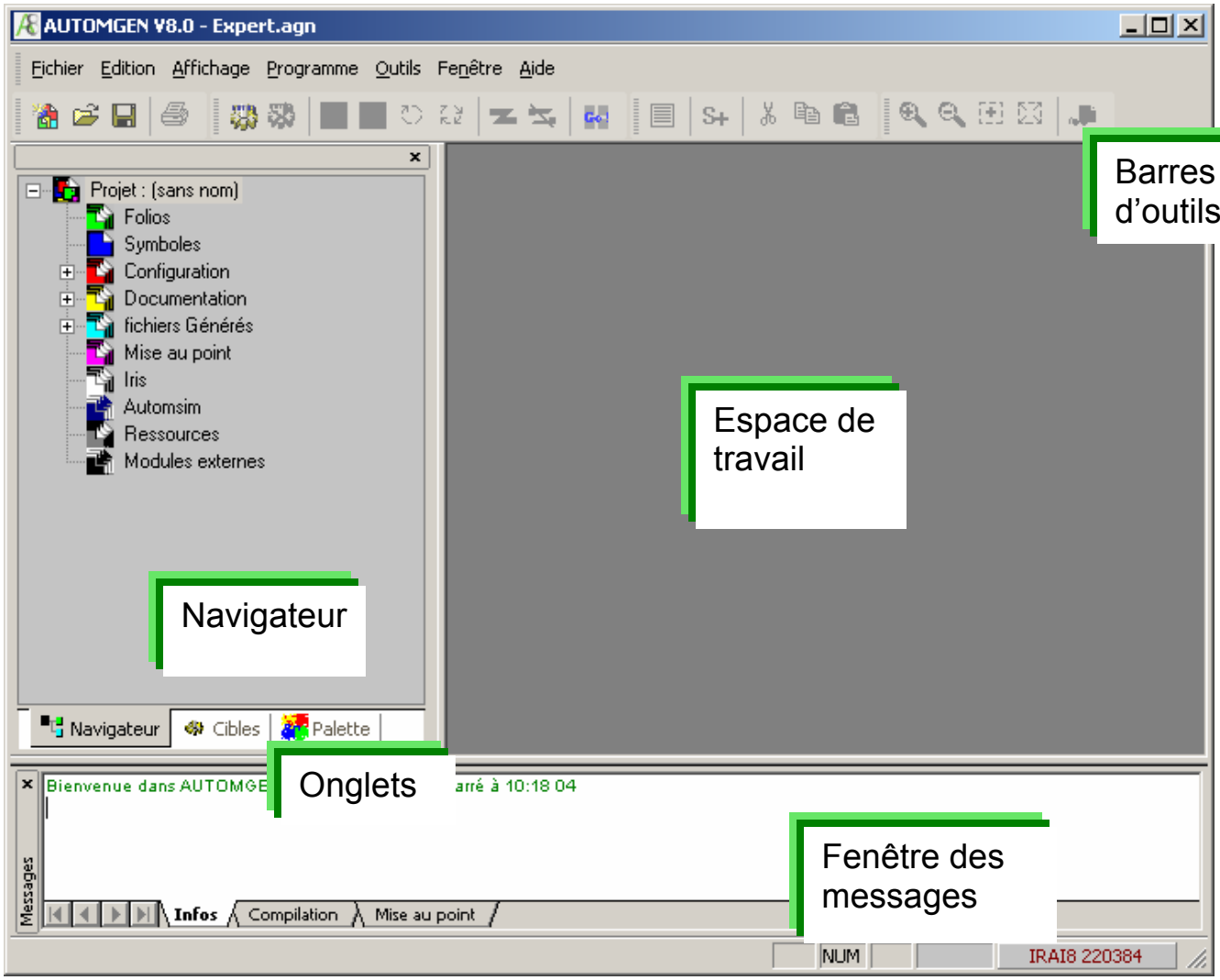
L'environnement

Démarrage

Au lancement d'AUTOMGEN, le choix d'un mode pour l'environnement est proposé. Les modes « Débutants » permettent de commencer à utiliser AUTOMGEN dans une configuration « dépouillée » avec peu d'options dans les menus et des palettes simplifiées. Ce mode convient particulièrement aux personnes utilisant AUTOMGEN pour la première fois. Le mode expert propose l'ensemble des fonctionnalités. Vous pourrez créer vos propres modes (voir « modèles de projets »).



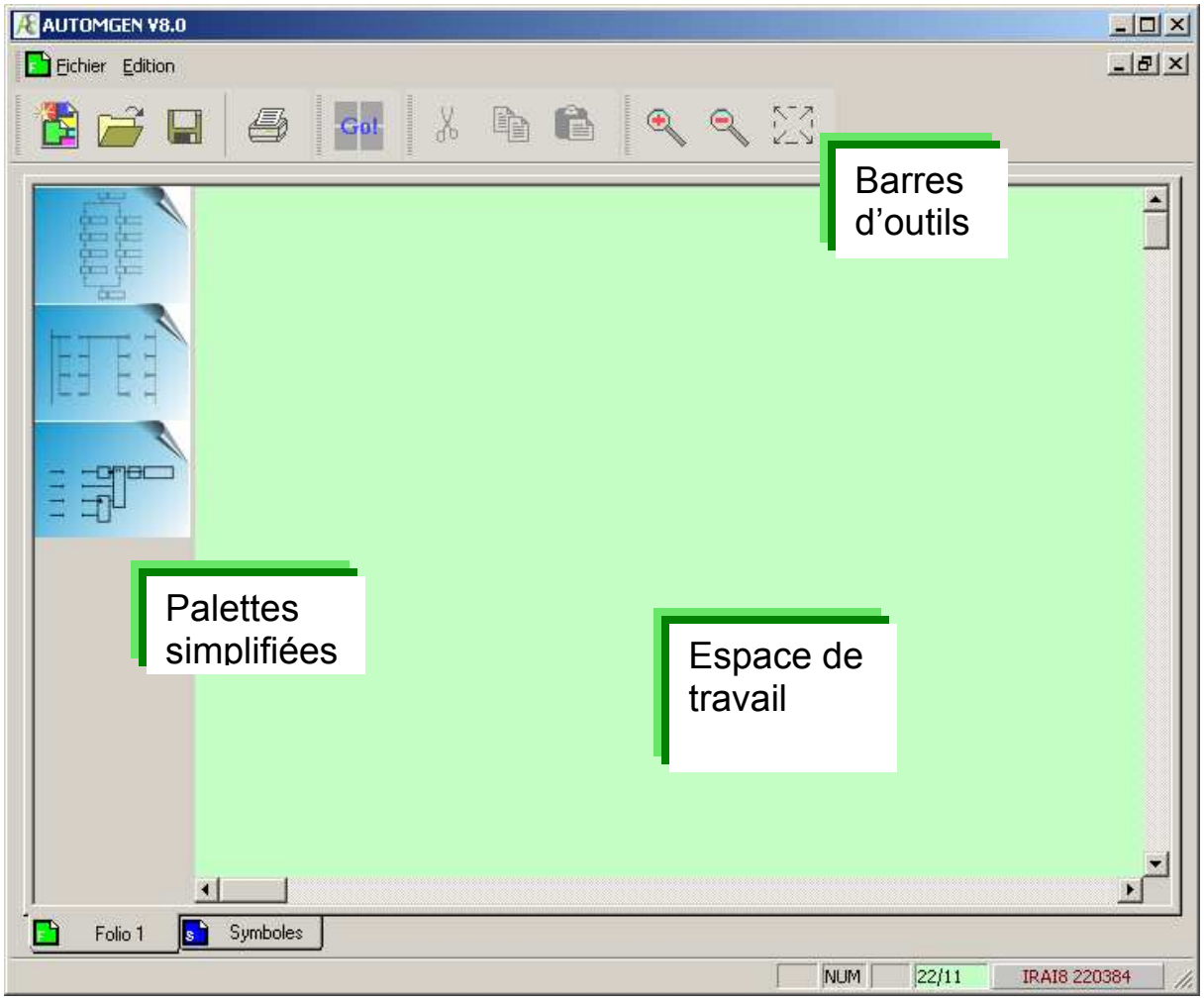
Vues générales



La fenêtre principale d'AUTOMGEN en mode « Expert »

L'environnement est entièrement paramétrable. Les barres d'outils peuvent être déplacées (en les saisissant par) et paramétrées (menu « Outils/Personnaliser l'environnement »).

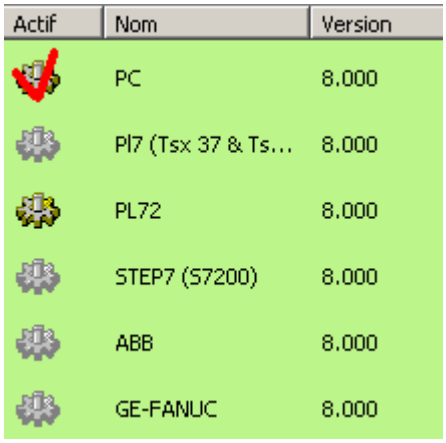
L'état de l'environnement est sauvegardé lorsqu'on le quitte. Cet état peut aussi être sauvegardé dans un fichier projet (voir les options du projet).



La fenêtre principale d'AUTOMGEN en mode « Débutant »

Choix des cibles en mode expert

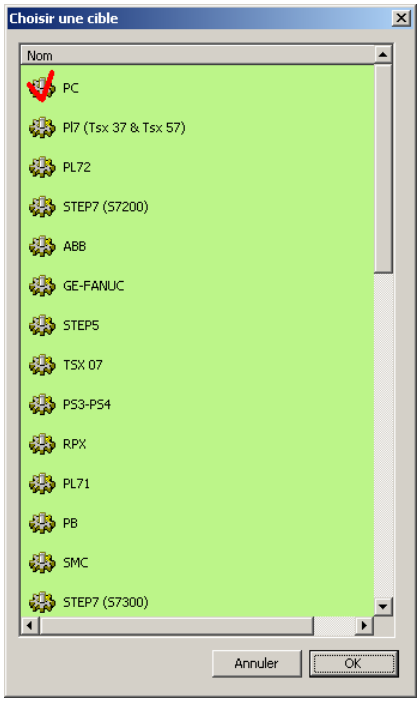
En bas de la fenêtre du navigateur se trouve un onglet « Cibles » permettant d'accéder à la liste des post-processeurs installés.



La cible active est marquée d'une coche rouge. L'accès aux cibles apparaissant en grisé n'est pas autorisé par rapport à la licence installée (voir le chapitre « Licences » pour plus de détails). Pour modifier la cible courante, double cliquez sur la ligne correspondante. Les cibles apparaissant dans cette liste sont celles sélectionnées à l'installation. Si la cible que vous souhaitez utiliser n'apparaît pas dans cette liste, relancez l'installation d'AUTOMGEN et installez la.

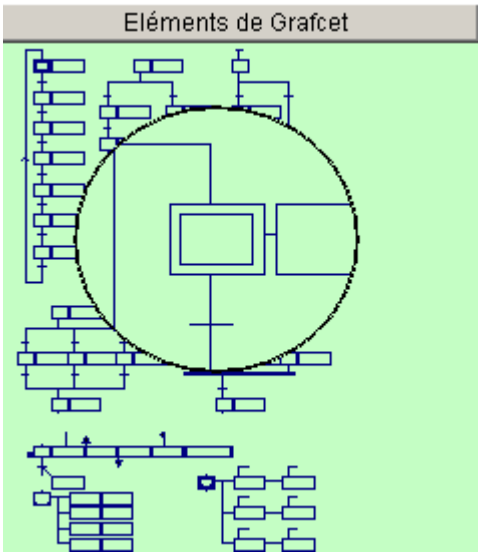
Choix des cibles en mode débutant

En mode débutant, la fenêtre ci-dessous s'ouvre pour permettre le choix de la cible à chaque exécution de programme.



Palettes en mode expert

En bas de la fenêtre du navigateur se trouve un onglet « Palette » permettant d'accéder à des éléments de dessin de programmes.



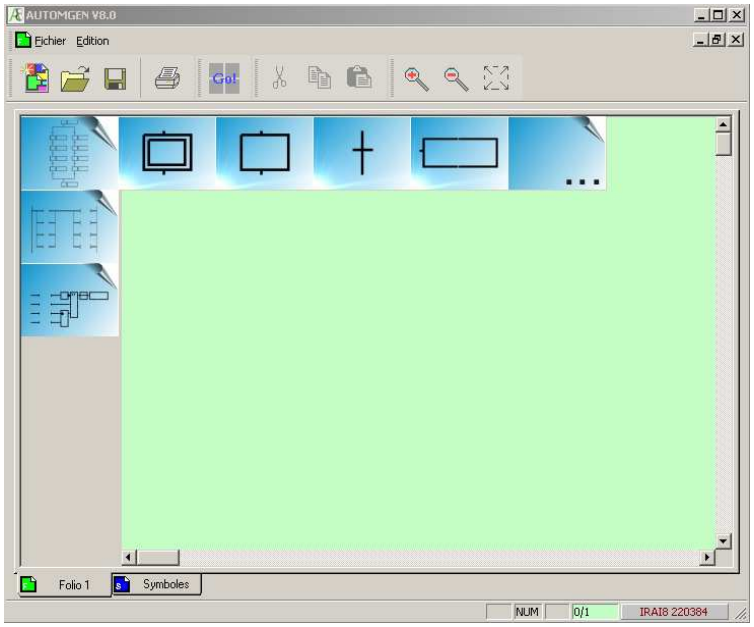
La palette donne un ensemble d'éléments qui peuvent être sélectionnés et déposés sur les folios. Pour sélectionner un élément, cliquez avec le bouton gauche de la souris dans la palette, étirez la sélection, relâchez le bouton de la souris, cliquez dans la zone sélectionnée et déplacez la zone vers le folio.

La palette contient également la liste des symboles du projet. Vous pouvez les saisir et les faire glisser sur un test ou une action sur un folio.

Une loupe s'affiche automatiquement sur la palette lorsque les éléments affichés sont petits.

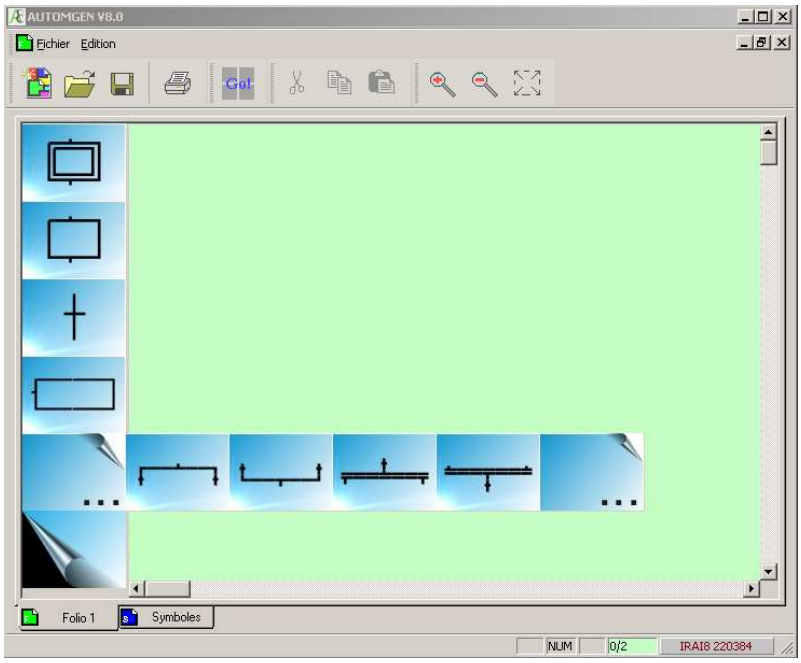
Palettes en mode débutant

La palette du mode débutant regroupe de façon réduite les éléments les plus simples pour dessiner les programmes. Cliquer avec le bouton gauche de la souris sur une catégorie « ouvre cette catégorie » à la manière d'une arborescence.



La palette en mode « débutant »

Les éléments «  » permettent d'accéder à des sous éléments.



La palette en mode « débutant » montrant la catégorie « Grafcet » ouverte

L'élément «  » permet de refermer une catégorie.

Montrer ou cacher la fenêtre du projet ou les fenêtres de messages

Sélectionnez l'option « Projet » ou « Messages » dans le menu « Fenêtres ».

Afficher l'espace de travail en mode plein écran

Sélectionnez l'option « Plein écran » dans le menu « Affichage ». Cliquez sur  pour sortir du mode plein écran.

Raccourcis claviers

Les raccourcis claviers sont inscrits dans les menus. Des raccourcis « masqués » sont également utilisables :

CTRL + ALT + F7	Définir les menus accessibles

CTRL + ALT + F8	Sauvegarder le projet sous la forme d'un exécutable
CTRL + ALT + F9	Sauvegarder le projet
CTRL + ALT + F10	Accéder aux propriétés du projet
CTRL + ALT + F11	Montrer ou cacher la fenêtre AUTOMGEN

Résumé Environnement

L'environnement est entièrement paramétrable, son état est sauvegardé lorsqu'on quitte AUTOMGEN. On peut cacher les fenêtres de l'environnement. Le menu « Fenêtres » permet de les afficher de nouveau. L'espace de travail peut être affiché en mode plein écran. Les onglets en bas de la fenêtre du navigateur permettent d'accéder au choix du post-processeur courant ainsi qu'à la palette de dessin.

Licences

Une licence définit les droits d'utilisation d'AUTOMGEN. Les éléments suivants sont déterminés par une licence :

- le nombre d'entrées / sorties tout ou rien utilisables,
- les post-processeurs utilisables,
- le nombre d'utilisateurs (licence réseau uniquement).

Enregistrer une licence

Lorsque vous installez AUTOMGEN, vous pouvez l'utiliser gratuitement pendant une durée de 40 jours.

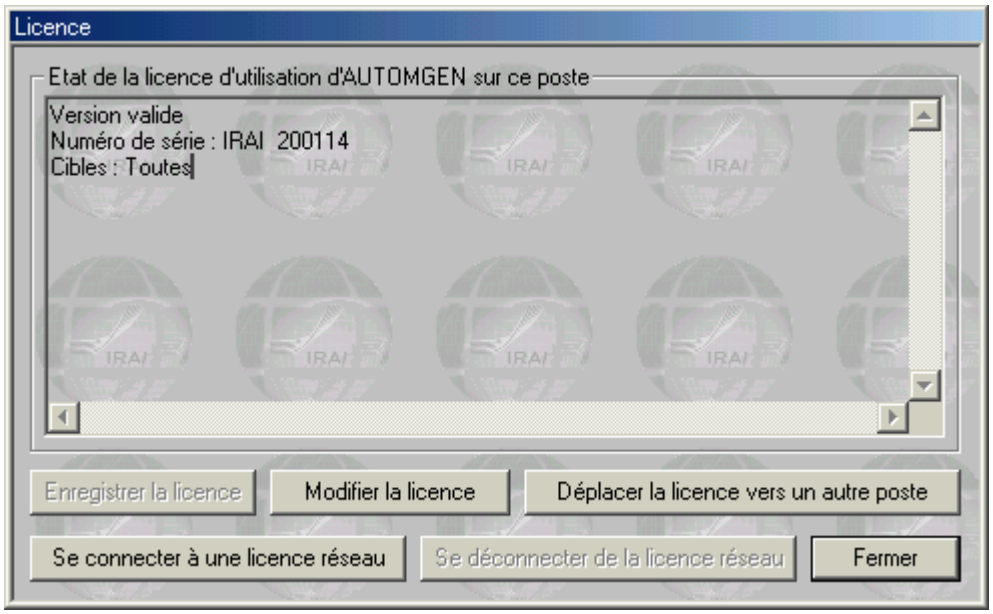
Pendant ces 40 jours, vous devez enregistrer votre licence.

Pour enregistrer votre licence, vous devez communiquer à IRAI :

- le numéro de série imprimé sur une étiquette collée sur la boîte du logiciel ou, à défaut, la référence de votre bon de livraison ou de votre commande,
- un code utilisateur fourni par le logiciel identifiant le PC sur lequel vous avez installé le produit.

Vous recevrez en retour un code de débridage (on parle aussi de code de validation).

L'option « Licence » du menu « Fichier » d'AUTOMGEN vous permet de visualiser le statut de votre licence et d'obtenir un code utilisateur (en cliquant sur « Enregistrer la licence »).



Etat de la licence.

Un code utilisateur est valide pendant une durée de 10 jours.

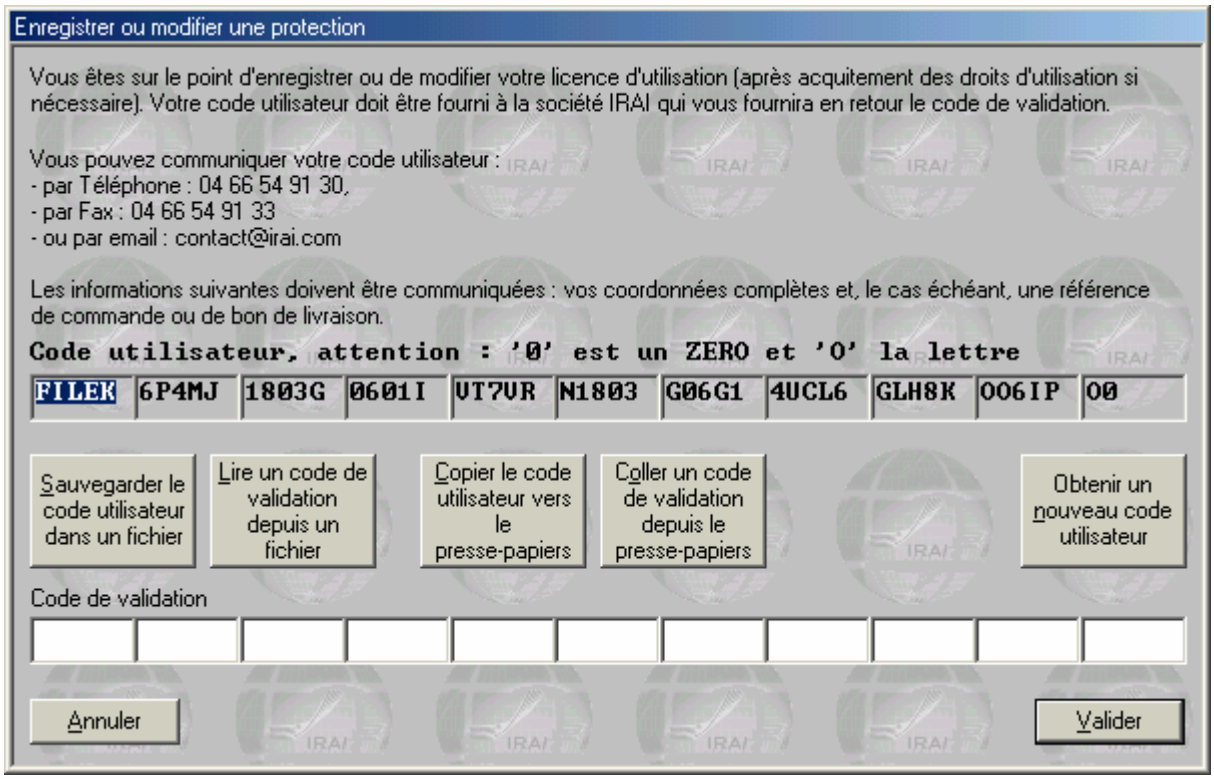
Il peut donc s'écouler un maximum de 10 jours entre le moment ou vous communiquez un code utilisateur à IRAI et le moment ou vous saisissez un code de débridage fourni par IRAI.

Envoyer un code utilisateur à IRAI

Plusieurs méthodes sont à votre disposition. La méthode d'échange des codes par email est fortement recommandée pour limiter les risques d'erreur.

Une seule erreur sur le code entraînera un échec de l'enregistrement d'une licence.

Envoyer un fichier par email (la meilleur solution)



La boîte de dialogue d'enregistrement d'une licence.

Pour générer un fichier contenant votre code utilisateur, cliquez sur « Sauvegarder le code utilisateur dans un fichier ». Vous pouvez ensuite transmettre ce fichier portant l'extension « .a8u » comme pièce jointe d'un email à envoyer à l'adresse contact@irai.com.

Copier le code utilisateur dans le message d'un email

En cliquant sur « Copier le code utilisateur vers le presse papier », vous pourrez ensuite coller le code dans le corps du message d'un email pour le transmettre à l'adresse email contact@irai.com.

Par fax (déconseillé)

En cliquant sur « Copier le code utilisateur vers le presse papier », vous pourrez ensuite coller le code dans un document et l'envoyer par fax au +33 4 66 54 91 33. Evitez si possible l'écriture manuscrite, prenez soin d'imprimer avec une fonte permettant de différencier la lettre « O » du chiffre zéro.

Par téléphone (fortement déconseillé)

En téléphonant au +33 4 66 54 91 30. Prenez soin de différencier la lettre « O » du chiffre zéro. Attention aux consonnes difficiles à différencier au téléphone (« S » et « F » par exemple).

Entrer le code de validation / débridage

Débridage par un fichier reçu par email

Si vous avez reçu un fichier « .a8v » par email, enregistrez le fichier reçu sur votre disque dur, cliquez sur « Lire un code de validation depuis un fichier », et sélectionnez le fichier.

Débridage par un code reçu dans le texte dans un email

Sélectionnez le code dans le texte du message (prenez soin de ne sélectionner que le code et de ne pas ajouter d'espaces à la fin). Cliquez sur « Coller un code de validation depuis le presse papier ».

Débridage par un code reçu par fax ou au téléphone

Saisissez le code dans les cases se trouvant sous l'intitulé « Code de validation ».

Modifier une licence

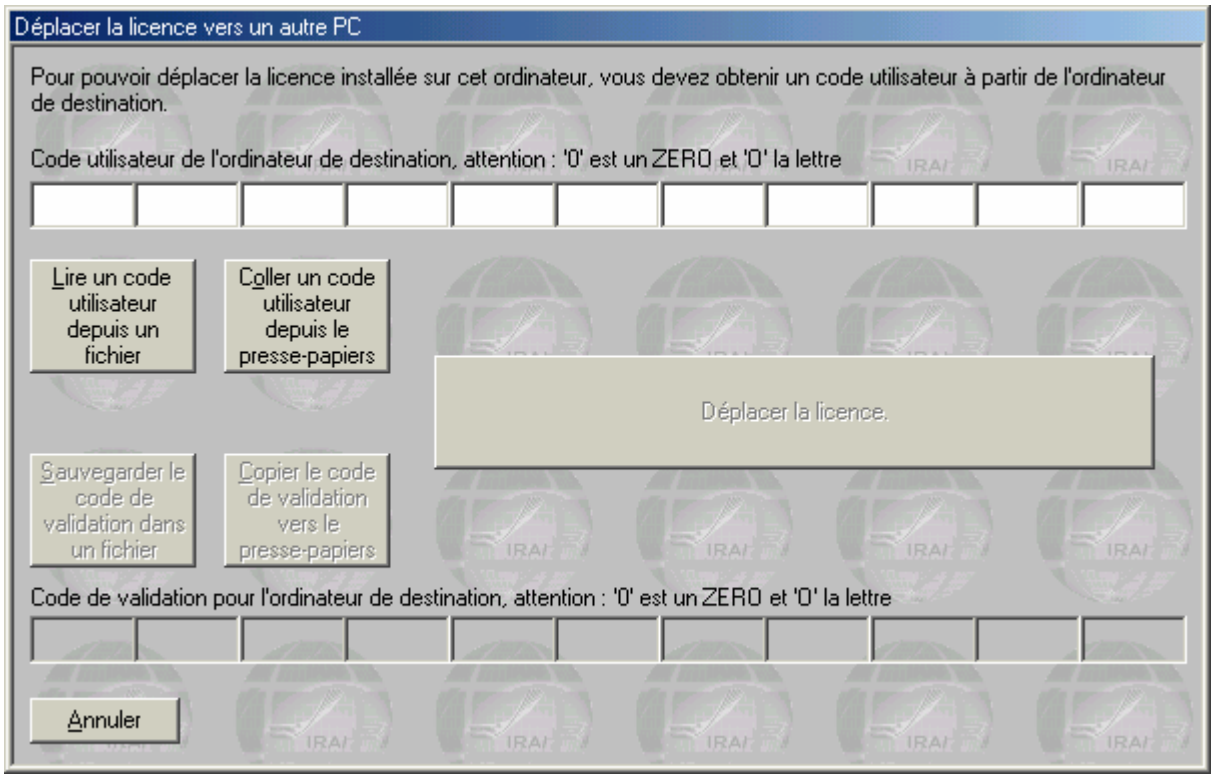
La modification d'une licence consiste à faire évoluer les éléments autorisés par la licence (ajout d'un post-processeur par exemple).
La procédure de modification d'une licence est en tout point identique à l'enregistrement.

Déplacer une licence d'un ordinateur à un autre

Cette procédure est la plus complexe. Les instructions qui suivent doivent être scrupuleusement respectées pour obtenir un bon résultat. Dans les explications qui suivent, le PC « source » désigne l'ordinateur où se trouve la licence et le PC « de destination » l'ordinateur où doit être déplacée la licence.

1- si ce n'est pas déjà fait, installez AUTOMGEN sur le PC de destination,
2- générez un fichier de code utilisateur « .a8u » sur le PC de destination et déplacer ce fichier vers le PC source (en le copiant sur une disquette par exemple),

3- sur le PC source, choisissez l'option « Déplacer la licence vers un autre poste »,

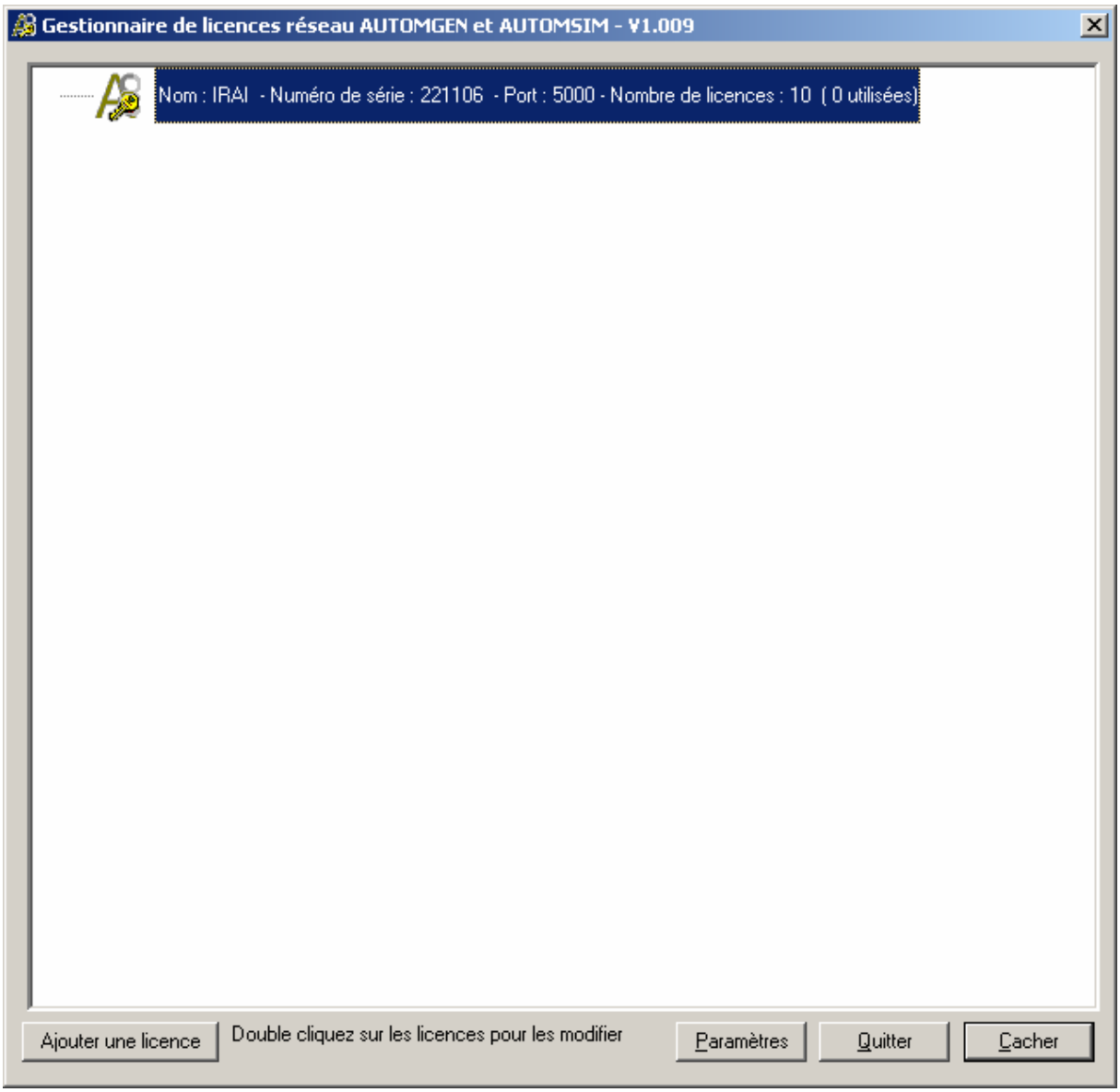


La boîte de dialogue de déplacement d'une licence

4- sur le PC source, cliquez sur « Lire un code utilisateur depuis un fichier » et sélectionnez le fichier « .a8u » provenant du PC de destination,
5- sur le PC source cliquez sur « Déplacer la licence »,
6- sur le PC source cliquez sur « Sauvegarder le code de validation dans un fichier », recopier le fichier « .a8v » généré vers le PC de destination,
7- sur le PC de destination cliquez sur « Lire un code de validation depuis un fichier » et sélectionnez le fichier « .a8v » en provenance du PC source.

Licences réseau

L'exécutable « akey8.exe » est le gestionnaire de licence réseau. Cet exécutable doit être lancé sur un des ordinateurs du réseau. Le réseau doit permettre l'utilisation du protocole TCP IP. Au lancement, le gestionnaire de licences réseau est masqué et seul une icône apparaît dans la barre des tâches de WINDOWS. Pour visualiser la fenêtre du gestionnaire de licence réseau, double cliquez sur l'icône dans la barre des tâches.



Le gestionnaire de licences réseau

Jusqu'à 16 licences différentes peuvent être gérées par le gestionnaire de licences réseau. Une licence réseau est caractérisée par un nombre d'utilisateurs et un type de droit (nombre d'entrées / sorties tout ou rien et post-processeurs utilisables). Pour chaque licence est affiché le nombre d'utilisateur(s) possible(s), le nombre d'utilisateur(s) connecté(s) et la liste des utilisateurs connectés (en train d'utiliser AUTOMGEN) sous forme d'arborescence attachée à chaque licence. A chaque licence est associée un numéro de port (une valeur numérique à partir de 5000 par défaut). Le premier numéro de port utilisé peut être configuré en cliquant sur « Paramètres ».

Ajouter une licence réseau

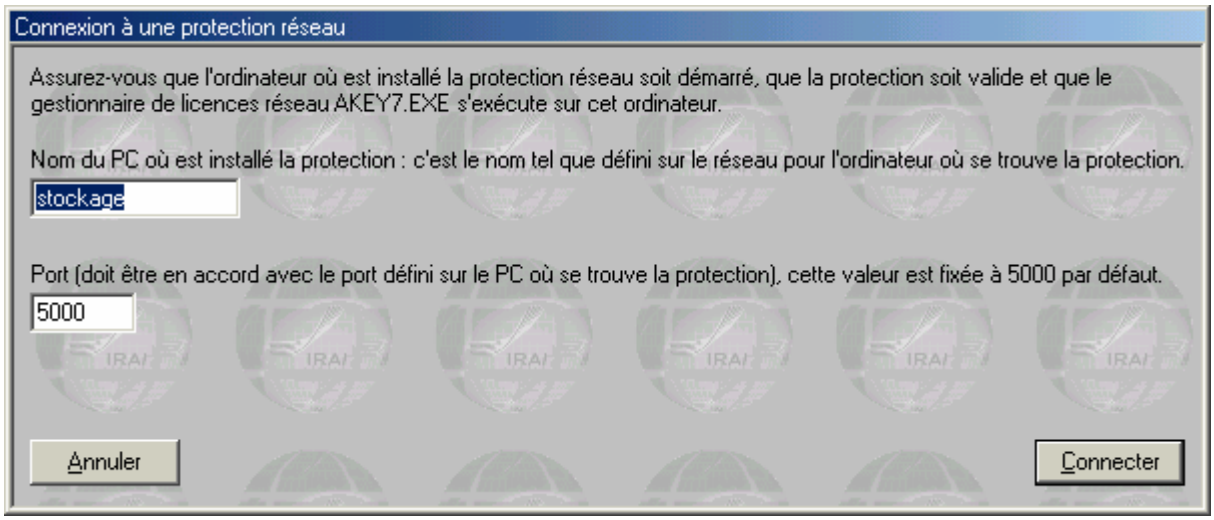
En cliquant sur « Ajouter une licence » vous pouvez ajouter une licence. Le principe d'enregistrement des licences est le même que pour les versions monopostes.

Modifier une licence

Double cliquez sur les licences pour les modifier. La procédure de modification des licences est identique à celle utilisée pour les licences monopostes.

Connexion des postes clients

Cliquez sur « Se connecter à une licence réseau » pour connecter un poste client à une licence réseau.



Connexion à une licence réseau

Le nom du PC (tel que vue sur le réseau) où a été lancé « akey8.exe » doit être fourni ainsi que le numéro du port correspondant à la licence souhaitée.

Résumé Licences

Vous devez enregistrer votre licence auprès de IRAI (contact@irai.com) en envoyant par email votre code utilisateur (menu « Fichier/Licence »). Le gestionnaire de licences réseau permet de gérer plusieurs licences sur un des PCs du réseau TCP IP.

Complément d'information sur l'installation d'AUTOMGEN en réseau

Généralités

Il faut distinguer deux aspects de l'installation d'AUTOMGEN[8] : l'installation des fichiers d'une part et la gestion des licences d'autre part. Ces deux éléments sont totalement séparés : on peut choisir d'installer les fichiers sur le disque dur des PCs clients ou sur un serveur de fichiers et choisir indépendamment d'installer une licence en local sur un PC ou une licence réseau sur un gestionnaire de licences réseau.

Installation d'AUTOMGEN[8] sur un serveur de fichiers

Intérêt : installer une seule fois les fichiers d'AUTOMGEN[8] sur un serveur de fichiers, les mises à jour s'en trouvent simplifiées.

Procédure sur le serveur de fichier : installer AUTOMGEN[8]. Droits nécessaires : un accès en lecture est suffisant.

Procédure sur les postes clients : créer un raccourci vers l'exécutable « autom8.exe » qui se trouve dans le répertoire d'installation d'AUTOMGEN[8] sur le serveur de fichiers.

Installation d'une ou plusieurs licences AUTOMGEN[8] sur un gestionnaire de licences réseau

Intérêt : les licences ne sont plus immobilisées sur un PC mais peuvent être utilisées par l'ensemble des PCs connectés au réseau (licences flottantes).

Principe : une ou plusieurs licences sont installées sur un des PCs du réseau. Une licence autorise de 1 à n utilisateurs. AUTOMGEN[8] pourra être lancé sur les PCs clients à concurrence du nombre d'utilisateur(s) maximum. Une licence possède les mêmes caractéristiques pour tous les utilisateurs en termes de nombre d'entrées/sorties utilisables et de

types de post-processeur utilisables. Si plusieurs configurations (plusieurs types de licences) sont nécessaires, autant de licences que de types de configurations différentes seront créées. Au lancement d'AUTOMGEN[8] sur les PCs clients, une connexion sera réalisée avec l'une ou l'autre des licences en fonction des caractéristiques souhaitées.

Exemple concret : mise en réseau de 4 licences 16 E+16 S PL72, 4 licences 16 E+16 S PL7 + 2 licences PL7 E/S illimitées. Pour ceci : 3 licences seront créées sur le gestionnaires de licences réseau : 1 licence 4 utilisateurs 16 E+16 S PL72, 1 licence 4 utilisateurs 16 E+16 S PL7, 1 licence 2 utilisateurs E/S illimitées PL7.

Où installer le gestionnaire de licences réseau : sur un PC du réseau (pas forcément le serveur) qui devra fonctionner en permanence (dés qu'un utilisateur voudra utiliser AUTOMGEN[8]).

Contraintes techniques : le réseau doit supporter TCP-IP, le PC où se trouve le gestionnaire de licences réseau devra être capable d'exécuter un programme WINDOWS (application ou service).

Installation sur le gestionnaire de licences réseau : installez sur le PC où seront gérées les licences réseau le module principal d'AUTOMGEN[8] + le gestionnaire de licences réseau.

Enregistrement d'une ou plusieurs licences sur le gestionnaire de licences réseau : lancez le gestionnaire de licences réseau : (exécutable AKEY8.EXE se trouvant dans le répertoire d'installation d'AUTOMGEN[8]). Au lancement, le gestionnaire de licences se met en icône en bas à droite de la barre des tâches de WINDOWS. Cliquez une fois avec le bouton gauche de la souris pour afficher la fenêtre.

Cliquez sur « Ajouter une licence » pour ajouter une licence.

Cliquez sur « Sauvegarder le code utilisateur dans un fichier » pour générer un fichier .n8u que vous nous transmettrez par email à l'adresse « contact@irai.com » : nous vous retournerons un fichier .n8v que vous relierez en cliquant sur le bouton « Lire un code de validation depuis un fichier ».

Les licences ainsi installées apparaissent ensuite dans le gestionnaire de licences réseau avec le numéro de série, les caractéristiques de la licence et le numéro de port associé. C'est ce numéro de port qui permettra aux clients de se connecter sur telle ou telle licence.

Installation sur les clients : lancez AUTOMGEN[8], allez dans le menu « Fichier / Licence » et sélectionnez « Se connecter à une licence réseau ».

Entrez le nom du PC ou s'exécute le gestionnaire de licences réseau (ou son adresse IP) ainsi que le numéro du port (ce numéro permet de désigner la licence à laquelle on souhaite se connecter s'il y a plusieurs licences).

Il est également possible d'ajouter un argument dans le raccourci de lancement d'AUTOMGEN[8] afin de forcer la connexion à une licence réseau.

L'argument est :

/NETLICENSE=<nom du PC où se trouve le gestionnaire de licences réseau>,<port>

Attention à l'orthographe de « NETLICENSE » : S et non C à la fin.

Par exemple :

/NETLICENSE=MONSERVEUR,5001

Il est possible de créer plusieurs raccourcis de lancement pour se connecter à différentes licences.

Problèmes possibles : si vous utilisez un firewall, veillez à autoriser l'accès aux ports utilisés par le gestionnaire de licences réseau (ceux affichés dans le gestionnaire de licences réseau).

Installation du gestionnaire de licence réseau en tant que service sous WINDOWS NT, 2000, XP, 2003 et VISTA : voir le chapitre suivant.

Affichage de l'état des licences à distance : pour afficher l'état du gestionnaire de licences réseau sur un autre PC que celui où est lancé le gestionnaire de licences réseau (ou si c'est la version « service » du gestionnaire de licences réseau qui est utilisée), utilisez l'utilitaire « spya8protnet.exe » qui se trouve dans le répertoire d'installation d'AUTOMGEN[8].

Installation du serveur de licences réseau sous la forme d'un service

Le serveur de clé « Service NT » permet de gérer les licences réseau AUTOMGEN[8] sur un poste WINDOWS NT4, 2000, 2003, XP ou VISTA sans ouvrir de session. Contrairement à la version « exécutable » AKEY8.EXE, AKEY8NT.EXE ne permet de visualiser ni les protections ni les utilisateurs connectés.

Avant d'installer le serveur de clé en tant que « service NT », il est fortement recommandé de s'assurer du bon fonctionnement du serveur de clé avec la version « exécutable » : AKEY8.EXE.

Lancez la ligne de commande « akey8nt –i » pour installer le service serveur de clé NT. L'exécutable AKEY8NT.EXE est installé dans le répertoire d'installation d'AUTOMGEN.

Pour que le service démarre automatiquement :

- sous WINDOWS NT4 : allez dans le menu « Démarrer/Paramètres/Panneau de configuration », choissez l'icône « Services » la ligne « AKEY8 », cliquez sur le bouton démarrage et choisissez le bouton « Automatique ».

  Redémarrez votre PC pour que le serveur de clé soit actif.

- sous WINDOWS 2000, 2003, XP ou VISTA : allez dans le menu « Démarrer/Paramètres/Panneau de configuration », choisissez l'icône « Outils d'administration » puis l'icône « Services » . Cliquez avec le bouton droit de la souris sur la ligne « AKEY8 » et choisissez « propriétés ». Dans l'option « Type de démarrage », choisissez « Automatique ». Dans l'onglet « Récupération », choisissez « Redémarrer le service » dans la zone « Première défaillance ».

Lancez la commande « akey8nt –u » pour installer le service serveur de clé NT.

Après avoir désinstallé le service AKEY8NT.EXE, utilisez AKEY8.EXE pour déterminer la cause d'éventuels dysfonctionnements.

Le projet

La notion de projet est très forte dans AUTOMGEN. Un projet regroupe l'ensemble des éléments composant une application. Le navigateur (voir page 39) affiche sous forme arborescente tous les éléments d'un projet (folios, symboles, configuration, objets IRIS, etc …).

Le nouveau format de fichier d'AUTOMGEN (fichiers portant l'extension « .AGN ») encapsule tous les éléments d'un projet.

Lorsque vous sauvegardez un fichier « .AGN » vous avez l'assurance de sauvegarder la totalité des éléments d'une application. Vous pouvez échanger facilement et efficacement les applications créées avec AUTOMGEN.

Les fichiers « .AGN » sont compactés avec la technologie « ZIP », nul n'est donc besoin de les compresser pour les échanger, leur taille est déjà optimisée. Tous les fichiers générés par AUTOMGEN[8] peuvent être relus avec toutes les versions d'AUTOMGEN[8] : compatibilité ascendante et descendante.

Fichiers générés avec AUTOMGEN[7]

Les fichiers créés avec AUTOMGEN[7] peuvent être directement ouverts dans AUTOMGEN[8].

Importer une application d'une ancienne version d'AUTOMGEN (version 6 ou antérieure)

Vous devez importer l'ensemble des folios (fichiers « .GR7 ») et l'éventuel fichier des symboles (fichier « .SYM »). Pour cela utilisez les procédures d'importation décrites dans les chapitres suivants.

Importer un projet créé avec un autre atelier logiciel

La commande « Importer » du menu « Fichier » permet d'importer des fichiers « .FEF » provenant des ateliers logiciels SCHNEIDER.

Générer un fichier exécutable distribuable gratuitement

La commande « Générer un exécutable » du menu « Fichier » permet de générer un exécutable à partir du projet en cours (un ficher « .EXE » exécutable sur PC sous WINDOWS). La « visionneuse » AUTOMGEN est intégrée automatiquement à l'exécutable généré (l'utilisateur de

l'exécutable n'a pas besoin d'AUTOMGEN). Cette visionneuse permet d'utiliser l'application sans la modifier. Vous pouvez ainsi déployer vos applications facilement. L'exécutable généré est libre de droit. Cette technique est typiquement utilisée pour produire une application de supervision.

Modifier les propriétés du projet

Cliquez avec le bouton droit de la souris sur l'élément « Projet » dans le navigateur et choisissez « Propriétés » dans le menu.

Modifier les options de sécurité

Vous pouvez restreindre l'accès en lecture ou en modification du projet par des mots de passe.

Options avancées

« Sauver l'aspect de l'environnement avec le projet » : si coché, alors la position des fenêtres ainsi que l'aspect des barres d'outils sont sauvegardés dans le fichier « .AGN ». A l'ouverture du projet, ces éléments seront restitués.

« Cacher la fenêtre principale au lancement … » : si coché, la fenêtre d'AUTOMGEN est cachée à l'ouverture du projet. Seuls les objets d'IRIS incorporés au projet sont visibles. Cette option est typiquement utilisée pour créer des applications « packagées » ne laissant apparaître que des objets IRIS. La combinaison de touche [CTRL] + [F11] permet de faire réapparaître la fenêtre AUTOMGEN.

Les autres options permettent d'agir sur l'affichage de la fenêtre AUTOMGEN à l'ouverture du projet.

Interface utilisateur

« Interdire la configuration des objets IRIS » : si coché, la configuration des objets IRIS ne peut être modifiée par l'utilisateur.

Les autres options permettent de modifier le comportement de l'interface utilisateur.

Modèle

« Ce projet est un modèle de document » : si coché, à son ouverture, toutes les options et documents qu'il contient servent de modèle à la création d'un nouveau projet. Cette fonctionnalité permet de créer des configurations standards pouvant être chargées au lancement

d'AUTOMGEN (un fichier de symbole par défaut ou une configuration automate par défaut par exemple).

Définir un mode

Pour définir un mode utilisable au lancement d'AUTOMGEN (à la façon des modes « Expert » et « Débutant »), enregistrez un modèle de projet dans le sous-répertoire « models » du répertoire d'installation d'AUTOMGEN. Une image peut être associée à un modèle, pour ceci, créez un fichier au format « jpg » portant le même nom que le fichier « .agn ». Ce fichier doit avoir les dimensions suivantes : 120 pixels de large par 90 pixels de haut.

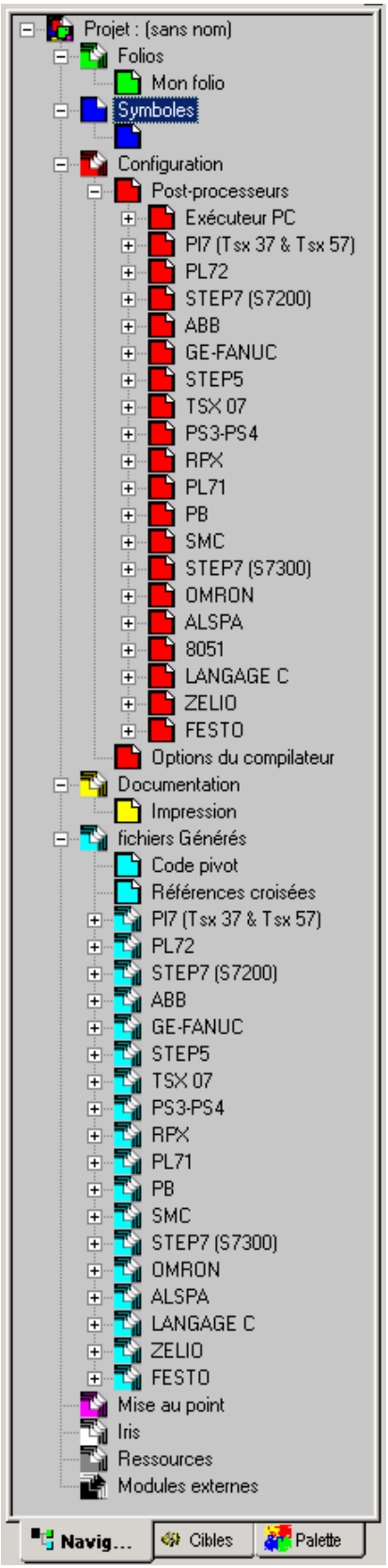
GO automatique

«Go automatique au lancement du projet » : si coché, l'exécution de l'application est automatique à l'ouverture du projet.

Résumé Projet

Le projet permet de regrouper l'ensemble des éléments d'une application d'AUTOMGEN. Ainsi regroupés, les éléments ne forment plus qu'un fichier compacté qui porte l'extension « .AGN ». Les modèles de projet permettent de gérer facilement différentes configurations du logiciel. La génération d'exécutables rend aisé le déploiement des applications.

Le navigateur



Elément central de la gestion des applications, le navigateur permet un accès rapide aux différents éléments d'une application : folios, symboles, configuration, impressions, objets IRIS, etc …

Les icônes « + » et « - » permettent de développer ou de rétracter les éléments du projet.

Les actions sur le navigateur sont réalisées en double cliquant sur les éléments (ouverture de l'élément) ou en cliquant avec le bouton droit (ajout d'un nouvel élément au projet, action spéciale sur un élément, etc…).

Certaines opérations sont réalisées en saisissant les éléments et en les déplaçant dans le navigateur (drag and drop).

Les couleurs (rappelées en général sur le fond des documents dans l'espace de travail) permettent d'identifier la famille des éléments.

L'arborescence du navigateur.

Folios

Un folio est une page sur laquelle est dessinée un programme ou une partie de programme.

La manipulation des folios est simplifiée à l'extrême dans AUTOMGEN. Les ordres de chaînages de folios nécessaires dans les versions précédentes ne sont plus utilisés. Pour que plusieurs folios soient compilés ensembles, il suffit qu'ils se trouvent dans le projet. Les icônes associées aux folios sont les suivantes :

- folio normal,
- folio normal (exclu de la compilation),
- folio contenant une expansion de macro-étape,
- folio contenant une encapsulation,
- folio contenant un programme de bloc-fonctionnel,
- folio contenant une tâche,
- folio contenant une tâche (exclu de la compilation).

Des icônes barrées d'une croix indiquent un folio fermé (non visible dans l'espace de travail). Double cliquer sur une icône de ce type ouvre (montre) le folio associé.

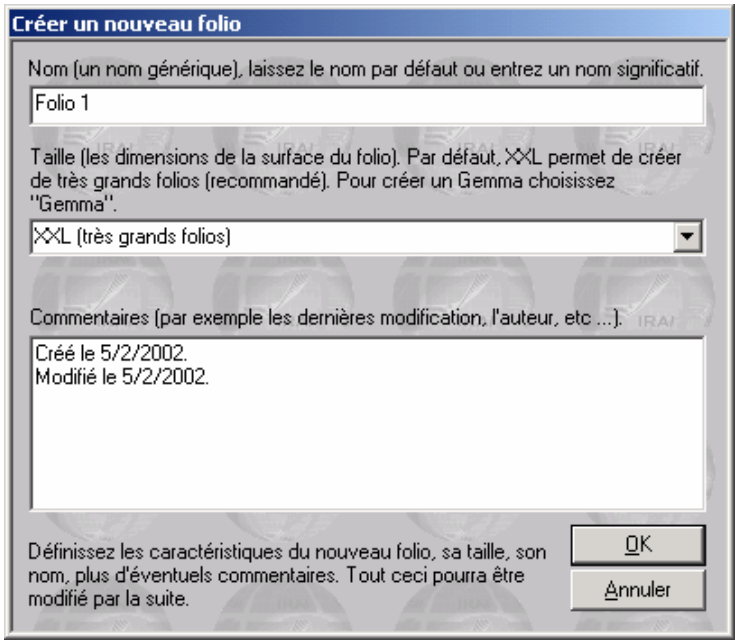
Ajouter un nouveau folio

Cliquez avec le bouton droit de la souris sur l'élément « Folios » dans le navigateur puis choisissez « Ajouter un nouveau folio ».

Choisissez la taille du folio (XXL est le format recommandé, les autres formats sont ceux des anciennes versions d'AUTOMGEN, GEMMA doit uniquement être utilisé pour créer un modèle GEMMA).

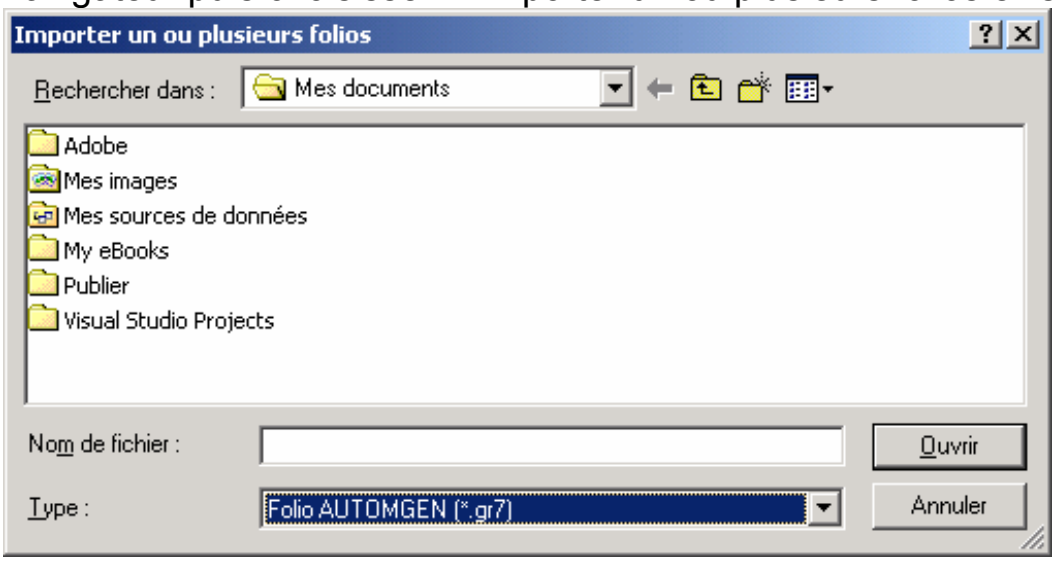
Le nom du folio peut être quelconque mais doit rester unique pour chaque folio du projet.

La zone commentaire est laissée à votre discrétion pour l'évolution des modifications ou autres informations relatives à chacun des folios.



Importer des folios d'anciennes versions d'AUTOMGEN, importer des folios CADEPA

Cliquez avec le bouton droit de la souris sur l'élément « Folios » dans le navigateur puis choisissez « Importer un ou plusieurs folios existants ».



Sélection d'un ou plusieurs folios à importer.

Choisissez dans la liste « Type » le type du folio à importer « AUTOMGEN » ou « CADEPA » puis cliquez sur OK.

Certaines restrictions sont applicables à l'importation des folios CADEPA :

- les numéros d'étapes doivent être uniques (pas d'utilisation du même numéro d'étape sur plusieurs folios),
- les renvois doivent être convertis en liens dans CADEPA avant de pouvoir être importés.

En maintenant la touche [CTRL] enfoncée, vous pouvez sélectionner et importer plusieurs folios.

Modifier l'ordre de compilation des folios

Les folios sont compilés dans l'ordre dans lequel ils sont listés dans le projet. Pour modifier cet ordre, cliquez sur un folio avec le bouton gauche de la souris dans le navigateur et déplacez le dans la liste.

Supprimer un folio de la liste

Cliquez avec le bouton droit de la souris sur le folio à supprimer dans le navigateur et choisissez « Supprimer » dans le menu.

Exporter un folio vers un fichier « .GR7 »

Cliquez avec le bouton droit de la souris sur le folio à supprimer dans le navigateur et choisissez « Exporter » dans le menu.

Copier, Couper, Coller un folio

Cliquez avec le bouton droit de la souris sur le folio dans le navigateur et choisissez « Copier / couper » dans le menu. Pour coller, cliquez avec le bouton droit de la souris sur l'élément « Folios » dans le navigateur et choisissez « Coller ».

Cette option permet de copier ou de transférer des folios d'un projet à un autre.

Renommer un folio

Voir ci-après « Modifier les propriétés ».

Modifier les propriétés d'un folio.

Cliquez avec le bouton droit de la souris sur le folio dans le navigateur et choisissez « Propriétés » dans le menu.

Vous pouvez modifier le nom du folio, la syntaxe utilisée pour le langage littéral et le nom des variables. L'option « Ne pas compiler ce folio » permet d'exclure le folio de la compilation. L'option « Afficher sous la forme d'un GEMMA » disponible uniquement si le format du folio est GEMMA permet d'afficher et de modifier le folio sous la forme d'un GEMMA. L'option « Interdire l'utilisation des entrées / sorties autres que les symboles définis » interdit l'utilisation des variables i, %i, o %q non attribuées à des symboles. La zone « commentaires » est laissée à votre discrétion. L'accès au folio peut être protégé par un mot de passe.

Symboles

La liste des symboles donne la correspondance entre des noms « symboliques » et des noms de variables. Un projet ne peut contenir qu'une seule table de symboles.

Créer une table de symboles

Cliquez avec le bouton droit de la souris sur l'élément « Symboles » dans le navigateur et choisissez « Créer une table de symboles » dans le menu.

Importer une table de symboles

Cliquez avec le bouton droit de la souris sur l'élément « Symboles » dans le navigateur et choisissez « Importer une table de symboles » dans le menu.

Configuration

Post-processeurs

Sous cette rubrique se trouvent tous les éléments de configuration des post-processeurs (voir le manuel de référence des post-processeurs pour plus d'informations).

Options du compilateur

Double cliquez sur cet élément pour modifier le réglage des options du compilateur.

Documentation

Permet d'accéder à la fonction d'impression de dossier (double clic sur l'élément « Impression »). Vous pouvez imprimer un dossier complet composé d'une page de garde, de la table de références croisées, de la liste des symboles et des folios. La fonction d'aperçu avant impression permet de visualiser l'ensemble de ces éléments.

Fichiers générés

Générer la liste des instructions en code pivot

En double cliquant sur l'élément « Code pivot » vous générez un listing en langage littéral bas niveau (le code pivot d'AUTOMGEN). L'observation du code généré est généralement réservée à des spécialistes soucieux de comprendre les méthodes de traductions utilisées par le compilateur.

Générer la liste des références croisées

Un double clic sur l'élément « Références croisées » génère et affiche la liste des variables utilisées dans l'application avec leurs éventuelles variables automates associées ainsi que le nom du ou des folios où elles sont utilisées.

Post-processeurs

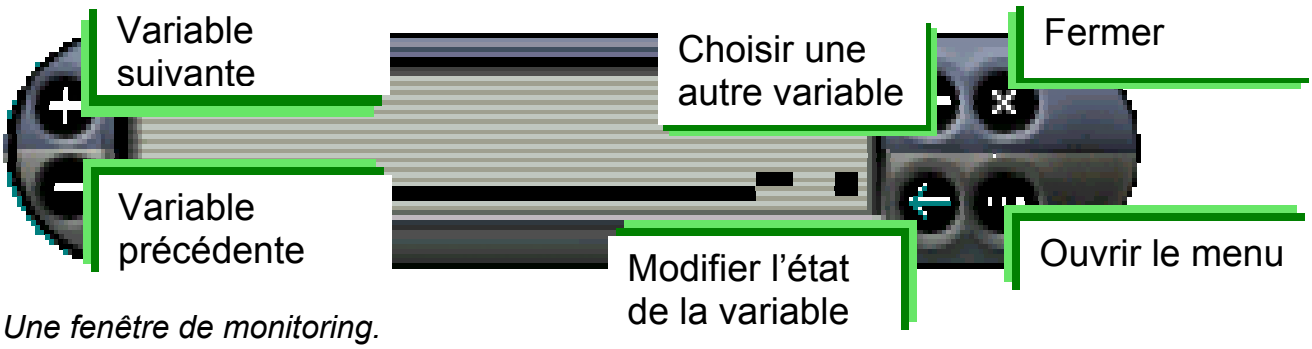
Les autres éléments concernent les fichiers générés par les post-processeurs : listes d'instructions en langage automate.

Mise au point

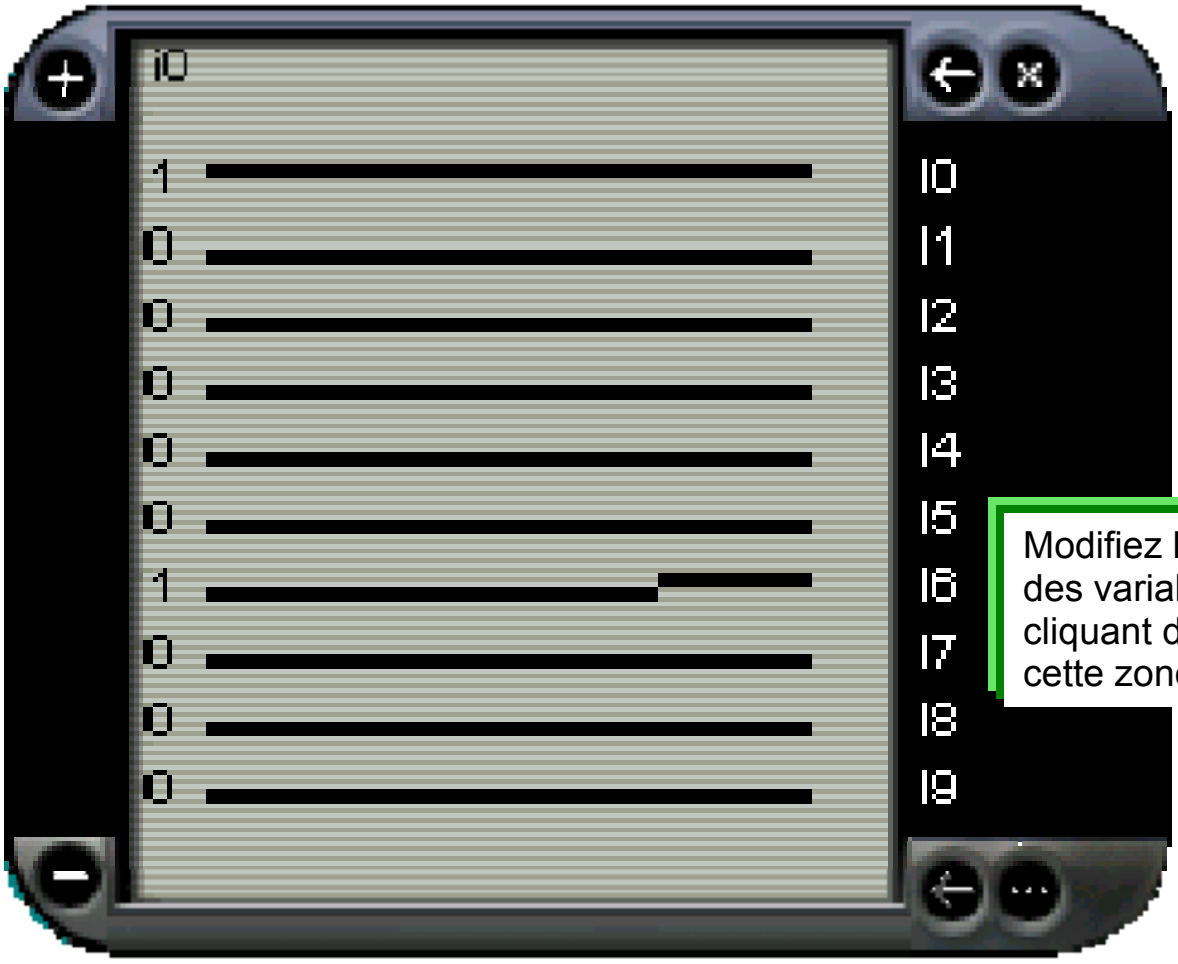
Regroupe des outils permettant la visualisation et la modification de l'état des variables.

Voir et modifier une variable ou une table de variables

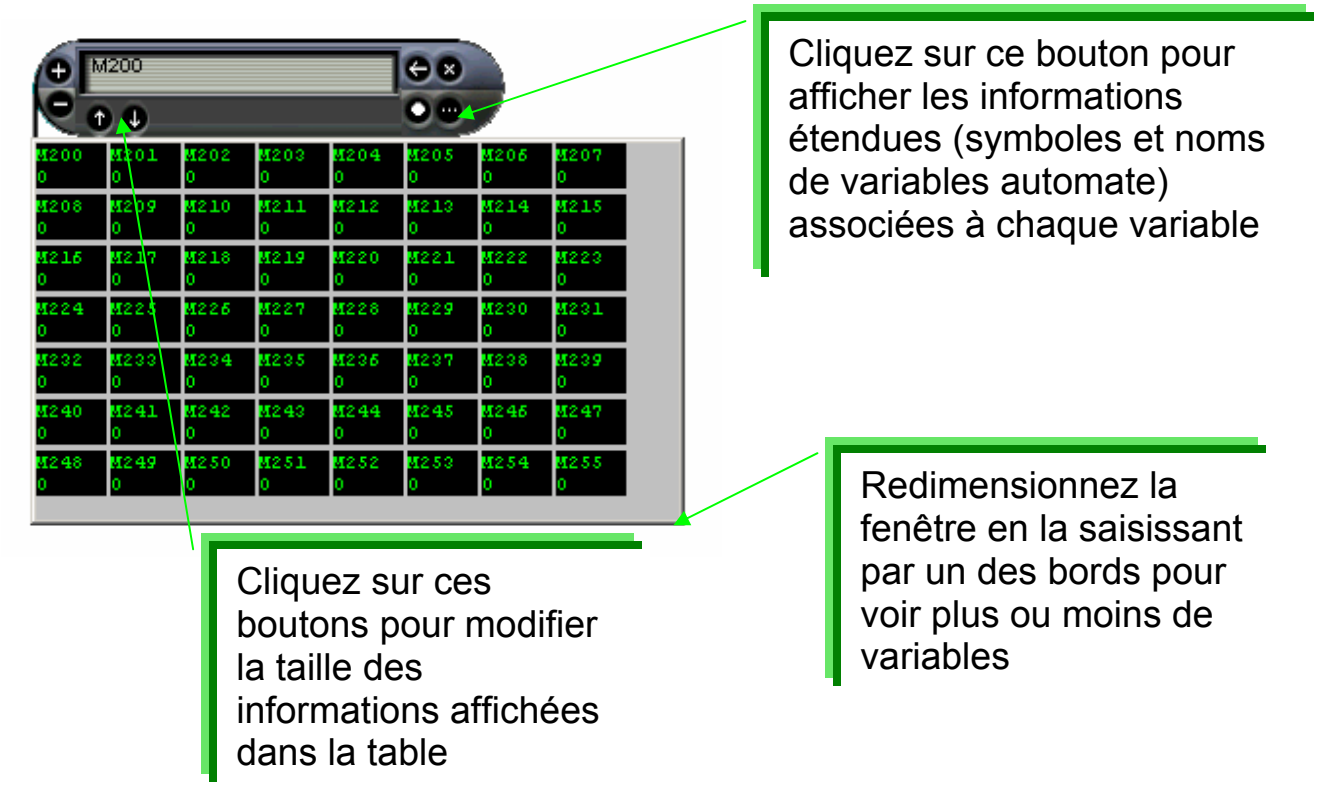
En cliquant avec le bouton droit sur l'élément « Mise au point » et en choisissant « Monitoring », vous ouvrez un élément vous permettant d'observer l'état d'une variable ou d'une table de 10 variables ou d'une table d'un nombre quelconque de variables.



Une fenêtre de monitoring.



La fenêtre monitoring en mode « Table de variables (10) ».



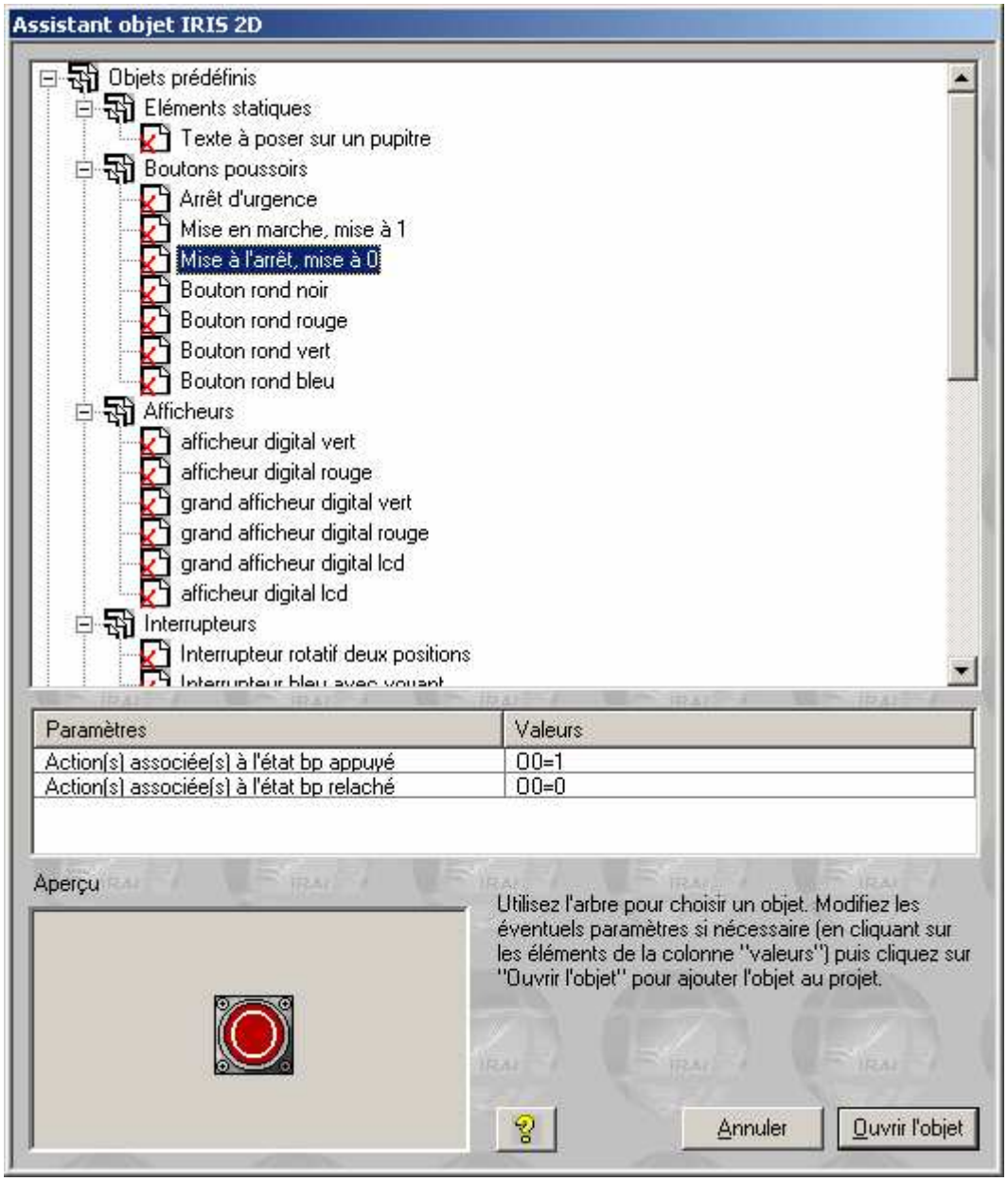
La fenêtre monitoring en mode « Table de variables »

Objets IRIS

Les objets IRIS 2D permettent de créer des pupitres, des applications de supervision et des applications de simulation de parties opératives 2D). IRIS 3D permet de créer des applications de simulation de parties opératives 3D. Chaque objet IRIS 2D apparaît dans l'arborescence du projet (voir les chapitres Références IRIS 2D et Références IRIS 3D pour plus de détails).

Ajouter un objet IRIS 2D

En cliquant avec le bouton droit de la souris sur l'élément « Ajouter un objet IRIS 2D ». L'assistant de sélection d'un objet vous permet de le choisir et de le paramétrer.



L'assistant de sélection d'un objet IRIS 2D.

Supprimer un objet IRIS 2D

Cliquez avec le bouton droit de la souris sur l'objet IRIS dans le navigateur et choisissez « Supprimer » dans le menu.

Montrer ou cacher un objet IRIS 2D

Cliquez avec le bouton droit de la souris sur l'objet IRIS dans le navigateur et choisissez « Montrer/cacher » dans le menu.

Copier, couper, coller un objet IRIS 2D

Cliquez avec le bouton droit de la souris sur l'objet IRIS dans le navigateur et choisissez « Copier » ou « Coller » dans le menu.

Pour coller un objet IRIS, cliquez avec le bouton droit de la souris sur l'élément « IRIS » dans le navigateur et choisissez « Coller ».

Pour coller un objet IRIS dans un pupitre, choisissez « Coller » dans le menu du pupitre ou cliquez avec le bouton droit de la souris sur le pupitre dans le navigateur et choisissez « Coller ».

Ajouter un nouvel objet IRIS 2D sur un pupitre

Choisissez « Ajouter un objet » dans le menu du pupitre ou cliquez avec le bouton droit de la souris sur le pupitre dans le navigateur et choisissez « Ajouter un objet sur le pupitre » dans le menu (pour plus de détails sur le pupitre, reportez vous au chapitre L'objet « Pupitre »)

Modifier les propriétés d'un objet IRIS 2D

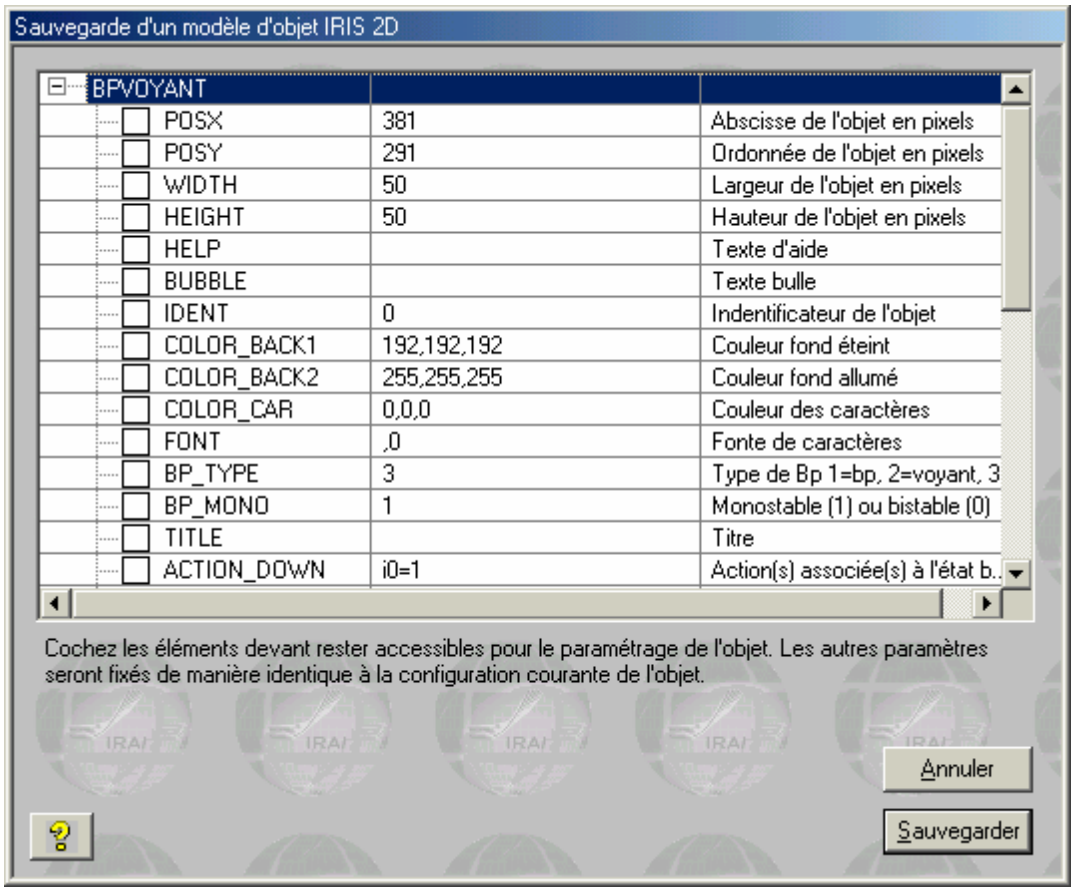
Cliquez avec le bouton droit de la souris sur l'objet IRIS dans le navigateur et choisissez « Propriétés ». Pour les objets de plus haut niveau (les objets parents), des propriétés spéciales sont accessibles :

Les propriétés des objets de haut niveau.

La visibilité détermine sous quelle condition l'objet est visible ou caché. L'option de réinitialisation permet de replacer un objet dans son état initial à l'enclenchement de la visualisation dynamique (typiquement utilisé pour les applications de simulation de PO).

Définir un modèle d'objet accessible dans l'assistant

Cliquer sur l'objet IRIS avec le bouton droit de la souris dans le navigateur et choisissez « Enregistrer comme modèle » dans le menu.



La sélection des paramètres modifiables par les utilisateurs de vos modèles.

Vous pouvez sélectionner la liste des paramètres devant rester accessibles à l'utilisateur dans l'assistant. En cliquant sur « Sauvegarder », vous enregistrer votre modèle d'objet. Le répertoire de stockage des modèles d'objet est « <répertoire d'installation d'AUTOMGEN>\i2d\lib ». Un sous répertoire nommé « mes objets » vous est réservé pour sauvegarder vos propres modèles.

Importer un objet IRIS 2D d'une version précédente d'AUTOMGEN

Cliquez avec le bouton droit de la souris sur l'élément « IRIS » dans le navigateur et choisissez « Importer des objets IRIS 2D ». Sélectionnez un ou plusieurs fichiers « .AOF ».

Créer un pupitre IRIS 3D

Cliquez avec le bouton droit de la souris sur l'élément « IRIS » dans le navigateur et choisissez « Ajouter un pupitre IRIS 3D » (voir le chapitre consacré à IRIS 3D pour plus de détails).

Ressources

Cet élément du projet permet d'ajouter tout type de fichier au projet. Les fichiers ainsi ajoutés feront partie intégrante du projet et seront sauvegardés avec les autres éléments. Pour faire référence au pseudo répertoire où se trouvent les ressources, le mot clé « <RESDIR> » peut être utilisé dans le nom des répertoires spécifiés dans AUTOMGEN. Les objets IRIS peuvent par exemple faire référence à des bitmaps se trouvant dans les ressources.

Ajouter un fichier dans les ressources

Cliquez avec le bouton droit de la souris sur l'élément « Ressources » dans le navigateur et choisissez « Ajouter » dans le menu.

Supprimer un fichier contenu dans les ressources

Cliquez avec le bouton droit de la souris sur le fichier ressource dans le navigateur et choisissez « Supprimer ».

Renommer un fichier contenu dans les ressources

Cliquez avec le bouton droit de la souris sur le fichier ressource dans le navigateur et choisissez « Renommer ».

Modifier un fichier contenu dans les ressources

Cliquez avec le bouton droit de la souris sur le fichier ressource dans le navigateur et choisissez « Modifier ».

Ajouter et convertir des fichiers 3D STUDIO dans les ressources

Les fichiers 3D STUDIO peuvent être convertis en fichiers .x et ajoutés dans les ressources en cliquant avec le bouton droit de la souris sur l'élément « Ressources » du navigateur et en choisissant « Importer un ou des fichiers 3D » (voir le chapitre Références IRIS 3D pour plus de détails).

Modules externes

Ces éléments sont réservés à des modules exécutables développés par des tiers et interfacés avec AUTOMGEN.

Résumé navigateur

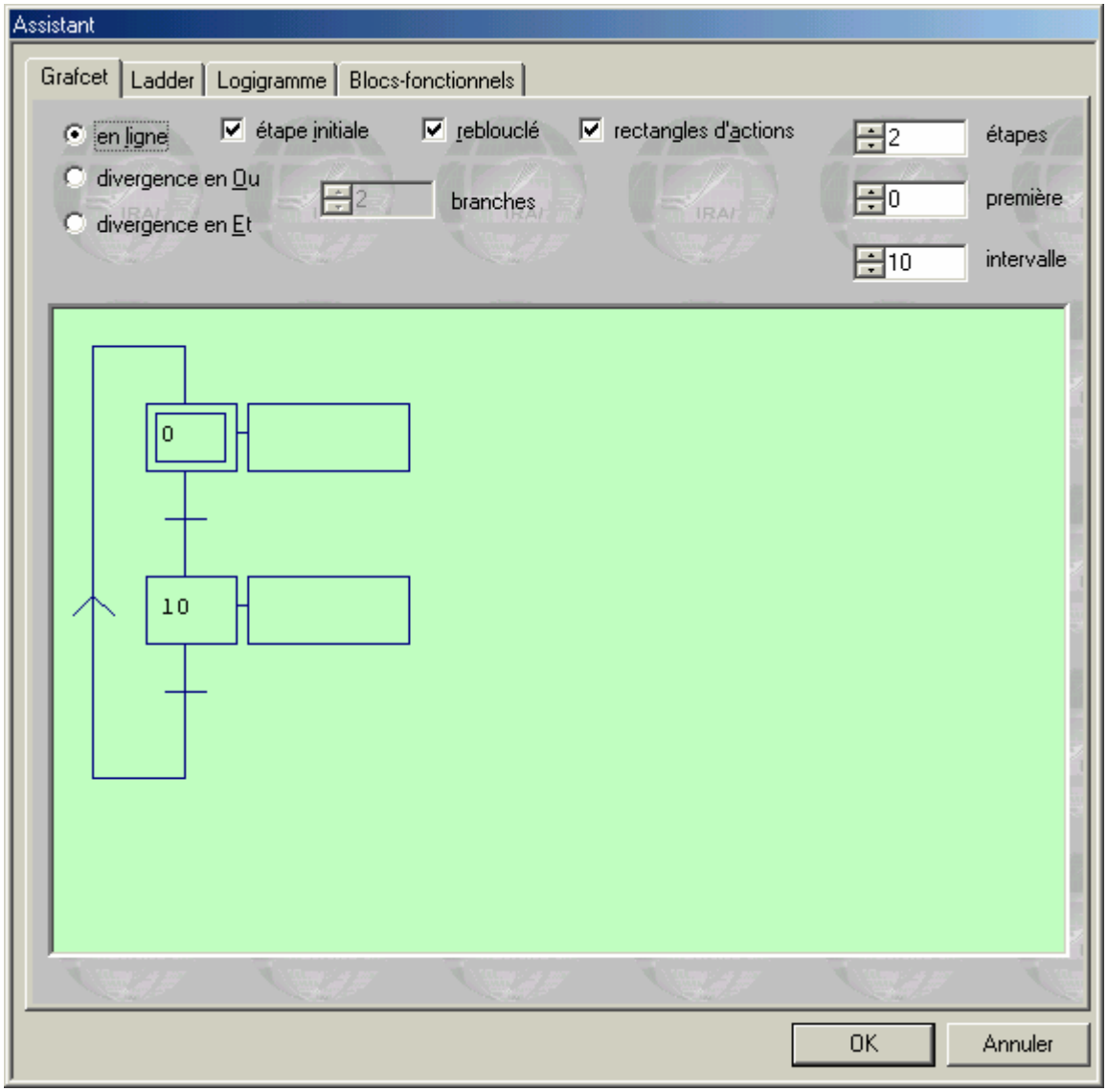
Le navigateur permet de voir et de manipuler l'ensemble des éléments d'un projet. En double cliquant sur les éléments ou en cliquant avec le bouton droit de la souris, on accède aux différentes fonctions applicables à chaque élément.

Dessiner des programmes

Pour dessiner des programmes, plusieurs outils sont à votre disposition :

Dessiner avec l'assistant

C'est sans doute le plus simple lorsqu'on débute avec AUTOMGEN. Cliquez avec le bouton droit de la souris sur un folio ouvert dans l'espace de travail et choisissez « Assistant » dans le menu. Laissez vous ensuite guider dans les choix. Lorsque vous avez fini, cliquez sur « OK » et poser le dessin sur le folio en cliquant avec le bouton gauche de la souris.



L'assistant.

Dessiner avec le menu contextuel

En cliquant avec le bouton droit de la souris sur un folio ouvert dans l'espace de travail, le menu vous propose une série d'éléments que vous pouvez poser sur le folio. C'est un mode de création instinctif et rapide.

Dessiner avec la palette

En sélectionnant des éléments dans la palette, vous pouvez créer rapidement des programmes à partir d'éléments déjà créés.

Enrichir et personnaliser la palette

La définition de la palette est composée de fichiers « .GR7 » se trouvant dans le répertoire « <répertoire d'installation d'AUTOMGEN>\pal ». Vous pouvez effacer, modifier, renommer ou ajouter des fichiers. Pour générer des fichiers « .GR7 », utiliser la commande « Exporter » en cliquant avec le bouton droit de la souris sur un folio dans le navigateur. Les noms affichés dans la palette sont ceux des fichiers « .GR7 ». Relancez AUTOMGEN pour que le nouvel élément apparaisse dans la palette.

Dessiner avec les touches du clavier

Chaque touche est associée à un des blocs de dessin. L'élément « Blocs » de la palette donne également accès à l'ensemble de ces blocs. Le tableau ci-dessous liste les blocs et leur utilisation.

Bloc d'effacement

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[A]	Effacement	Permet de remettre à blanc une cellule	Tous

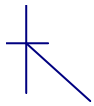
Blocs de liaison

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[E]	Liaison verticale	Liaison haut vers bas ou bas vers haut	Tous
	[F]	Liaison horizontale	Liaison droite vers gauche ou gauche vers droite	Tous

	[G]	Coin supérieur gauche	Liaison bas vers droite ou gauche vers bas	Tous
	[H]	Coin supérieur droit	Liaison bas vers gauche ou droite vers bas	Tous
	[I]	Coin inférieur gauche	Liaison haut vers droite ou gauche vers haut	Tous
	[J]	Coin inférieur droit	Liaison haut vers gauche ou droite vers haut	Tous
	[Z]	Croisement	Croisement de deux liaisons	Tous

Blocs Grafcet

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[B]	Etape	Etape normale	Grafcet
	[C]	Etape initiale sans activation	Etape initiale sans activation	Grafcet
	[D]	Etape initiale	Etape initiale	Grafcet
		Macro-étape	Accessible dans le menu contextuel uniquement	Grafcet
	[+]	Etape encapsulante	Une encapsulation doit être associée	Grafcet
	[-]	Etape encapsulante initiale	Une encapsulation doit être associée	Grafcet
		Marqueur d'état initial	Définit l'état initial pour une encapsulation	Grafcet
	[T]	Transition	Transition	Grafcet
	[$]	Transition source	Peut remplacer le symbole transition	Grafcet

	[£]	Transition puit	Peut remplacer le symbole transition	Gracet
		Lien pour action sur franchissement de transition	Utilisez l'élément suivant pour dessiner le rectangle d'action	Grafcet
		Début de rectangle d'action au franchissement de transition	Utilisez les éléments [X] et [Y] pour terminer le rectangle	Grafcet
	[K]	Bord gauche d'une divergence en « Et »	Obligatoirement à gauche des divergences en « Et »	Grafcet
	[L]	Branche supplémentaire d'une divergence en « Et » ou convergence en « Et »	Ne pas utiliser comme bord gauche ou droit d'une divergence en « Et »	Grafcet
	[M]	Bord droit d'une divergence en « Et »	Obligatoirement à droite d'une divergence en « Et »	Grafcet
	[N]	Prolongation d'une divergence en « Et »	Se place entre les blocs [K], [L], [M], [P] ou [O],[P],[Q], [L]	Grafcet
	[O]	Bord gauche d'une convergence en « Et »	Obligatoirement à gauche d'une convergence en « Et »	Grafcet
	[P]	Branche supplémentaire d'une convergence en « Et » ou divergence en « Et »	Ne pas utiliser comme bord gauche ou droit d'une convergence en « Et »	Grafcet
	[Q]	Bord droit d'une convergence en « Et »	Obligatoirement à droite d'une convergence en « Et »	Grafcet
	[R]	Divergence en « Ou »	Ne pas utiliser comme bord d'une convergence en « Ou »	Grafcet

	[S]	Convergence en « Ou »	Ne pas utiliser comme bord d'une divergence en « Ou »	Grafcet
	[U]	Saut ou reprise d'étape à gauche	Convergence ou divergence en « Ou »	Grafcet
	[V]	Saut ou reprise d'étape à droite	Convergence ou divergence en « Ou »	Grafcet
	[ESPACE] sur un bloc [E]	Liaison orientée vers le haut	Pour les rebouclages et les reprises d'étapes	Grafcet

Blocs Logigrammes

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[0] (zéro)	Affectation logigramme	Sépare la zone « test » de la zone « action »	Logigrammes
	[1]	Fonction « Pas »	Complémente le signal d'entrée du bloc	Logigrammes
	[2]	Fonction « Et »	Combine les entrées en un « Et » logique	Logigrammes
	[3]	Fonction « Ou »	Combine les entrées en un « Ou » logique	Logigrammes
	[4]	Milieu de bloc	Agrandit un bloc fonction « Et » ou « Ou »	Logigrammes
	[5]	Bas de bloc	Termine un bloc fonction « Et » ou « Ou »	Logigrammes

Blocs Ladder

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[(]	Partie gauche bobine	Débute une action	Ladder
	[)]	Partie droite bobine	Termine une action	Ladder

	[U]	Bord gauche	Termine le schéma	Ladder
	[V]	Bord droit	Débute le schéma	Ladder
	[R]	Connexion	Fonction « Ou »	Ladder
	[S]	Connexion	Fonction « Ou »	Ladder

Blocs Action

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[W]	Bord gauche rectangle d'action	Débute une action	Grafcet et Logigrammes
	[X]	Milieu d'un rectangle d'action	Prolonge une action	Grafcet et Logigrammes
	[Y]	Bord droit d'un rectangle d'action	Termine une action	Grafcet et Logigrammes
	[.]	Bord gauche d'un rectangle d'action double	Débute un rectangle d'action double	Grafcet et Logigrammes
	[/]	Milieu d'un rectangle d'action double	Prolonge un rectangle d'action double	Grafcet et Logigrammes
	[%]	Bord droit d'un rectangle d'action double	Termine un rectangle d'action double	Grafcet et Logigrammes
	[S]	Divergence Action	Permet de juxtaposer des rectangles d'action verticalement	Grafcet et Logigrammes
	[V]	Divergence Action	Permet de juxtaposer des rectangles d'action verticalement	Grafcet et Logigrammes
	[#]	Action sur activation	Définit le type d'action	Grafcet

	[_]	Action sur désactivation	Définit le type d'action	Grafcet
	[@]	Action évènementielle	Définit le type d'action	Grafcet

Blocs Test

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[7]	Bord gauche d'un test	Débute un test	Logigrammes et Ladder
	[6]	Bord droit d'un test	Termine un test	Logigrammes et Ladder

Blocs Organigramme

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[<]	Entrée d'organigramme	Marque l'entrée dans un rectangle	Organigrammes
	[=]	Sortie « Faux »	Sortie si faux d'un rectangle de test	Organigrammes

Blocs Bloc fonctionnel

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[8]	Coin supérieur gauche d'un bloc fonctionnel	Débute le nom du bloc fonctionnel	Bloc fonctionnel
	[9]	Coin supérieur droit d'un bloc fonctionnel	Termine le nom du bloc fonctionnel	Bloc fonctionnel
	[:]	Coin inférieur gauche d'un bloc fonctionnel	Ajoute une entrée au bloc fonctionnel	Bloc fonctionnel
	[;]	Bord gauche d'un bloc fonctionnel	Ajoute une entrée au bloc fonctionnel	Bloc fonctionnel
	[>]	Bord droit d'un bloc fonctionnel	Ajoute une sortie au bloc fonctionnel	Bloc fonctionnel

	[?]	Coin inférieur droit d'un bloc fonctionnel	Ajoute une sortie au bloc fonctionnel	Bloc fonctionnel

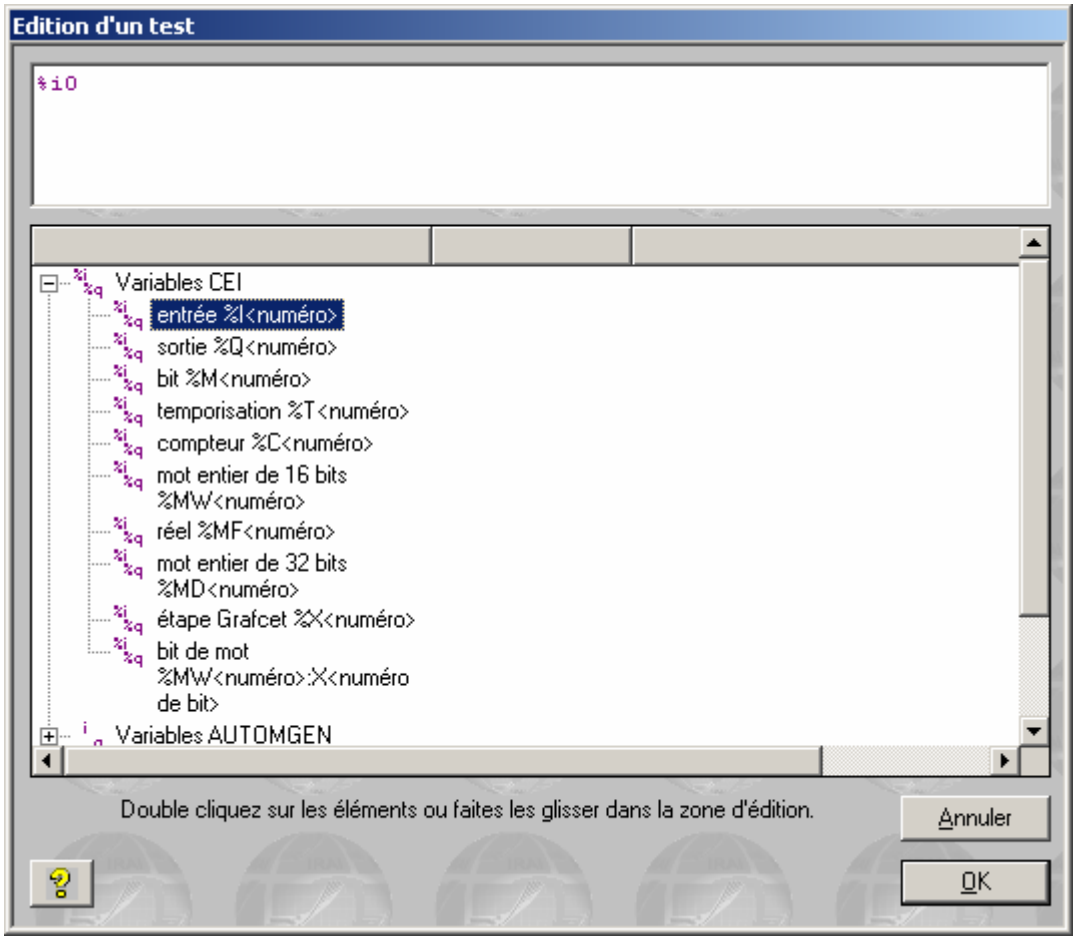
Autres blocs

Aspect	Touche associée	Nom générique	Commentaires	Langages
	[*]	Lien combinatoire / transition	Ce bloc est un lien entre les langages Logigrammes ou Ladder et le langage Grafcet	Grafcet / Logigrammes / Ladder

Documenter les éléments de programme

Pour documenter les éléments de programme, cliquez dessus avec le bouton gauche de la souris. Pour créer des commentaires, cliquez sur un espace vide du folio. Pour valider les modifications, appuyez sur la touche [Entrée] ou cliquez en dehors de la zone d'édition avec le bouton gauche de la souris. Pour annuler les modifications, appuyez sur la touche [Echap] ou cliquez avec le bouton droit de la souris en dehors de la zone d'édition.

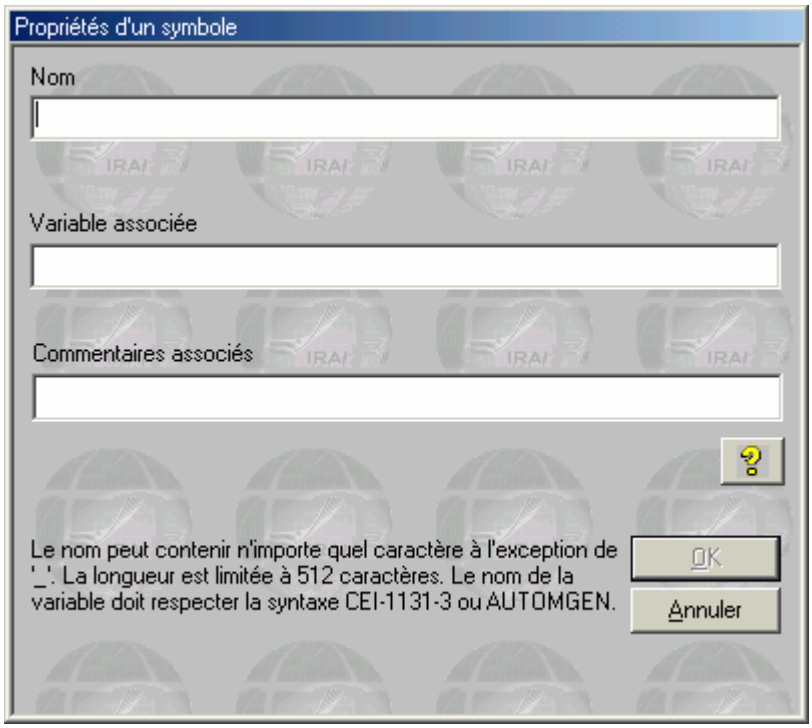
En édition des tests et des actions, un bouton « … » apparaît sous la zone d'édition. En cliquant dessus, vous accédez à un assistant de création de tests ou d'actions.



L'assistant de création des tests.

Ajouter des symboles

Pour créer un symbole, cliquez avec le bouton droit de la souris sur la table des symboles dans l'espace de travail et choisissez « Ajouter ». Ou cliquez sur le bouton S+ dans la barre d'outils. Vous pouvez également lancer la compilation d'un programme contenant des symboles non définis. Les variables correspondantes aux symboles vous seront alors demandées pendant la compilation.



Attribution des symboles pendant la compilation.

Résumé dessin des programmes

Pour dessiner facilement un programme, créez un nouveau folio, puis cliquez avec le bouton droit de la souris sur le fond du folio. Choisissez « Assistant » dans le menu puis laissez vous guider par l'assistant.

Exécuter une application

Exécuter facilement une application

Le bouton  de la barre d'outils, représente la méthode la plus rapide pour observer le résultat de l'exécution d'une application. Ce bouton poussoir active les mécanismes suivants :

- compilation de l'application si elle n'est pas à jour (pas déjà compilée depuis les dernières modifications),
- installation du module d'exécution (avec téléchargement si la cible courante est un automate et suivant les options de connexions),
- passage de la cible en RUN,
- activation de la visualisation dynamique.

Mettre fin à l'exécution

Cliquez sur  . Sur cible automate, le programme continu à s'exécuter sur la cible. Sur exécuteur PC, le programme est stoppé.

Uniquement compiler

Cliquez sur  .

Stopper la compilation

Cliquez sur  .

Se connecter à un automate ou installer l'exécuteur PC

Cliquez sur  .

Se déconnecter d'un automate ou désinstaller l'exécuteur PC

Cliquez sur  .

Mettre la cible en mode RUN

Cliquez sur  .

Mettre la cible en mode STOP

Cliquez sur  .

Initialiser la cible

Cliquez sur  .

Faire un cycle programme sur la cible (généralement non supporté sur les automates)

Cliquez sur  .

Activer la visualisation dynamique

Cliquez sur  .

Résumé Exécuter une application

Pour exécuter une application, cliquez sur le bouton « GO ». Pour mettre fin à l'exécution, cliquez de nouveau sur le même bouton.

Le compilateur

Le compilateur traduit les folios en un ensemble d'équations de langage pivot (visualisables en double cliquant sur l'élément « Code généré / langage pivot » dans le navigateur).

Le langage pivot est ensuite traduit en langage exécutable par un post-processeur (le post-processeur courant est visible et sélectionnable par un double clic dans le volet « Cibles » accessible en cliquant sur l'onglet « Cibles » en bas de la fenêtre où se trouve le navigateur).

Modifier les options du compilateur

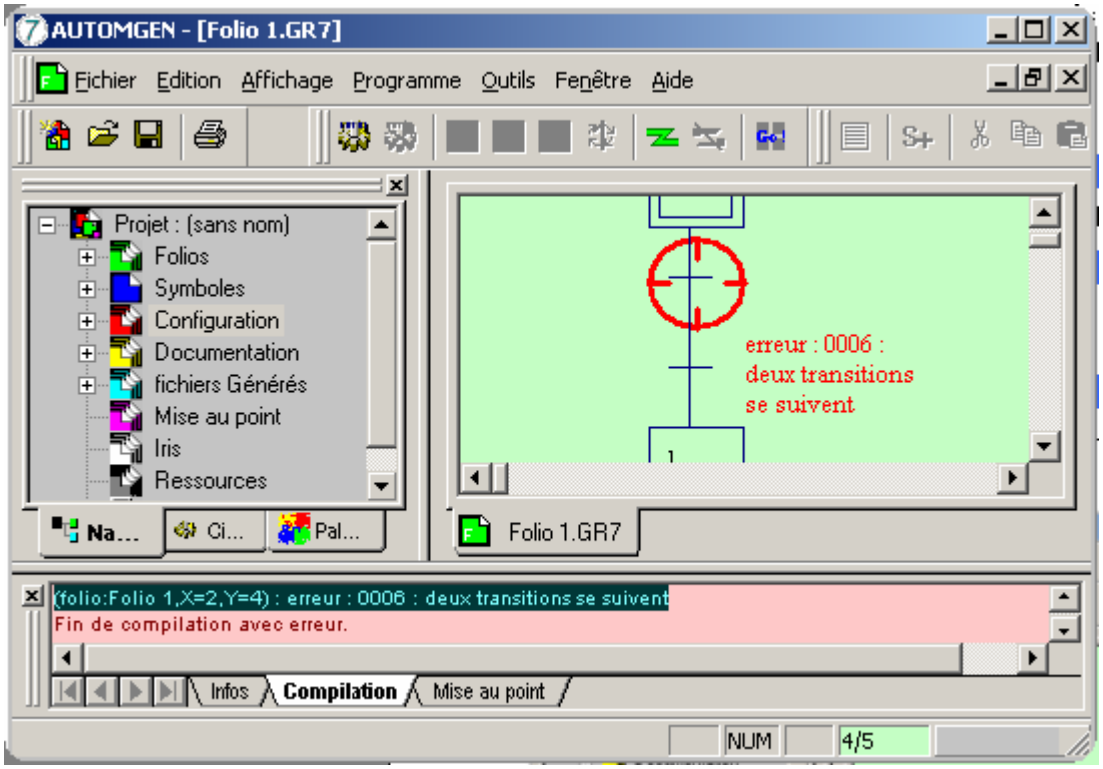
Double cliquez sur l'élément « Configuration / Options du compilateur ».

Visualiser les messages de compilation

Le volet « Compilation » de la fenêtre des messages contient les comptes rendus de la dernière compilation.

Localiser une erreur

En double cliquant sur messages d'erreurs, vous pouvez en localiser la source.



Un message d'erreur et sa source.

Si les fenêtres de messages sont cachées (en mode « Débutant » par exemple) et qu'une ou plusieurs erreurs sont détectées par le compilateur, alors une boîte de dialogue annonce la première erreur détectée (pour faire apparaître les fenêtres de messages : utilisez la commande « Messages » du menu « Fenêtres »).

Résumé Compilateur

A la fin de la compilation, la fenêtre « Compilation » donne la liste des éventuelles erreurs. En double cliquant sur les messages d'erreur, l'emplacement du programme qui a provoqué l'erreur est affiché.

Exécution des programmes sur PC

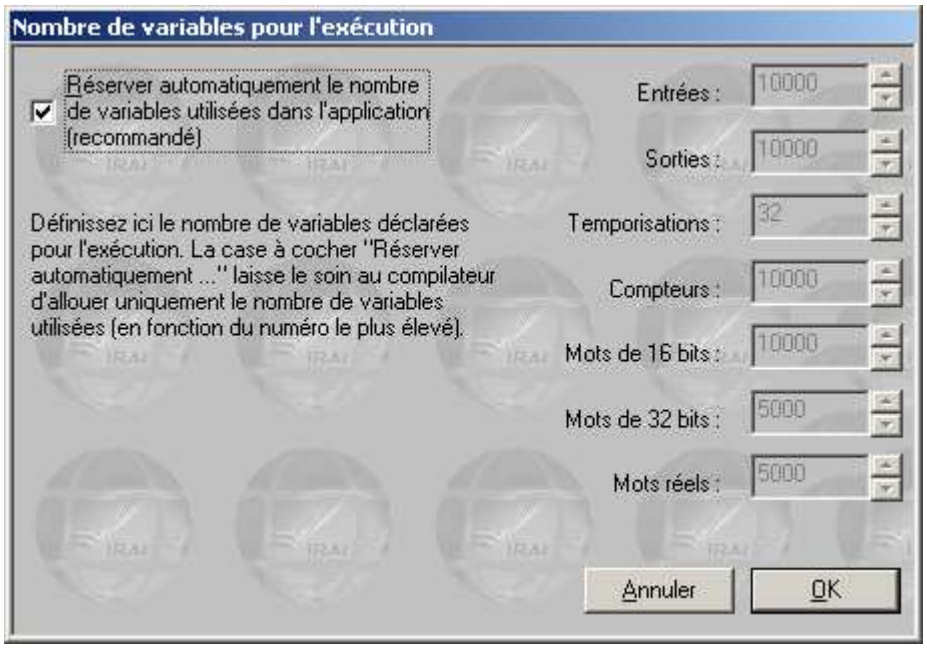
La cible « Exécuteur PC » est un véritable automate embarqué dans votre PC.

Vous pouvez :

- tester vos applications,
- piloter une partie opérative virtuelle réalisée avec IRIS 2D ou 3D,
- piloter des cartes d'entrées / sorties connectées au PC.

Configurer le nombre de variables

Double cliquez sur l'élément « Configuration / Post-processeurs / Exécuteur PC / Variables ».



Sélection du nombre de variables

Par défaut l'espace nécessaire pour les variables utilisées dans l'application est automatiquement réservé. Vous pouvez sélectionner manuellement la quantité de mémoire réservée pour chaque type de variables. Ceci peut être nécessaire si un adressage indexé est utilisé pour accéder à une table de variable.

Variables système de l'exécuteur PC

Les bits 0 à 99 et les mots 0 à 199 sont des variables système à ne pas utiliser comme variable utilisateur dans vos applications. Les deux tableaux ci-dessous donnent le détail des variables systèmes de l'exécuteur PC.

Bits	Utilisation
0	actif au premier cycle, activation des étapes Grafcet initiales
1 à 4	réservés aux drivers d'E/S
5 à 7	réservés aux erreurs pour les drivers d'E/S
8	erreur sur débordement de chien de garde si égal à 1
9 et 10	erreur : défaillance générale de l'exécuteur
11	mode de marche : 1=RUN, 0=STOP
12	arrêt d'urgence : passe à 1 en cas d'erreur ou peut être forcé à 1 pour bloquer le programme
13 à 29	réservés aux drivers
30	bit associé au timer 1
31	bit associé au timer 2
32	bit associé au timer 3
33	bit associé au timer 4
34	bit associé au timer 5
35	bit de reprise secteur (passe à 1 sur reprise secteur, la r.a.z. est à la charge du programmeur)
36	la mise à 1 de ce bit provoque une lecture de l'horloge temps réel et un transfert dans les mots Système 4, 5, 6, 7, 8, 51 et 52
37	la mise à 1 de ce bit provoque l'écriture des mots Système 4, 5, 6, 7, 8, 51 et 52 dans l'horloge temps réel
38 à 55	réservés
56	division par zéro
57 à 67	réservés pour les versions futures
68 à 99	réservés pour la pile des traitements booléens

Mots	Utilisation
0	réservé pour la partie haute du résultat de la multiplication ou pour le reste de la division
1 à 3	timers en millisecondes
4	timer en 1/10 seconde
5	timer en secondes
6	timer en minutes
7	timer en heures
8	timer en jours

9 à 29	réservés aux drivers E/S
30	compteur du timer 1
31	compteur du timer 2
32	compteur du timer 3
33	compteur du timer 4
34	compteur du timer 5
35	consigne du timer 1
36	consigne du timer 2
37	consigne du timer 3
38	consigne du timer 4
39	consigne du timer 5
40	partie basse de la référence horloge
41	partie haute de la référence horloge
42 à 50	réservés aux drivers E/S
51	timer en mois
52	timer en années

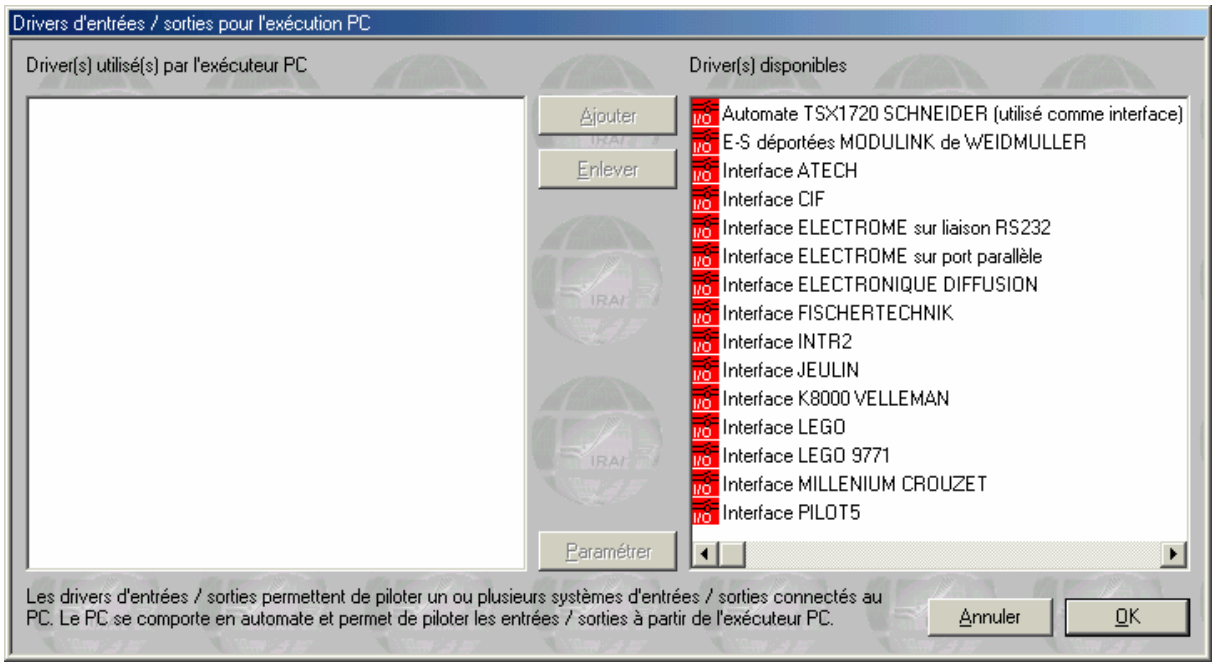
Modifier la période d'exécution

Double cliquez sur « Post-processeur / Exécuteur PC / Exécution ».

Réglage de la période d'exécution.

Piloter des entrées / sorties

Double cliquez sur « Configuration / Post-processeur / Exécuteur PC / Drivers d'E/S ».



Ajout d'un driver d'E/S

Sélectionnez un driver dans la liste de droite, puis cliquez sur « Ajouter ».

« Paramétrer » permet de configurer certains drivers.

Résumé Exécution sur PC

L'exécuteur PC transforme votre PC en automate programmable, il permet de piloter des entrées / sorties connectées directement à votre ordinateur.

Références IRIS 2D

Les objets IRIS 2D vous permettent de créer des applications de supervision et de simulation de parties opératives en 2D.

Le lien entre les objets et l'application d'automatisme est toujours réalisé par échange de l'état des variables.

Les objets IRIS 2D sont encapsulés dans des fenêtres WINDOWS.

Un objet IRIS 2D

Les objets IRIS 2D ont deux états possibles : le mode « Configuration » (on peut modifier les caractéristiques de l'objet) et le mode « Utilisateur » (on utilise l'objet). Le mode « Utilisateur » est également appelé mode « Exploitation ».

Modifier la visibilité des objets

Les objets peuvent être cachés ou visibles. Cette propriété peut être spécifiée à l'ouverture de l'objet ou sur changement d'état de la visualisation dynamique dans l'environnement. Seuls les objets de plus haut niveau (pas les objets posés sur un pupitre) peuvent être montrés ou cachés. Les objets posés sur un pupitre sont montrés ou cachés en même temps que le pupitre.

Pour modifier en dynamique la visibilité des objets, cliquez avec le bouton gauche de la souris sur les objets dans le navigateur et choisissez « Montrer / Cacher ».

Pour modifier les propriétés de visibilité, cliquez avec le bouton gauche de la souris sur les objets dans le navigateur et choisissez « Propriétés ».

Les propriétés de visibilité d'un objet.

Modifier les caractéristiques d'un objet

Effacer un objet

Méthode 1 : cliquez sur le bouton ☒ qui se trouve sur la surface de l'objet.
Méthode 2 : cliquez avec le bouton droit de la souris sur l'objet dans le navigateur et choisissez « Supprimer » dans le menu.

Dimensionner un objet

En saisissant l'objet par un de ses bords, vous pouvez l'étirer ou le rétrécir (on peut modifier précisément la taille d'un objet en accédant à ses propriétés, voir plus loin).

Déplacer un objet

Saisissez l'objet en cliquant avec le bouton gauche de la souris sur la mini barre se trouvant en haut de sa surface.

Passer un objet en mode « Utilisateur »

Méthode 1 : cliquez sur le bouton ☒ qui se trouve sur l'objet avec le bouton gauche de la souris.
Méthode 2 : cliquez avec le bouton droit de la souris sur l'objet.

Passer un objet en mode « Configuration »

Cliquez avec le bouton droit de la souris sur l'objet.

Modifier les caractéristiques d'un objet

Méthode 1 : cliquez sur le bouton .
Méthode 2 : enfoncez la touche [CTRL] du clavier et cliquez avec le bouton droit de la souris sur l'objet, relâchez ensuite la touche [CTRL].

Méthode 3 : cliquez avec le bouton droit de la souris sur l'objet dans le navigateur et choisissez « Propriétés » dans le menu.

Verrouillez l'accès à la configuration de tous les objets

Cliquez avec le bouton droit de la souris sur « Projet » dans le navigateur, choisir « Propriétés » et cochez la case « Interdire la configuration des objets IRIS 2D » dans l'onglet « Avancé ».

Objets de base, objets prédéfinis

Les objets de bases définissent des grands types de fonctionnalités. Les objets prédéfinis s'appuient sur un type de base et une configuration pour répondre à un besoin spécifique. Par exemple, un bouton poussoir d'arrêt d'urgence est un objet dérivé de l'objet de base permettant de réaliser des boutons poussoirs et des voyants. Pour accéder aux objets prédéfinis, utilisez l'assistant en cliquant avec le bouton droit de la souris sur l'élément « IRIS » dans le navigateur et en choisissant « Ajouter un objet IRIS 2D ».

Liste des objets de base

L'objet « Pupitre »

L'objet pupitre est le seul qui peut contenir d'autres objets sur sa surface. Vous l'utiliserez pour créer pupitres de commandes et surfaces d'animation pour parties opératives virtuelles. Cet objet possède un bouton poussoir ... qui permet de gérer les objets sur sa surface : ajout, déplacement, suppression, etc …

L'objet « Bouton et voyant »

Vous l'utiliserez pour créer boutons poussoirs et voyants en interactions avec des variables de l'application d'automatisme.

L'objet « Objet »

C'est un élément polymorphique que vous utiliserez principalement pour simuler des parties opératives.

L'objet « Valeur digitale »

Vous l'utiliserez pour afficher des valeurs numériques de l'application d'automatisme sous forme de nombres.

L'objet « Ecran, clavier, liste de messages »

Vous l'utiliserez pour afficher des informations de l'application d'automatisme sous forme de textes.

L'objet « Son »

Vous l'utiliserez pour produire des sorties sonores sur changement de l'état des variables de l'application d'automatisme.

L'objet « Archivage de données »

Vous l'utiliserez pour afficher sous forme de tables ou de courbes et sauvegarder dans le mémoire de l'ordinateur ou sur le disque des données de l'application d'automatisme.

L'objet « Programme »

Vous l'utiliserez pour effectuer des traitements qui s'exécuteront indépendamment de l'application d'automatisme.

L'objet « Boîte de dialogue »

Vous l'utiliserez pour afficher des messages sous forme de fenêtres surgissant sur changement d'état de variables de l'application d'automatisme.

L'objet « Valeur analogique »

Vous l'utiliserez pour afficher sous forme d'éléments analogiques (barres, cadrans, etc …) des variables numériques de l'application d'automatisme.

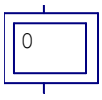
Prise en main

Ce chapitre va vous permettre de créer très rapidement votre première application IRIS 2D. Nous allons créer un pupitre, y déposer un bouton poussoir et lier l'objet aux variables de l'application d'automatisme.

Etape 1

Créons une application minimale avec AUTOMGEN voir chapitre Dessiner des programmes.

Il s'agit d'un Grafcet comportant une étape tel que montré ci-après.



Etape 2

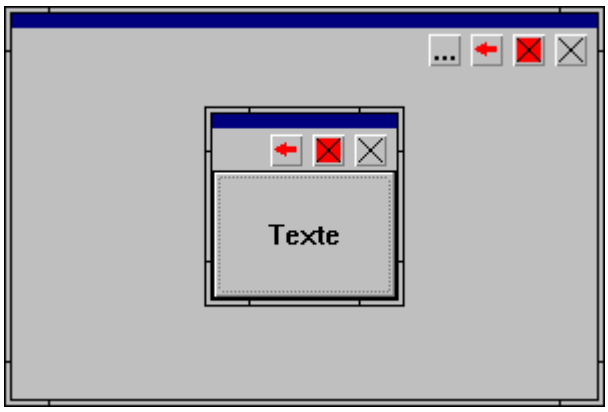
Lancez l'exécution de l'application AUTOMGEN (cliquez sur le bouton « Go » dans la barre d'outils).

Etape 3

Cliquez avec le bouton droit de la souris dans le navigateur sur l'élément « IRIS » puis choisissez « Ajoutez un objet IRIS 2D » dans le menu. Dans la catégorie « Objets de base », double cliquez sur « Pupitre ».
A ce stade l'objet doit apparaître à l'écran sous cette forme :

Etape 4

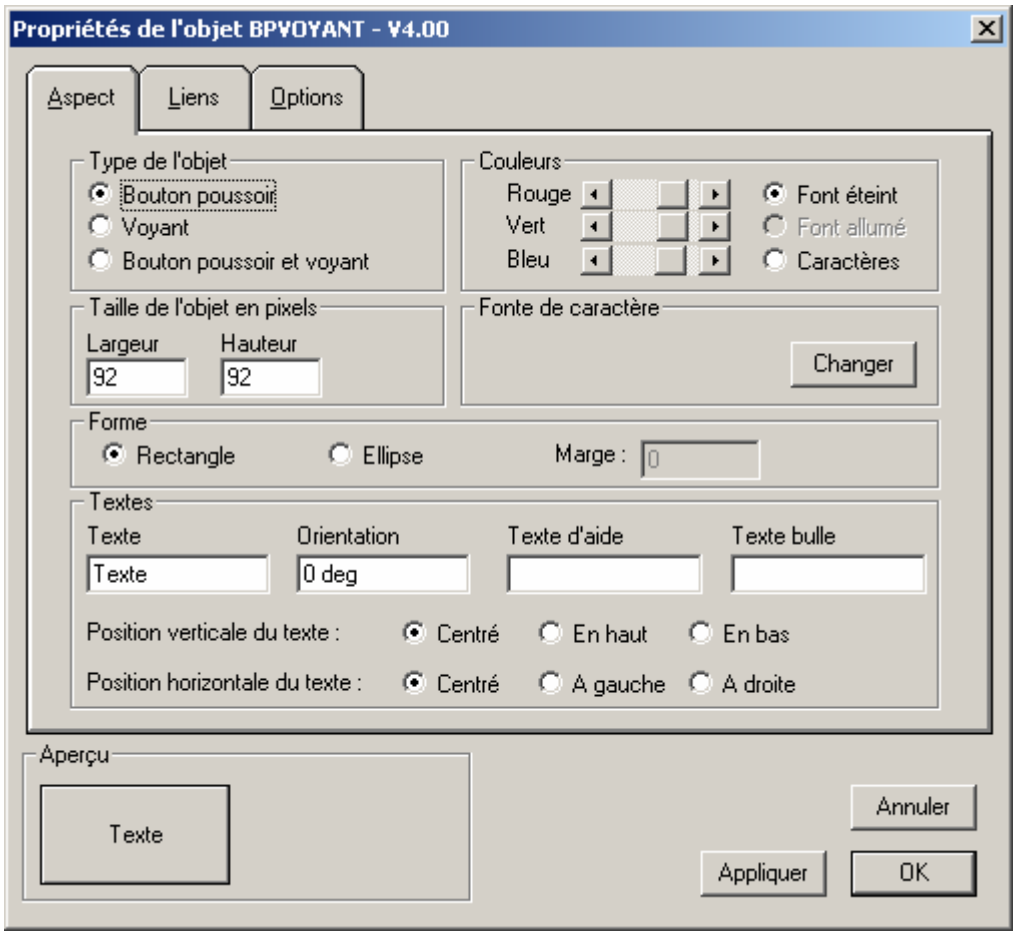
Pour ajouter un bouton poussoir sur le pupitre vous devez maintenant cliquer sur l'icône  du pupitre (accès au menu) et sélectionner l'option « Ajouter un objet ». Dans la catégorie « Objets de base », double cliquez sur « Bouton voyant ».
L'objet doit alors apparaître sur le pupitre :



Etape 5

Nous allons associer le bouton poussoir à une sortie de l'application d'automatisme, %Q4 par exemple. Cliquez sur l'icône  du bouton

poussoir (pas du pupitre). La boîte de dialogue des propriétés du bouton poussoir s'ouvre :



Cliquez sur l'onglet « Liens » (en haut de la fenêtre de dialogue). Sous la rubrique « Action lorsque le bouton est enfoncé » entrez « %Q4=1 ». Sous la rubrique « Action lorsque le bouton est relâché » entrez « %Q4=0 ». Cliquez ensuite sur le bouton poussoir « OK » en bas de la fenêtre de dialogue. Les actions sur le bouton poussoir ont pour effet de piloter la sortie 4 de l'application d'automatisme. Vous pouvez ouvrir une fenêtre « Monitoring » dans le menu « Mise au point » en cliquant avec le bouton droit de la souris dans le navigateur. Visualisez l'état de la sortie 4 pendant que vous cliquez puis relâchez le bouton poussoir.

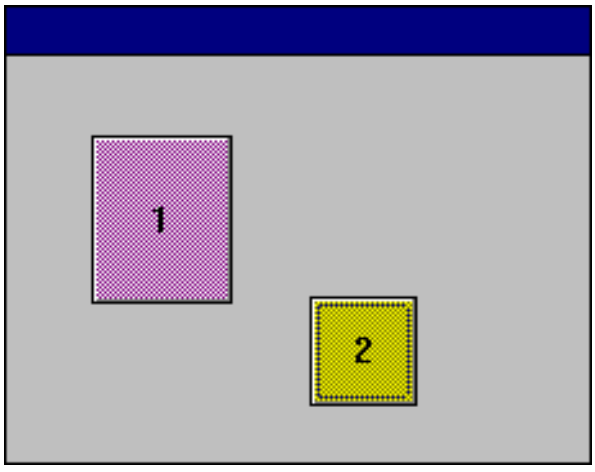
Etape 6

Nous allons associer un voyant à l'objet « Bouton Voyant », ce voyant sera associé à une entrée de l'application d'automatisme (I2 par exemple). Cliquez à nouveau sur l'icône  du bouton poussoir. Dans l'onglet « Aspect » cliquez sur le bouton radio « Bouton poussoir et voyant ». Cliquez sur l'onglet « Liens » et entrez « %i2 » sous la rubrique « Etat du voyant ». Cliquez sur le bouton poussoir « OK » en bas de la

fenêtre de dialogue des propriétés. Vous pouvez maintenant modifier l'état de la variable « %i2 » (avec une fenêtre « Monitoring » ou en modifiant l'état de l'entrée physique si elle existe).

Etape 7

Nous allons dupliquer l'objet « Bouton Voyant » présent sur le pupitre pour obtenir un deuxième bouton poussoir dont nous modifierons les propriétés. Cliquez sur le bouton poussoir avec le bouton gauche de la souris en maintenant la touche [SHIFT] enfoncée. Des carrés noirs apparaissent autour de l'objet sélectionné. Cliquez sur l'icône ... du pupitre et choisissez l'option « Copier ». Cliquez sur l'icône ... du pupitre et choisissez l'option « Coller ». Il y a maintenant deux objets « Bouton Voyant » superposés. Saisissez celui du dessus (c'est le seul accessible) par sa barre supérieure et déplacez le en dehors de l'autre bouton poussoir. L'objet ainsi dupliqué possède les mêmes propriétés que le premier. Vous pouvez maintenant paramétrer le deuxième objet pour qu'il soit par exemple lié à la sortie 5 et l'entrée 3.
Vous pouvez également personnaliser l'aspect des boutons poussoirs en utilisant l'onglet aspect pour les deux objets. Vous pouvez aussi modifier la taille des objets en les saisissant par un de leurs bords.
Les trois objets présents à l'écran (le pupitre et les deux boutons poussoirs) sont en mode « Configuration », c'est à dire qu'ils laissent apparaître une mini barre en haut de leur surface, des icônes et une bordure permettant de les dimensionner. Les objets ont un autre mode nommé « Exploitation » dans lequel ils ont un aspect définitif : plus de barre en haut, plus d'icône et plus de bordure pour les dimensionner. Pour basculer un objet d'un mode à l'autre, il suffit de cliquer dessus avec le bouton droit de la souris.
A ce stade vous devez avoir créé un objet qui ressemble à ceci :



Réaliser une application de supervision autonome

Pour réaliser une application de supervision autonome (sans développer l'application d'automatisme avec AUTOMGEN), suivez la démarche suivante :

- créez les correspondances de variables entre AUTOMGEN et les variables de l'automate en double cliquant sur l'élément « Configuration / Post-processeur / <nom du post-processeur> / Correspondance des variables » (reportez-vous au manuel de référence des post-processeurs pour plus de détails),
- compilez l'application en cliquant sur le bouton  de la barre d'outils (ceci a pour effet de valider la correspondance des variables),
- configurez le mode de connexion sur « Seulement connecter » en double cliquant sur l'élément « Configuration / Post-processeur / <nom du post-processeur> / Option de connexion ».

Remarques :

- l'option « Go automatique » du projet vous permet d'obtenir une application qui se connecte automatiquement à la cible au démarrage.
- la commande « Générer un exécutable » du menu « Fichier » vous permet d'obtenir une application de supervision auto compactée et libre de droit sous la forme d'un seul fichier exécutable.

Syntaxe pour l'accès à l'état des variables

Vous pouvez utiliser les noms de variables en syntaxe AUTOMGEN, CEI 1131-3 ou un symbole. Les boutons poussoirs « … » se trouvant à proximité des zones de saisies dans les objets permettent d'accéder à l'assistant de sélection d'un nom de variable.

Etat booléen

Cette syntaxe est à utiliser dans les rubriques « états » des objets.

Pour tester l'état d'une variable booléenne, utilisez le nom de la variable, par exemple : « i0 », « %q0 », « vanne ouverte ».

Pour tester l'état complémenté d'une variable booléenne, ajouté un caractère « / » avant le nom de la variable, par exemple : « /i4 », « /%M100 », « /niveau haut ».

Pour tester l'égalité d'une variable numérique avec une constate, utilisez le nom de la variable numérique, suivi de « = », « < », « > » et d'une constante, par exemple : « %MW200=4 », « vitesse>2 ».

L'état complémenté permet de réaliser les tests « si différent », « si inférieur ou égal » et « si supérieur ou égal », par exemple : « /%MW201<300 ».

L'opérateur '&' permet de tester un bit d'une variable numérique, par exemple M200&4 teste le troisième bit (4 = 2 puissance 3) du mot m200.

Etat numérique

Cette syntaxe est à utiliser dans les rubriques « états » des objets.

Pour lire l'état d'une variable numérique, utilisez le nom de la variable, par exemple : « %MW300 », « m400 », « pression », « _+V_ ».

Modification d'état

Cette syntaxe est à utiliser dans les rubriques « ordre » des objets.

Pour réaliser la modification de l'état d'une variable, ajoutez à la suite du nom de la variable le signe « = » suivi d'une constante.

Pour les variables booléennes, les constantes suivantes sont utilisables : « 0 », « 1 », « F1 » (forçage à 1), « F0 » (forçage à 0), « UF » (fin de forçage), par exemple : « %Q0=1 », « %I10=F1 », « %I12=UF ».

Pour les variables numériques, la constante est un nombre, par exemple : « M200=1234 », « vitesse=4 ».

Ordres spéciaux

Les mots clés suivants peuvent être utilisés dans les rubriques ordres des objets :

« RUN » : passe la cible en mode RUN,

« STOP » : passe la cible en stop,

« INIT » : initialise la cible,

« STEP » : effectue un pas sur la cible,

« GO » : identique à la commande GO de l'environnement,

« ENDGO » : termine la commande GO,

« EXIT » : quitte l'environnement,

« UCEXIT » : quitte l'environnement sans demander de confirmation,

« OPENAOF(<objet>) » : montre un objet. « <objet> » désigne un objet soit par le titre de l'objet soit par son numéro d'identificateur (configuré dans les propriétés des objets) avec la syntaxe « #identificateur ».

« CHAINAOF(<objet>) » : montre un objet et cache l'objet courant. « <objet> » désigne un objet soit par le titre de l'objet soit par son numéro d'identificateur (configuré dans les propriétés des objets) avec la syntaxe « #identificateur ».

Echanges entre objets

« PARENTPARAM(paramètre {+n} {-n}) »
Permet à un objet enfant d'accéder à un paramètre du pupitre parent. Le paramètre doit être défini dans la rubrique « Liens / Données pour les objets enfants » du pupitre parent. Voir chapitre L'objet « Pupitre »
SISTERPARAM(identificateur , paramètre)
Utilisé par l'objet OBJET, cette syntaxe permet de lire une valeur propre à un objet. Voir l'objet « Objet ».
SETPARAM(identificateur , paramètre , valeur)
Permet de modifier le paramètre d'un objet.
Pour accéder à la liste des paramètres modifiables, cliquez sur le bouton droit de la souris pendant l'édition des zones actions d'un objet « Bouton Voyant », puis choisissez la commande « Paramètres ».

Détail de l'objet « Pupitre »

Onglet « Aspect »

Fenêtre

Permet de définir l'aspect de la fenêtre composant le pupitre : présence d'une bordure, d'une barre de titre (dans ce cas un titre peut être précisé), présence des icônes de fermeture et de réduction. La case à

cocher « Affichage des messages d'aides » permet de définir une zone de message en bas de la fenêtre, la taille de cette zone est déterminée automatiquement en fonction de la fonte choisie (voir ci-après). Si une telle zone n'est pas définie, alors les messages provenant des enfants seront affichés soit sur le pupitre parent du pupitre, soit en bas de la fenêtre de l'environnement AUTOMGEN (si l'objet n'a pas de parent).

Fond du pupitre

Détermine l'aspect du fond du pupitre : coloré (voir ci-après), transparent (accessible seulement si le pupitre est l'enfant d'un autre pupitre), bitmap (le fond est défini par un fichier « .BMP » créé avec PAINTBRUSH par exemple).

Couleurs

Permet de choisir la couleur pour le fond du pupitre (si un fond coloré est sélectionné - voir plus haut), le fond et les caractères de la zone d'affichage des messages d'aide (si cette zone est validée - voir plus haut).

Fonte de caractères pour l'aide

Détermine la fonte utilisée pour afficher les messages d'aide en bas du pupitre.

Taille de l'objet

Détermine les dimensions de l'objet en nombre de points. La modification de ces valeurs permet de régler très précisément la taille de l'objet.

Textes

Le texte d'aide et le texte bulle.

Onglet « Bitmap »

Bitmap

Si le fond du pupitre contient un bitmap (voir onglet « Aspect ») la zone d'édition doit contenir un nom d'accès complet vers un fichier « .BMP » (les formats 16 couleurs, 256 couleurs et 24 bits sont supportés).
Les boutons poussoirs « PARCOURIR » et « EDITER » permettent respectivement de rechercher un fichier « .BMP » et d'éditer un fichier avec le logiciel PAINTBRUSH de WINDOWS.

Onglet « Liens »

Données pour les objets enfants

Cette zone d'édition permet de définir des paramètres auxquels pourront accéder les objets enfants avec le mot clé « PARENTPARAM ». Il faut écrire une définition par ligne. Chaque définition doit respecter la syntaxe suivante : « PARAMETRE=VALEUR ».

Onglet « Options »

Grille

Permet de définir une grille (invisible) pour le positionnement des objets. Seule la commande « Déplacer » du menu intégré au pupitre utilise la grille. Les valeurs pour la grille sont exprimées en nombre de pixels. Les valeurs 0 et 1 annulent l'effet de la grille. Cette fonction doit être utilisée pour aligner parfaitement les objets.

Rafraîchissement des enfants

La case à cocher « Continuer à rafraîchir les enfants ... » détermine si les enfants doivent continuer à être mis à jour lorsque le pupitre est mis en icône. Cette option permet, lorsqu'elle n'est pas sélectionnée, d'augmenter les performances du système lorsqu'un pupitre mis en icône ne contient que des éléments visuels.

Onglet « Enfants »

Enfants

Cette rubrique contient la liste des objets enfants du pupitre. Le bouton poussoir « Propriétés » permet d'ouvrir directement la boîte de dialogue des propriétés de l'enfant sélectionné dans la liste. Le bouton poussoir « Détruire » élimine l'objet sélectionné. Les zones d'édition « Positions » permettent de régler la position des objets.

Onglet « Externe »

Nom de l'exécutable

Nom d'un fichier exécutable fonctionnant dans le pupitre.

Paramètres

Paramètres fournis sur la ligne de commande à l'exécutable.

Détail de l'objet « Bouton Voyant »

Onglet « Aspect »

Type de l'objet

Permet de choisir la nature de l'objet : bouton poussoir, voyant ou bouton poussoir intégrant un voyant.

Couleurs

Permet de choisir les couleurs de l'objet. Si l'objet est un bouton poussoir, le réglage « Fond éteint » représente la couleur du bouton poussoir. Si l'objet est un voyant ou un bouton poussoir intégrant un voyant le réglage « Fond allumé » détermine la couleur du fond lorsque le voyant est allumé et « Fond éteint » la couleur lorsque le voyant est éteint. Seule la couleur des caractères est réglable si l'aspect de l'objet est déterminé par un bitmap.

Taille de l'objet

Détermine les dimensions de l'objet en nombre de points. La modification de ces valeurs permet de régler très précisément la taille de l'objet. Ceci est nécessaire si l'aspect de l'objet est déterminé par un bitmap.

Fonte de caractères

Permet de choisir la fonte et la taille des caractères. Le fichier de fonte utilisé devra être présent sur le PC où s'exécute le programme.

Texte

Permet de spécifier le texte affiché sur l'objet, sa position, son orientation ainsi qu'un texte d'aide qui apparaît lorsque le bouton est pressé et un texte bulle qui apparaît lorsque le curseur est placé sur l'objet.

Onglet « Liens »

Action lorsque ...

Permet de définir les actions à réaliser lorsque le bouton est enfoncé et lorsqu'il est relâché.
Une action peut être le forçage de l'état d'une variable, par exemple :

```
O0=1, m200=4, _depart cycle_=3
```

Ou un mot clé prédéfini.
Exemple de configuration pour que l'entrée i10 reflète l'état du bouton poussoir (i10 à 0 si le bouton est relâché, i10 à 1 si le bouton est enfoncé) :

Action lorsque le bouton est enfoncé : `i10=1`
Action lorsque le bouton est relâché : `i10=0`

Etat du voyant
Détermine l'état du voyant. Cette rubrique doit contenir le nom d'une variable qui pilotera le voyant : 0 = voyant éteint, 1 = voyant allumé.

Par exemple :

```
b31, o4, _voyant init_, m200=50, m400<8, m500&16
```

Identificateur
Permet de référencer l'objet par rapport aux autres objets.

Condition de désactivation
Permet de désactiver le voyant. Si cette rubrique contient le nom d'une variable, alors cette variable désactive l'objet si elle est vraie.

Onglet « Options »

Type du bouton poussoir
Détermine si le bouton poussoir est bistable (il reste enfoncé), monostable ou une combinaison des deux : monostable sur simple clic et bistable sur double clic.

Clavier
Permet d'associer une touche au bouton poussoir. Si cette touche ou cette combinaison de touche est présente au clavier alors le bouton poussoir sera enfoncé.

Pour préciser le code de la touche différentes syntaxes sont utilisables :
- un simple caractère : par exemple A, Z, 2,
- le caractère $ suivi du code de la touche en hexadécimal,
- le nom d'une touche de fonction, par exemple F5.

Pour les combinaisons de touches il faut ajouter au début CTRL+ ou SHIFT+.
Par exemple : CTRL+F4 ou SHIFT+Z.

Bitmap
Permet de spécifier un fichier bitmap qui contiendra le dessin de l'objet.

L'option « Redimensionner l'image » permet d'étendre le bitmap sur toute la surface de l'objet.

Le fichier bitmap contient les quatre aspects possibles de l'objet : bouton relâché voyant éteint, bouton enfoncé voyant éteint, bouton relâché voyant allumé et bouton enfoncé voyant allumé.

Même si le fichier est un bouton poussoir sans voyant ou un voyant il y a toujours quatre aspects dont seuls deux sont utilisés.
Le fichier bitmap est découpé en quatre horizontalement.
Exemple :

L'option « Aspect différent si le curseur est sur l'objet … » permet d'afficher une image différente lorsque l'objet passe sur l'objet.

Si cette option est cochée, le fichier bitmap contient 8 aspects, 4 aspects supplémentaires sont ajoutées à droite du bitmap pour contenir le dessin de l'objet lorsque le curseur est sur l'objet.

Exemple :

Sons

En choisissant des fichiers .WAV, l'objet peut produire des sons si l'objet est enfoncé, relâché, ou si le curseur passe sur l'objet.

Détail de l'objet « Valeur digitale »

Onglet « Aspect »

Format

Permet de définir le type d'affichage :

- Toujours montrer le signe : affiche le signe '+' pour les valeurs signées positives,
- Valeur signée : définit le mode signé ou non signé pour les entiers 16 ou 32 bits (base 10 uniquement),
- Afficher tous les digits : affiche des 0 au début de la valeur si nécessaire.

Base

- Détermine la base d'affichage pour les entiers 16 et 32 bits.

Couleurs

Permet de choisir les couleurs du fond de l'objet (s'il n'est pas transparent) et des caractères.

Fonte de caractères

Permet de choisir la fonte et la taille des caractères. Le fichier de fonte utilisé devra être présent sur le PC où s'exécute le programme.

Nombre de digits
Définit la longueur de la partie entière et décimale.

Fond
Permet de choisir entre un fond coloré ou transparent (si l'objet est posé sur un pupitre seulement).

Taille de l'objet
Détermine les dimensions de l'objet en nombre de points. La modification de ces valeurs permet de régler très précisément la taille de l'objet.

Onglet « Textes »

Texte Bulle
Texte affiché dans une bulle lorsque l'utilisateur déplace le curseur sur l'objet.

Texte affiché avant et après la valeur
Permet d'afficher des informations à gauche et à droite de la valeur numérique.

Onglet « Liens »

Variable ou symbole
Désigne la variable à afficher. Pour accéder au compteur ou à la consigne d'une temporisation il faut utiliser la syntaxe suivante :
- pour le compteur : COUNT(temporisation), exemple : COUNT(t3),
- pour la consigne : PRED(temporisation), exemple : PRED(t7),

L'état de la variable est modifiable ...
Si cette case est cochée alors l'utilisateur peut modifier l'état de la variable en cliquant sur l'objet.

Détail de l'objet « Valeur analogique »

Onglet « Aspect »

Objets
Permet de définir le type d'affichage.

Orientation

Détermine l'orientation : horizontale ou verticale.

Couleurs

Permet de choisir les couleurs du fond de l'objet (s'il n'est pas transparent) et de l'objet.

Fond

Permet de choisir entre un fond coloré ou transparent (si l'objet est posé sur un pupitre seulement).

Taille de l'objet

Détermine les dimensions de l'objet en nombre de points. La modification de ces valeurs permet de régler très précisément la taille de l'objet.

Textes

Le texte bulle.

Onglet « Liens »

Variable ou symbole

Désigne la variable liée à l'objet (un mot ou un compteur).

Les actions de l'utilisateur ...

Détermine si l'état de la variable peut être modifié par l'utilisateur.

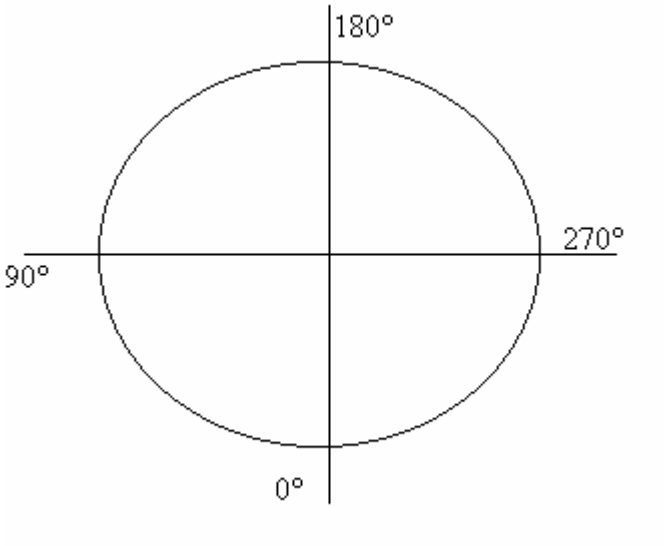
Onglet « Bornes »

Minimum, maximum

Valeurs minimales et maximales.

Angle de départ, angle de fin

Pour l'affichage de type cadran détermine l'angle de départ et l'angle d'arrivée. Les valeurs sont précisées en degrés :



Onglet « Graduations »

Utiliser les graduations

Valide ou invalide l'utilisation des graduations.

Valeur de départ, valeur de fin

Valeurs affichées pour les graduations, ces valeurs peuvent être des nombres signés et/ou des nombres à virgule.

Pas des petites graduations, pas des grandes graduations

Pas des graduations (deux niveaux) par rapport aux valeurs de départ et de fin. Ces valeurs peuvent être des nombres à virgule.

Police de caractère

Définit les caractères utilisés pour les graduations.

Zone N°1, zone N°2 et zone N°3

Permet de définir des zones colorées. « Valeur de départ » et « Valeur de fin » définissent chaque zone. La couleur pour chaque zone est spécifiée par trois composantes le rouge, le vert et le bleu comprises entre 0 et 255.

Couleurs

Détermine la couleur des caractères et des graduations. Les couleurs sont ici aussi exprimées par leurs trois composantes : le rouge, le vert et le bleu.

Détail de l'objet « Ecran, clavier, liste de messages »

Liens avec l'application

Le lien entre l'objet et l'application se fait par des tables de mots.

Pour envoyer des données au type d'objet (avec ou sans clavier) il faut placer les données à partir du deuxième mot de la table de réception puis la longueur des données dans le premier mot de la table (la longueur maximale est 255). Chaque mot contient une donnée.

Les données peuvent être : un caractère ASCII, un numéro de message prédéfini + 8000 hexa, ou une commande spéciale : 100 hexa efface la fenêtre, 200 hexa affiche la date, 300 hexa affiche l'heure, 400 affiche le numéro du message.

Lorsque l'objet a relu les données disponibles dans une table il remet le premier mot à 0 pour signaler que l'opération a été effectuée.

Pour les objets « avec clavier » le principe est le même : le premier mot de la table d'émission contient le nombre de caractères entrés au clavier, les mots suivants contiennent les caractères (un par mot). L'application doit remettre à 0 le premier mot lorsqu'elle a utilisé les données.

Pour l'objet « Boîte à messages, liste d'alarmes » la table d'échange àaune longueur fixe de 10 mots. Comme pour le type « Ecran » le premier mot déclenche l'affichage de message. S'il est différent de 0 il désigne un numéro de message à afficher. Seuls des messages enregistrés peuvent être affichés. Le premier mot peut également prendre la valeur ffff hexa pour vider la boîte à messages.

Description des 10 mots utilisés pour les échanges avec le type « Boîte à messages » :

Mot 0 représente le premier mot de la table, Mot 1 le deuxième, etc ...
Mot 0 : numéro de message à afficher ou 0 si pas de message ou ffff hexa pour tout effacer,
Mot 1 : numéro de classe pour le message (voir chapitre Les classes de messages pour plus d'explications),

Les mots qui suivent déterminent la date et l'heure qui peuvent être affichées pour chaque message. Une valeur égale à ffff hexa demande à l'objet d'utiliser la date et l'heure courante de l'ordinateur (cela ne concerne pas les millisecondes).
Mot 2 : jour
Mot 3 : mois
Mot 4 : année
Mot 5 : heures
Mot 6 : minutes
Mot 7 : secondes
Mot 8 : millisecondes
Mot 9 : réservé (mettre 0)

Les classes de messages

Les classes de messages permettent de classifier les messages en famille qui partagerons les caractéristiques suivantes : la couleur du fond, la couleur des caractères et une icône.
Il existe deux classes prédéfinies :

- la classe message d'information : caractères bleus sur fond blanc, icône  , elle porte le numéro -1,
- la classe message d'alarme : caractères blancs sur fond rouge, icône  , elle porte le numéro -2.

D'autres classes peuvent être définies par l'utilisateur.
Un texte bulle peut être associé à l'objet.

Onglet « Aspect »

Type de l'objet

Permet de définir un des types d'objet. Voir le chapitre Liens avec l'application.

Couleurs

Permet de choisir les couleurs du fond de l'objet et des caractères.

Fonte

Permet de choisir la fonte de caractères utilisée pour afficher les textes.

Taille de l'objet

Détermine les dimensions de l'objet en nombre de points. La modification de ces valeurs permet de régler très précisément la taille de l'objet.

Textes

Le texte bulle.

Onglet « Liens »

Réception, émission

Définissent les premières variables de la table de réception et d'émission. Ces zones peuvent contenir un nom de variable ou un symbole.

Onglet « Liste »

Ces rubriques ne concernent que les objets de type « Boîte à messages ».

Icônes

Cette case à cocher détermine s'il faut afficher une icône avant les messages.

Classes

Cette case à cocher détermine s'il faut afficher le numéro de la classe du message.

Jours, Mois, Années, Heures, Minutes, Secondes, 1/1000 secondes

Ces cases à cocher déterminent s'il faut afficher chacun de ces éléments.

Messages

Cette case à cocher détermine s'il faut afficher un message.

Numéros

Cette case à cocher détermine s'il faut afficher le numéro d'apparition des messages.

Classes de messages

Cette zone d'édition permet de définir de nouvelles classes de messages. Chaque ligne définit une classe. Sur chaque ligne on doit trouver dans l'ordre et séparés par des virgules : la couleur du fond, (trois composantes rouge, vert et bleu) la couleur des caractères (trois composantes rouge, vert et bleu), le nom de la classe, le nom d'un fichier bitmap pour l'icône associé à la classe.
Par exemple :
255,0,0,0,0,0,ALARME,alarme.bmp
Signifie :

Couleur du fond rouge, couleur des caractères noir, nom de la classe ALARME, fichier contenant l'icône : « alarme.bmp ».

Onglet « Options »

Afficher les codes hexadécimaux des caractères

Cette option permet d'afficher le code hexadécimal pour chaque caractère à la place de sa représentation ASCII. Elle est utilisable pour les objets de type « Ecran ... » et permet notamment la mise au point des programmes.

Barre de défilement horizontale, verticale

Montre ou cache les barres de défilements.

Convertir les caractères OEM -> ANSI

Si cette case est cochée, alors les caractères provenant de l'application d'automatisme sont automatiquement convertis du jeu de caractères OEM (MS-DOS) au jeu de caractères ANSI (WINDOWS). La conversion inverse est appliquée aux caractères allant de l'objet à l'application d'automatisme.

Dupliquer les messages vers ...

Cette rubrique peut recevoir un nom de fichier ou de périphérique (« LPT1 » par exemple pour l'imprimante). Il est possible de spécifier plusieurs fichiers et/ou périphériques en les séparant par une virgule. Les affichages seront automatiquement dupliqués : impression dite « au fil de l'eau ».

Associer un fichier de stockage des messages ...

Permet de définir un fichier qui sera associé à l'objet et utilisé pour le stockage des informations. Si un tel fichier existe alors les messages sont sauvegardés (à concurrence du nombre défini sous la rubrique « nombre de lignes mémorisées », si ce nombre est atteint les données les plus anciennes sont effacées). Lorsque l'objet est ouvert, et si un fichier de stockage existait lors de sa dernière utilisation, alors les données contenues dans le fichier sont transférées dans l'objet.

Ecrire les messages périmés vers ...

Permet de définir un fichier ou un périphérique qui recevra les messages périmés (les messages qui sont éliminés du fichier de stockage pour libérer de la place).

Nombre de lignes mémorisées ...

Détermine la capacité du fichier de stockage des messages en nombre de lignes. La valeur 0 attribue le maximum d'espace mémoire utilisable (pas de limite fixe).

Onglet « Messages »

Messages prédéfinis

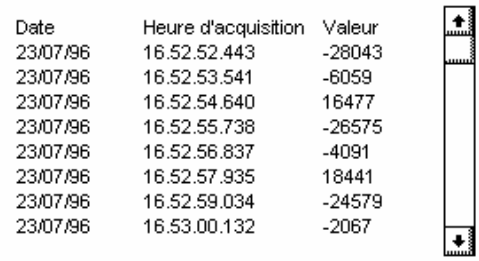
Cette boîte d'édition permet de documenter les messages prédéfinis (un par ligne).

Détail de l'objet « Archivage de données »

Onglet « Aspect »

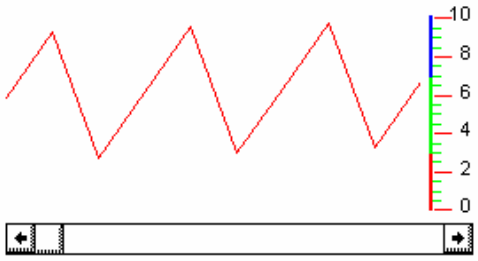
Objets

Permet de définir le type d'affichage.
L'objet peut être représenté sous la forme d'une table (*figure 1.1*) ou sous la forme de tracés de courbes (*figure 1.2*).



Date	Heure d'acquisition	Valeur
23/07/96	16.52.52.443	-28043
23/07/96	16.52.53.541	-6059
23/07/96	16.52.54.640	16477
23/07/96	16.52.55.738	-26575
23/07/96	16.52.56.837	-4091
23/07/96	16.52.57.935	18441
23/07/96	16.52.59.034	-24579
23/07/96	16.53.00.132	-2067

(*figure 1.1*)



(*figure 1.2*)

Couleurs

Permet de choisir la couleur de la police de caractère lorsque l'objet est sous la forme d'une table ainsi que la couleur de marquage des valeurs sur les courbes.

Taille de l'objet

Détermine les dimensions de l'objet en nombre de points. La modification de ces valeurs permet de régler très précisément la taille de l'objet.

Texte

Le texte bulle associé à l'objet.

Onglet « Données »

Première variable à lire

Permet de sélectionner la première variable qu'il faudra archiver.

Nombre de variables à lire

Signale à l'objet ARCHIVE le nombre de variables consécutives à la « Première variable à lire » qu'il faudra archiver .

Nombre d'enregistrements mémorisés

Permet de dimensionner la base de données en mémoire .
Un enregistrement représente une acquisition de « n » variables (« n » étant le nombre de variables à lire).

Lecture périodique

L'acquisition des variables se fera à intervalle fixe dés l'exécution de l'objet ARCHIVE.

Lecture déclenchée

Une acquisition des variables sera effectuée dés que le « Mot de contrôle » en aura donné l'ordre .

Période

Permet de fixer le temps qui sépare deux acquisitions. Le temps est au format Jour(s) /Heure(s) /Minute(s) /Seconde(s) /Milliseconde(s) :
J pour les jours
H pour les heures
M pour les minutes
S pour les secondes
MS pour les millisecondes

Ex : 2J
Ex : 2H10M15S

Contrôle

Permet de définir une variable (un mot) qui contrôlera l'objet ARCHIVE. Dés la valeur prise en compte, son contenu est forcé à 0 par l'objet ARCHIVE.

Valeur	**Action**
0	Rien
1	Déclenchement d'une acquisition (Lecture déclenchée)

2	Geler les acquisitions
3	Reprendre l'archivage (après avoir gelé)
4	Vider la base de données en mémoire
5	Détruire le fichier d'archivage
6	Activer le mode « Suivi des dernières acquisitions »
7	Annuler le mode « Suivi des dernières acquisitions »

Onglet « Options »

Utiliser le fichier image

Le fichier image est utilisé :
En fin d'utilisation de l'objet ARCHIVE, pour sauvegarder la base de données présente en mémoire.
Au lancement de l'objet ARCHIVE, pour reconstituer la base de données qui était présente en mémoire lors de la dernière utilisation.

Utiliser le fichier archive

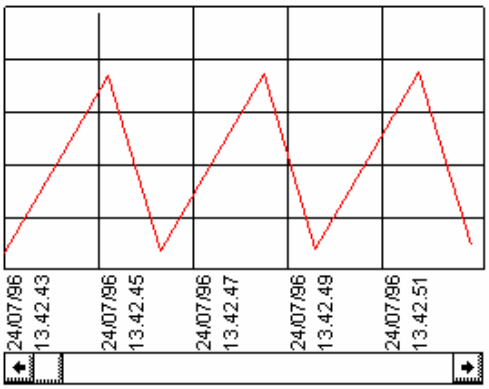
Chaque acquisition est enregistrée dans ce fichier au format base de données standard.

Affichage

Date d'acquisition: Permet d'afficher la date d'acquisition d'un enregistrement.
Heure d'acquisition: Permet d'afficher l'heure d'acquisition d'un enregistrement.
Heures, minutes, secondes, millisecondes: Permet de configurer l'affichage de l'heure d'acquisition.
L'affichage du temps se fait en aval de l'affichage des acquisitions pour l'objet TABLE (*figure 3.1*) ou sous la grille lorsqu'elle est définie pour l'objet COURBE (*figure 3.2*)

Date	Heure d'acquisition	Valeur
24/07/96	13.42.43.112	3141
24/07/96	13.42.44.211	25217
24/07/96	13.42.45.309	-18451
24/07/96	13.42.46.408	3489
24/07/96	13.42.47.506	25525
24/07/96	13.42.48.605	-17931
24/07/96	13.42.49.703	4085
24/07/96	13.42.50.802	26149
24/07/96	13.42.51.900	-17375
24/07/96	13.42.52.999	4709

(figure 3.1)



(figure 3.2)

Onglet « Tables »

Police de caractères

Permet de sélectionner une police de caractères pour l'affichage du nom des colonnes, du temps et la valeur des acquisitions.

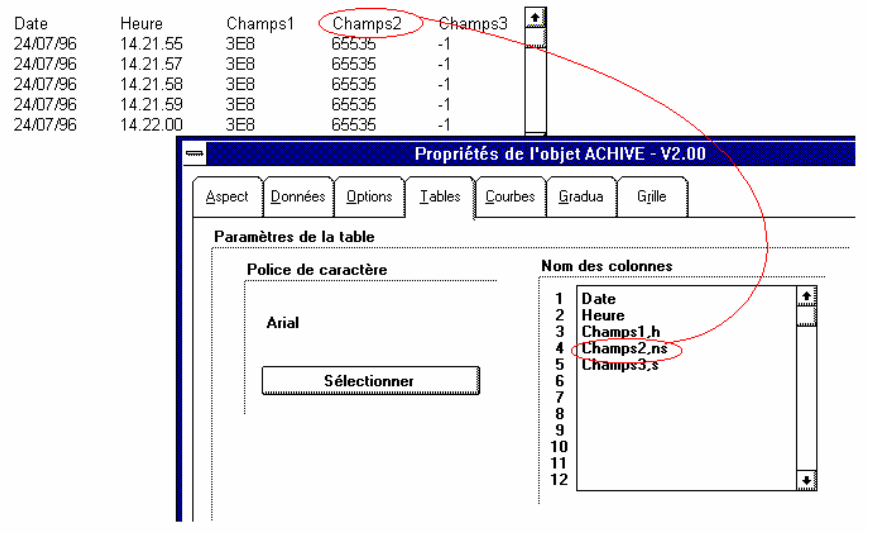
Nom des colonnes

Permet de définir le nom des colonnes pour l'objet TABLE ainsi que le format d'affichage de ces colonnes (*figure 4.1*).
syntaxe :
nom, format

format *	**Affichage**
pas de format spécifié	Signé, décimal, visible
h	Hexadécimal
d	Décimal
ns	Non signé
s	Signé
nv	Non visible
v	Visible

* On peut combiner les différentes options, par exemple :

Format	**Affichage**
d,ns,v	Décimal sans signe visible



(*figure 4.1*)

Onglet « Courbe »

Valeur minimale, maximale

Permet de sélectionner la valeur minimale et maximale pour l'affichage des courbes.
Seules les valeurs comprises entre la valeur minimale et la valeur maximale seront visibles sur l'écran.

Visualiser

Permet de définir le temps de visualisation. Celui-ci est communiqué a l'objet ARCHIVE suivant le format Jour(s) / Heure(s) / Minute(s) / Seconde(s) / Milliseconde(s) :
J pour les jours
H pour les heures
M pour les minutes
S pour les secondes
MS pour mes millisecondes
Ex : **Visualiser** 2H30M10S
Ex : **Visualiser** 100MS

Marquage des valeurs sur la courbe

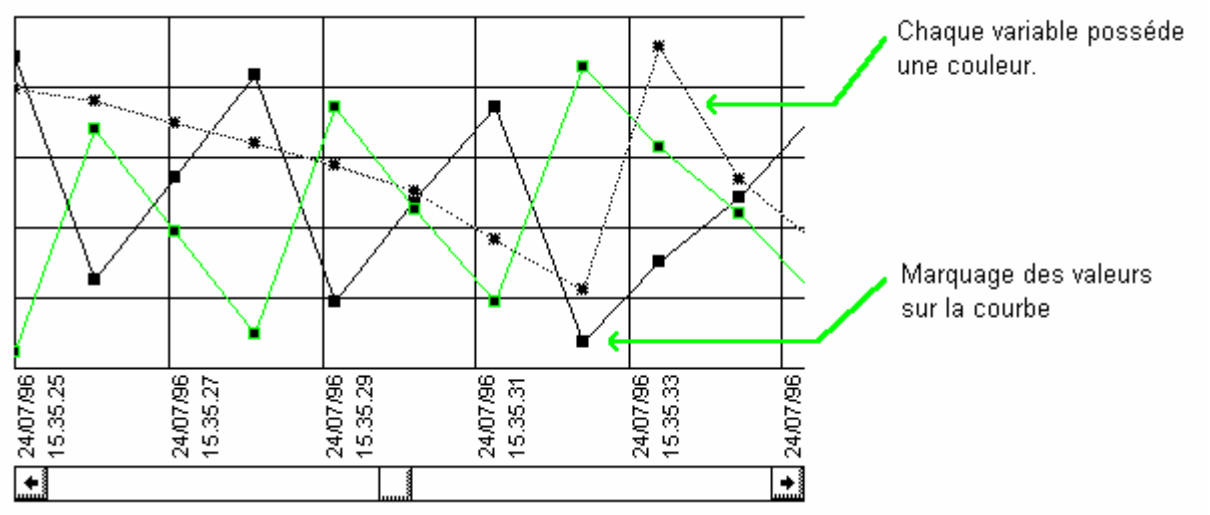
Permet de faire une marque sur la courbe pour chaque acquisition (*figure 5.1)*.

Affichage du temps

Permet d'afficher la date et l'heure d'acquisition d'une ou plusieurs variables sous la grille si elle est active. On peut définir la couleur et la police de caractères de l'affichage du temps.

Couleurs des tracés

Permet de définir une couleur pour chaque courbe. La première courbe possède la couleur de la première ligne, la deuxième courbe possède la couleur de la deuxième ligne etc...
Les couleurs sont au format Rouge, Vert, Bleu.
Ex: 255,0,0 tracé rouge
Si une couleur n'est pas définie sur une ligne, la courbe correspondante à cette ligne ne sera pas tracée.



(*figure 5.1)*

Onglet « Graduations »

Utiliser les graduations

Valide ou invalide l'utilisation des graduations (*figure 6.1).*

Valeur de départ, valeur de fin

Valeurs affichées pour les graduations, ces valeurs peuvent être des nombres signés et/ou des nombres à virgule.

Pas des petites graduations, pas des grandes graduations

Pas des graduations (deux niveaux) par rapport aux valeurs de départ et de fin. Ces valeurs peuvent être des nombres à virgule.

Police de caractère

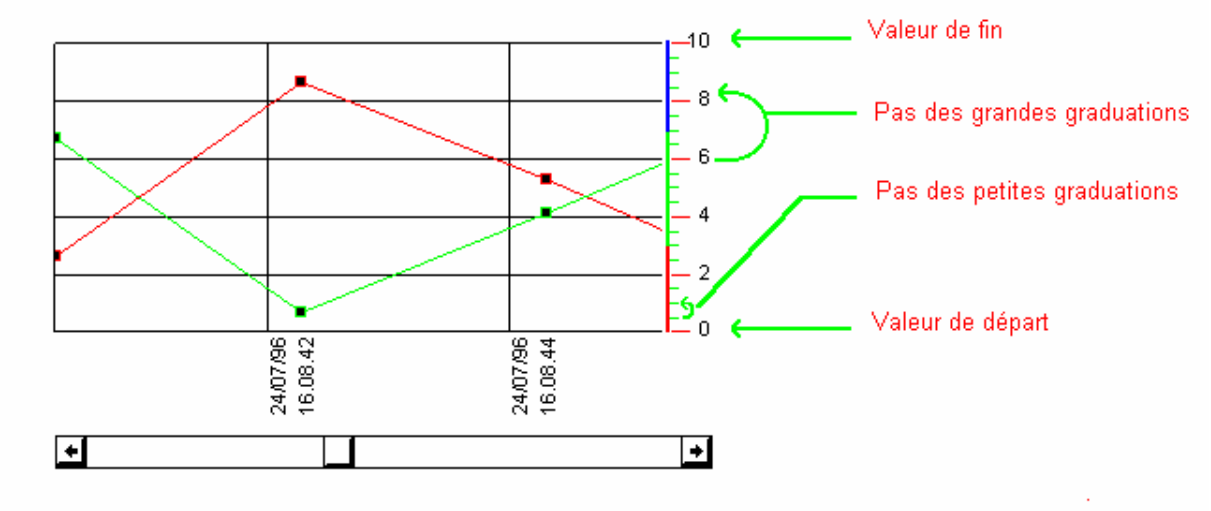
Définit les caractères utilisés pour les graduations.

Zone N°1, zone N°2 et zone N°3

Permet de définir des zones colorées. "Valeur de départ" et "Valeur de fin" définissent chaque zone. La couleur pour chaque zone est définie par trois composantes le rouge, le vert et le bleu comprises entre 0 et 255.

Couleurs

Détermine la couleur des caractères et des graduations. Les couleurs sont ici aussi exprimées par leur trois composantes : le rouge, le vert et le bleu.



(figure 6.1)

Onglet « Grille »

Afficher la grille

Valide ou invalide l'affichage de la grille.

Pas pour les ordonnées

Définit le pas vertical de la grille.

Pas pour les abscisses

Définit le pas horizontal de la grille. Le pas est au format Jour(s) / Heure(s) / Minute(s) / Seconde(s) / Milliseconde(s):
J pour les jours
H pour les heures
M pour les minutes
S pour les secondes
MS pour les millisecondes
Ex : 1J
Ex : 2H30M15S

Couleur

Permet de définir une couleur pour la grille.
La couleur est au format Rouge, Vert, Bleu
Ex : 255,0,0 Tracé rouge

Détail de l'objet « Objet »

Onglet « Aspect »

Type

Permet de définir un des types d'aspect de l'objet :

- « bitmap n aspects » : l'aspect de l'objet est donné par un fichier bitmap qui peut contenir plusieurs aspects voir le chapitre Onglet « Bitmap ».
- « bitmap n couleurs » : l'aspect de l'objet est donné par un fichier bitmap, la couleur est contrôlée par une variable de l'application d'automatisme qui remplace les pixels blancs du bitmap. Les autres pixels du bitmap doivent être noirs. La variable de l'application d'automatisme donne un numéro de couleur, les couleurs sont définies dans l'onglet « Couleurs ».
- « bitmap jauge » : l'objet est une jauge dont la forme est définie par un bitmap. Ce sont les pixels blancs du bitmap qui définissent la forme. Les autres pixels doivent être noirs. Le minimum, le maximum et l'orientation sont réglés dans l'onglet « Jauge ».
- « formes n couleurs » : un rectangle, un rectangle aux bords arrondis ou une ellipse. La couleur est gérée de la même façon que pour le type « bitmap n couleurs ».
- « formes jauges » : l'objet est une jauge rectangulaire. Le principe est le même que pour le type « bitmap jauge ».

Couleurs

Permet de choisir la couleur des caractères du texte affiché sur l'objet.

Fonte de caractères

Détermine la fonte utilisée pour afficher un texte sur l'objet.

Taille de l'objet

Détermine les dimensions de l'objet en nombre de points. La modification de ces valeurs permet de régler très précisément la taille de l'objet.

Textes

Le texte d'aide et le texte bulle.

Le texte affiché sur l'objet : la position et l'orientation peuvent être modifiées.

Onglet « Liens »

Objet cliqué, objet non cliqué

Définit les actions à réaliser respectivement lorsque l'utilisateur clique sur l'objet et lorsqu'il arrête de cliquer sur l'objet.
Une action peut être le forçage de l'état d'une variable, par exemple :
`O0=1, m200=4, _depart cycle_=3`
Ou un mot clé prédéfini.
Exemple de configuration pour que l'entrée i10 reflète l'état cliqué de l'objet (i10 à 0 si l'objet n'est pas cliqué, i10 à 1 si l'objet est cliqué) :
Objet cliqué : `i10=1`
Objet non cliqué : `i10=0`

Lier en permanence avec ...

Cette zone peut recevoir l'identificateur d'un objet frère. Si un tel objet existe alors la position de l'objet est calquée sur celui-ci. L'identificateur d'un objet est une valeur entière comprise entre 1 et 32767. Il est spécifié dans la zone d'édition « Identificateur » de la rubrique « Liens ».

Aspect / Couleur / Remplissage

Cette zone de la boîte de dialogue regroupe 8 zones d'édition qui permettent de définir différents types de comportement de l'objet par rapport aux variables de l'application d'automatisme.
Quel que soit le type de comportement il y aura toujours une position qui selon le type d'objet désignera :

- un aspect contenu dans un bitmap pour le type « bitmap n aspects »,
- un numéro de couleur pour les types « bitmap n couleurs » ou « forme n couleurs »,
- un remplissage pour les types « bitmap jauge » ou « forme jauge ».

La zone « Position » peut contenir un nom de variable numérique (C ou M). Les zones « + Position » et « - Position » peuvent contenir un nom de variables booléennes.
Deux types de fonctionnement sont possibles :

- si les zones « + Position » et « - Position » sont documentées alors les variables booléennes qu'elles contiennent pilotent la position : elles ajoutent ou enlèvent la valeur précisée dans la zone vitesse. Si la zone « Position » est documentée alors la position courante est écrite dans la variable dont elle contient le nom.

- si les zones « + Position » et « - Position » sont vides alors la valeur contenue dans la variable dont le nom est écrit dans la zone « Position » est lue comme position de l'objet.

La position peut varier entre les valeurs définies dans les zones « Mini » et « Maxi ».
Des capteurs peuvent êtres ajoutés (des noms de variables booléennes) qui seront vrais pour la position minimale et maximale (position égale au minimum ou au maximum).

Déplacement horizontal, déplacement vertical

Ces zones de la boîte de dialogue regroupent chacune 8 zones d'édition qui permettent respectivement de définir la position horizontale et verticale de l'objet. Le principe est identique à celui décrit ci-dessus.

Onglet « Formes »

Formes

Pour le type « Forme n couleurs » cette rubrique permet de sélectionner un rectangle, un rectangle avec des coins arrondis ou une ellipse.

Onglet « Bitmap »

Nom du fichier

Pour les types « Bitmap n aspects, bitmap n couleurs et bitmap jauge » cette zone d'édition doit contenir un nom d'accès complet vers un fichier « .BMP ». Ces fichiers peuvent être créés avec PAINTBRUSH ou tout autre éditeur graphique capable de créer des fichiers « .BMP ».
Les boutons poussoir « Parcourir » et « Editer » permettent respectivement de rechercher un fichier « .BMP » et d'éditer (lancement de PAINTBRUSH) le fichier « .BMP » dont le nom se trouve dans la zone d'édition.

Nombre d'aspects

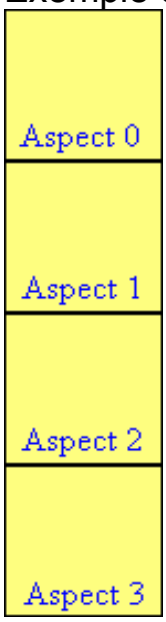
Cette zone d'édition doit contenir le nombre d'aspects (d'images) contenus dans un fichier « .BMP ». Cette option est utilisée pour le type « Bitmap n aspects ». Les différents aspects d'un objet doivent être dessinés les uns au dessous des autres. L'aspect le plus haut est le numéro 0.

Onglet « Wmf »

Nom du fichier

Pour les types « Métafichiers » cette zone d'édition doit contenir un nom d'accès complet vers un fichier « .WMF » ou « .EMF ».

Exemple de fichier « .BMP » à 4 aspects :



Le bitmap possède des zones transparentes ...

Cette option permet de créer un objet dont certaines parties seront transparentes (c'est le fond du pupitre père qui sera affiché). Les zones transparentes seront définies par des pixels possédant la même couleur, couleur qui sera déterminée par les trois composantes, rouge, verte et bleue. Pour régler ces composantes utilisez les trois barres de défilement. La couleur doit être réglée de façon très précise : exactement la même proportion de rouge, de vert et de bleu que la couleur des pixels des zones transparentes du bitmap.

Onglet « Couleurs »

Couleurs

Cette rubrique est utilisée pour les types « bitmap n couleurs » et « forme n couleurs ». Chaque ligne contient la définition pour une couleur. La syntaxe utilisée pour chaque ligne est :
proportion de rouge (entre 0 et 255), proportion de vert (entre 0 et 255) et proportion de bleu (entre 0 et 255). La première ligne désigne la couleur numéro 0, la deuxième la numéro 1, etc ...
Cette rubrique est aussi utilisée pour les types « bitmap jauge » et « forme jauge ». La première ligne (couleur 0) et la deuxième (couleur 1) déterminent les deux couleurs de la jauge (partie active et inactive).

Onglet « Jauge »

Jauge

Cette rubrique est utilisée pour les types « bitmap jauge » et « forme jauge ». Les zones « Valeur minimum » et « Valeur maximum » définissent les bornes pour la variable de pilotage de la jauge.

Orientation de la jauge

Détermine une des quatre directions possibles pour la jauge.

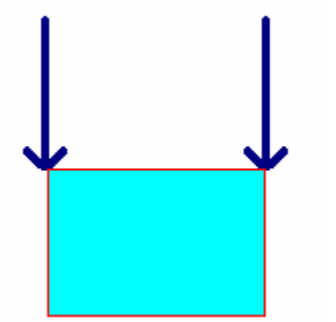
Onglet « Capteur »

L'objet OBJET peut être utilisé comme capteur. Le capteur est associé à une variable booléenne dont le résultat est vrai si le capteur est en contact avec une ou plusieurs couleurs prédéfinies (voir plus bas) ou faux autrement.

Position de la détection

Permet de déterminer le côté de l'objet qui doit détecter. La détection est réalisée sur les deux bords du côté choisi.

Exemple pour une détection au dessus :



Couleurs détectées

Un capteur est capable de détecter jusqu'à trois couleurs différentes. Si une de ces trois couleurs se trouve aux points de test alors la variable booléenne associée au capteur (voir chapitre Onglet « Liens ») est positionnée à 1, autrement elle est positionnée à 0.
Les trois zones d'édition peuvent contenir une définition de couleur sous la forme de trois valeurs comprises entre 0 et 255 et qui correspondent respectivement aux pourcentages de rouge, de vert et de bleu. Les pourcentages de ces trois couleurs doivent correspondre avec exactitude aux couleurs des objets à détecter pour que le capteur fonctionne.

Onglet « Options »

Touche

Définit une touche qui permet de simuler un clic sur l'objet.
Pour préciser le code de la touche, différentes syntaxes sont utilisables :

- un simple caractère : par exemple A, Z, 2,
- le caractère $ suivi du code de la touche en hexadécimal,
- le nom d'une touche de fonction, par exemple F5.

Pour les combinaisons de touches il faut ajouter au début « CTRL+ » ou « SHIFT+ »
Par exemple : « CTRL+F4 » ou « SHIFT+Z ».

La touche TAB permet d'accéder à cet objet

Si cette case n'est pas cochée, alors la touche TAB ne permet pas d'activer l'objet.

Techniques avancées

Lier des objets de façon dynamique

Cette possibilité permet de lier momentanément un objet à un autre. Pour lier un objet à un autre les paramètres « + Position » et « - Position » qui gèrent la position horizontale sont utilisés de façon particulière. Ces deux paramètres doivent contenir le nom d'une variable numérique (M). La variable « + Position » doit contenir la valeur f000 (hexadécimal) et la variable « - Position » l'identificateur de l'objet avec lequel l'attachement doit être effectué. La variable « + Position » est remise à zéro lorsque l'attachement a été réalisé. Pour annuler l'attachement d'un objet il faut placer la valeur f001 (hexadécimal) dans la variable « + Position ». Voir le chapitre : Exemple de simulation d'une partie opérative 1

Echange de paramètres entre deux objets

Un objet peut accéder aux paramètres d'un objet frère en utilisant le mot clé « SISTERPARAM ».
La syntaxe est :
SISTERPARAM(identificateur de l'objet frère , paramètre)
« paramètre » peut prendre les valeurs suivantes :

STATE	état de l'objet : valeur de Aspect / Couleur / Remplissage
-STATE	idem mais valeur négative
POSX	position sur l'axe horizontal
-POSX	idem mais valeur négative

POSY position sur l'axe des y
-POSY idem mais valeur négative
POSX+STATE position sur l'axe horizontal plus état
POSX-STATE position sur l'axe horizontal moins état
POSY+STATE position sur l'axe vertical plus état
POSY-STATE position sur l'axe vertical moins état

Détail de l'objet « Son »

Onglet « Aspect »

Taille de l'objet

Détermine les dimensions de l'objet en nombre de points. La modification de ces valeurs permet de régler très précisément la taille de l'objet.

Onglet « Sons »

Nom des fichiers sons

Le nom d'accès complet vers les fichiers « .WAV ».

Variables associées

La variable booléenne associée à chaque son.

Détail de l'objet « Boîte de dialogue »

Onglet « Aspect »

Type de boîte

Permet de choisir les différents contrôles présents dans la boîte de dialogue : un seul bouton OK, deux boutons OK et CANCEL, ou enfin deux boutons OUI et NON.

Icônes

Permet de choisir l'icône qui apparaîtra dans la boîte de dialogue. Il y a quatre icônes différentes, mais bien sur, on peut décider d'en afficher aucune. Il convient de noter aussi qu'à chaque icône est associé un son système particulier. Pour plus de renseignements à ce sujet, se reporter à la rubrique sur l'option BEEP.

Taille de l'objet

Détermine les dimensions de l'objet en nombre de points. La modification de ces valeurs permet de régler très précisément la taille de l'objet.

Beep

Permet de spécifier si l'apparition de la boîte de dialogue doit être accompagnée d'un avertissement sonore.

Titre

Permet de spécifier le titre de la boîte de dialogue.

Type de message

Il y a ici deux possibilités. Un message prédéfini est un message présent dans les variables utilisateurs de l'application d'automatisme. L'autre possibilité est de spécifier une liste de messages, et dans ce cas, le message affiché sera fonction de l'état de la variable surveillée.

Onglet « Liens »

Nom de la variable

Spécifie le nom de la variable à surveiller. On peut entrer indifféremment des variables booléennes ou numériques.

Par exemple :

m200, i0

Si la variable en question est booléenne, alors le message n°1 dans la liste s'affichera dès que l'état de cette variable passera à 1.

Pour une variable numérique, si l'option de configuration « Liste de message » est cochée, alors la boîte de dialogue s'affichera dès que la valeur sera comprise entre 1 et le nombre de messages mémorisés dans la liste.

Par exemple, si la liste contient 8 messages, alors il ne se passera rien quand la variable prendra des valeurs négatives ou supérieures à 8. Par contre, dès que sa valeur passe dans l'intervalle 1..8, alors le message approprié est affiché...

Si l'option « Message prédéfini » est activée, alors la boîte de dialogue s'affichera avec un message de longueur le contenu de la variable, et

situé dans les variables de l'application d'automatisme suivant cette variable.

Par exemple, si m200=4, cela signifie qu'un message de longueur 4 caractères est situé dans les 4 variables suivants m200, c'est à dire m201, m202, m203, m204.

Code de retour d'une boîte de dialogue

Dans le cas d'une variable booléenne, quelle que soit l'action effectuée par l'utilisateur, son contenu passera à 0. Pour une variable numérique, il existe différents codes de retour :

Appui sur un bouton OK : la variable prend la valeur 8000 (hexa)
Appui sur un bouton CANCEL: la variable prend la valeur 8001 (hexa)
Appui sur un bouton OUI : la variable prend la valeur 8002 (hexa)
Appui sur un bouton NON : la variable prend la valeur 8003 (hexa)

Remarque : L'activation d'une boîte de dialogue est basée sur le front montant, c'est à dire passage de 0 à 1 pour une variable booléenne, et passage d'une valeur extérieure à l'intervalle de la liste de messages, à une valeur inclue dans celui-ci, pour une variable numérique.

Identificateur

Permet de référencer l'objet par rapport aux autres objets.

Onglet « Messages »

Liste de messages

Entrer dans cette zone les différents messages prédéfinis.

Détail de l'objet « Programme »

Répartition du temps d'exécution

Les objets d'IRIS sont exécutés tour à tour. Cette répartition du temps d'exécution est gérée de façon transparente par le gestionnaire d'objet. Deux niveaux de priorité sont possibles pour les objets « PROG » : si la case à cocher « Exécution prioritaire » de l'onglet « Programme » est cochée, alors la totalité du programme s'exécute lorsque l'objet à la main. Dans le cas contraire, seule une ligne s'exécute avant que l'objet

ne rende la main. Des exceptions existent à cette règle : les fonctions d'accès aux variables d'automatisme (« READVAR » et « WRITEVAR ») peuvent provoquer un passage de la main, la fonction « YIELD » force un passage de la main. En mode d'exécution prioritaire, cette fonction doit obligatoirement être utilisée à l'intérieur d'une boucle pour ne pas bloquer l'exécution des autres objets.

Affichage

La surface de l'objet peut être utilisée pour afficher des informations. La fonction « PRINT » permet d'afficher des informations.

Syntaxe

Le caractère « ; » (point virgule) est utilisé comme séparateur. Des commentaires peuvent être écrits entre les chaînes « (* » et « *) ». Il n'y a pas de différence entre les majuscules et les minuscules pour les mots clés et les noms de fonctions, en revanche, pour les noms de variables la différenciation est faite.

Déclaration des variables

Les variables utilisées dans un programme doivent être déclarées avant le programme entre les mots clés « BEGINVAR; » et « ENDVAR; ».
Les types de variables utilisables sont :

INT	entier 16 bits signé
UINT	entier 16 bits non signé
LONG	entier 32 bits signé
ULONG	entier 32 bits non signé
STRING	chaîne de caractères
FLOAT	flottant

La syntaxe générale d'une déclaration est :
<type> <nom de variable>;
La syntaxe générale pour déclarer un tableau de variables est :
<type> <nom de variable> [<longueur>];

Par exemple :

```
BEGINVAR;
INT compteur;    (* un entier 16 bits signé *)
STRING chaine;   (* une chaîne *)
ULONG table[100];
(* un tableau de 100 entiers 32 bits non signés *)
ENDVAR;
```

Ecriture du programme

Le programme doit être écrit entre les mots clés « BEGIN; » et « END; ».

Exemple :

```
BEGIN;
print "Bonjour !";
END;
```

Constantes

- entier 16 bits : un nombre décimal compris entre -32768 et 32767 ou "$" suivi d'un nombre hexadécimal compris entre 0 et FFFF. Exemple : 12, -4, $abcd
- entier 32 bits : un nombre décimal compris entre -2147483648 et 214743647 suivi de "L" ou "$" suivi d'un nombre hexadécimal compris entre 0 et FFFFFFFF suivi de "L". Exemple : 10000L, -200000L, $12345678L
- chaîne de caractères : un caractère guillemet suivi des caractères de la chaîne suivi d'un caractère guillemet. Des caractères de contrôles peuvent être insérés dans une chaîne. « \ n » remplace un caractère LF (code ASCII 10), « \r » un caractère CR (code ASCII 13). Exemple : "Abcdef", "" (chaîne nulle), "Suite\r\n"
- - flottant : un nombre décimal suivi du caractère "R", le caractère "." sert de délimiteur entre la partie entière et la partie décimale. Exemple : 3.14r, -100.4r

Affectation

La chaîne « := » marque une affectation.

Exemple :

```
compteur:=4;
var:="ABCDEF";
```

Calculs

Les opérateurs de calculs sont évalués de la gauche vers la droite. Des parenthèses peuvent être utilisées pour spécifier une priorité de calcul.

Liste des opérateurs de calcul :

- + addition (concaténation pour les chaînes de caractères)
- - soustraction
- * multiplication
- / division
- << décalage à gauche
- >> décalage à droite
- ^ élévation à une puissance
- AND "et" binaire
- OR "ou" binaire
- XOR "ou exclusif" binaire

Exemples :

```
resultat:=var1*(var2+var3);
resultat:=resultat<<2;
```

Tests

Syntaxe :

```
IF <condition> THEN ... ENDIF;
```

ou

```
IF <condition> THEN ... ELSE ... ENDIF;
```

Exemple :

```
IF (count<100) AND (count>10)
      THEN
      count:=count+1;
      ELSE
      count:=0;
      ENDIF;
```

Boucles

Syntaxe :

```
WHILE <condition> DO ... ENDWHILE;
```


Exemple :

```
count:=0;
WHILE count<1000
      DO
      table[count]:=table[count+1];
      count:=count+1;
      ENDWHILE;
```

Adresse d'une variable ou d'un tableau de variables

La syntaxe &nom de variable ou &nom d'un tableau de variables retourne l'adresse d'une variable ou d'un tableau de variables. Cette syntaxe est nécessaire pour certaines fonctions.

Liste des fonctions

Pour les exemples proposés dans ce qui suit on admettra que :
vint est une variable de type INT, vlong une variable de type LONG, vuint une variable de type UINT, vulong une variable de type ULONG, vfloat une variable de type FLOAT et vstring une variable de type STRING.

PRINT

Fonction d'affichage. Les données à afficher sont écrites à la suite et séparées par des virgules. Exemple :

```
print "Le résultat est :",vint/12,"\n";
```

NOT

Complément. Cette fonction peut être utilisée avec le test if pour complémenter un résultat.
Exemple :

```
if not(1<2) then ...
```

ABS

Valeur absolue.
Exemple :

```
print abs(0-4); (* affiche 4 *)
```

VAL

Retourne la valeur d'une chaîne de caractères exprimée sous la forme d'un nombre décimal.
Exemple :

```
vlong=val("-123456"); (* vlong contiendra -123456 *)
```


HVAL

Retourne la valeur d'une chaîne de caractères exprimée sous la forme d'un nombre hexadécimal.
Exemple :

```
vuint=hval("abcd"); (* vuint contiendra abcd hexa *)
```

ASC

Retourne le code ASCII du premier caractère d'une chaîne.
Exemple :

```
vuint :=asc("ABCD"); (* vuint contiendra 65 : code ascii de 'A' *)
```

CHR

Retourne une chaîne composée de 1 caractère dont le code ASCII est passé en paramètre.
Exemple :

```
vstring:=chr(65); (*vstring contiendra la chaîne "A" *)
```

STRING

Retourne une chaîne composée de n caractères. Le premier argument est le nombre de caractères, le deuxième le caractère.
Exemple :

```
vstring:=string(100," ");
(* vstring contiendra une chaîne composée de 100 espaces *)
```

STR

Convertit une valeur numérique entière en une chaîne de caractères représentant cette valeur en décimal.
Exemple :

```
vstring:=str(100); (* vstring contiendra la chaîne "100" *)
```

HEX

Convertit une valeur numérique entière en une chaîne de caractères représentant cette valeur en hexadécimal.
Exemple :

```
vstring:=hex(100); (* vstring contiendra la chaîne "64" *)
```

LEFT

Retourne la partie gauche d'une chaîne de caractères. Le premier argument est la chaîne de caractères, le deuxième le nombre de caractères à extraire.
Exemple :

```
vstring:=left("abcdef",2); (* vstring contiendra "ab" *)
```

RIGHT

Retourne la partie droite d'une chaîne de caractères. Le premier argument est la chaîne de caractères, le deuxième le nombre de caractères à extraire.
Exemple :

```
vstring:=right("abcdef",2); (* vstring contiendra "ef" *)
```

MID

Retourne une partie d'une chaîne de caractères. Le premier argument est la chaîne de caractères, le deuxième la position où doit commencer l'extraction, le troisième le nombre de caractères à extraire.
Exemple :

```
vstring:=mid("abcdef",1,2); (* vstring contiendra "bc" *)
```

LEN

Retourne la longueur d'une chaîne de caractères.
Exemple :

```
vuint:=len("123"); (* vuint contiendra 3 *)
```

COS

Retourne le cosinus d'une valeur réelle exprimée en radian.
Exemple :

```
vfloat:=cos(3.14r); (* vfloat contiendra le cosinus de 3.14 *)
```

SIN

Retourne le sinus d'une valeur réelle exprimée en radian.
Exemple :

```
vfloat:=sin(3.14r); (* vfloat contiendra le sinus de 3.14 *)
```

TAN

Retourne la tangente d'une valeur réelle exprimée en radian.
Exemple :

```
vfloat:=tan(3.14r); (* vfloat contiendra la tangente de 3.14 *)
```

ATN

Retourne l'arc tangente d'une valeur réelle.
Exemple :

```
vfloat:=atn(0.5r); (* vfloat contiendra l'arc tangente de 0.5 *)
```

EXP

Retourne l'exponentielle d'une valeur réelle.
Exemple :

```
vfloat:=exp(1r); (* vfloat contiendra l'exponentielle de 1 *)
```


LOG

Retourne le logarithme d'une valeur réelle.
Exemple :

```
vfloat:=log(1r); (* vfloat contiendra le logarithme de 1 *)
```

LOG10

Retourne le logarithme base 10 d'une valeur réelle.
Exemple :

```
vfloat:=log10(1r);
(* vfloat contiendra le logarithme base 10 de 1 *)
```

SQRT

Retourne la racine carrée d'une valeur réelle.
Exemple :

```
vfloat:=sqrt(2); (* vloat contiendra la racine carrée de 2 *)
```

DATE

Retourne une chaîne de caractères représentant la date.
Exemple :

```
print "La date est :",date(),"\n";
```

TIME

Retourne une chaîne de caractères représentant l'heure.
Exemple :

```
print "L'heure est :",time(),"\n";
```

RND

Retourne un nombre aléatoire.
Exemple :

```
print rnd();
```

OPEN

Ouvre un fichier. Le premier argument est le nom du fichier, le deuxième le mode d'accès qui peut être : « r+b » ouverture en lecture / écriture, « w+b » ouverture en écriture (si le fichier existe il est détruit). La fonction retourne un long qui identifiera le fichier. Si l'ouverture échoue, la valeur retournée est 0.
Exemple :

```
vulong:=open("nouveau","w+b");
```

CLOSE

Ferme un fichier. L'argument est l'identificateur de fichier retourné par la fonction OPEN.
Exemple :

```
close(vulong);
```


WRITE

Ecrit des données dans un fichier. Le premier argument est l'identificateur de fichier retourné par la fonction OPEN. Le deuxième argument est l'adresse d'une variable, le troisième le nombre d'octets à écrire. La fonction retourne le nombre d'octets effectivement écrits.
Exemple :

```
vuint:=write(vulong,&buff,5);
```

READ

Lit des données dans un fichier. Le premier argument est l'identificateur de fichier retourné par la fonction OPEN. Le deuxième argument est l'adresse d'une variable, le troisième le nombre d'octets à lire. La fonction retourne le nombre d'octets effectivement lus.
Exemple :

```
vuint:=read(vulong,&buff,5);
```

SEEK

Déplace un pointeur de fichier. Le premier argument est l'identificateur de fichier retourné par la fonction OPEN, le deuxième la position.
Exemple :

```
seek(vulong,0l);
```

GOTO

Effectue un saut au label passé en argument. L'argument est une chaîne de caractères.
Exemple :

```
goto "fin"
...
fin:;
```

CALL

Effectue un saut à un sous-programme. L'argument est une chaîne de caractères contenant le label du sous-programme.
Exemple :

```
BEGIN;
(* programme principal *)
call "sp"
END;
BEGIN;
(* sous programme *)
sp:
print "Dans le sous-programme\n";
```



```
return;
END;
```

RETURN

Marque la fin d'un sous-programme.

READVAR

Lit une ou plusieurs variables de l'application d'automatisme. Le premier argument est le nom de la variable d'automatisme (nom de variable ou de symbole). Le deuxième argument est l'adresse d'une variable ou d'un tableau de variables 32 bits (longs ou flottants). Le troisième argument est le nombre de variables à lire. Si la fonction est exécutée sans erreur, la valeur retournée est 0.
Exemple :

```
readvar("i0",&buff,16); (* lecture de 16 entrées à partir de i0 *)
```

WRITEVAR

Ecrit une ou plusieurs variables de l'application d'automatisme. Le premier argument est le nom de la variable d'automatisme (nom de variable ou de symbole). Le deuxième argument est l'adresse d'une variable ou d'un tableau de variables 32 bits (longs ou flottants). Le troisième argument est le nombre de variables à écrire. Si la fonction est exécutée sans erreur, la valeur retournée est 0.
Exemple :

```
writevar("o0",&buff,16);
(* écriture de 16 sorties à partir de o0 *)
```

CMD

Exécute une commande. L'argument est une chaîne qui spécifie la commande à exécuter. Cette fonction permet d'utiliser les commandes prédéfinies d'IRIS. Pour plus de détails consultez le chapitre Ordres spéciaux. La valeur retournée est 0 si la commande a été exécutée sans erreur.
Exemple :

```
cmd("run");
```


YIELD

Passe la main. Cette fonction permet de ne pas monopoliser l'exécution lorsque l'objet est exécuté en mode prioritaire.
Exemple :

```
WHILE 1
      DO
      ...
      yield();
ENDWHILE;
```

DLL

Appelle une DLL. Le premier argument est le nom du fichier DLL. Le deuxième est le nom de la fonction. Le troisième est un pointeur sur une variable 32 bits qui recevra le code de retour de la fonction. Les autres arguments sont passés à la fonction.
Exemple :

```
dll "user","messagebeep",&vulong,-1;
```

Messages d'erreurs

« séparateur ';' manquant »	il manque un point virgule
« erreur de syntaxe »	une erreur de syntaxe a été détectée
« variable définie plusieurs fois »	une variable est définie plusieurs fois
« pas assez de mémoire »	l'exécution du programme a saturé la mémoire disponible
« variable non définie »	une variable utilisée dans le programme n'est pas définie
« constante trop grande »	une constante est trop grande
« programme trop complexe »	une expression est trop complexe, il faut la décomposer
« type de variable ou de constante incompatible »	une variable ou une constante n'est pas du type attendu
« ')' manquant »	il manque une parenthèse fermante

« ENDIF manquant »	il manque le mot-clé ENDIF
« 'ENDWHILE' manquant »	il manque le mot-clé ENDWHILE
« label introuvable »	un label de saut ou de sous-programme est introuvable
« ']' manquant »	il manque un crochet fermant
« numéro d'élément hors borne »	un élément de tableau hors limite a été utilisé
« trop de 'CALL' imbriqués »	trop de sous-programme imbriqués ont été utilisés
« 'RETURN' trouvé sans 'CALL' »	RETURN trouvé en dehors d'un sous-programme
« taille de variable trop petite »	la taille d'une variable est insuffisante
« fichier DLL introuvable »	le fichier DLL est introuvable
« fonction introuvable dans DLL »	la fonction est introuvable dans le fichier DLL
« division par zéro »	une division par 0 s'est produite
« erreur mathématique »	une fonction mathématique a provoqué une erreur

Onglet « Aspect »

Couleurs

Permet de choisir la couleur du fond et des caractères de l'objet.

Taille de l'objet

Détermine les dimensions de l'objet en nombre de points. La modification de ces valeurs permet de régler très précisément la taille de l'objet.

Texte

Permet de spécifier un texte bulle qui apparaît lorsque le curseur est placé sur l'objet.

Onglet « Programme »

Programme

Cette zone d'édition contient le programme.

Exécuter

Si cette case est cochée, alors le programme est exécuté.

Exécution prioritaire

Si cette case est cochée, alors le programme s'exécute plus rapidement.

Exécution au démarrage

Si cette case est cochée, alors le programme s'exécute à l'ouverture de l'objet. Cette option permet de sauvegarder un objet avec l'option « Exécuter » non cochée tout en demandant une exécution de l'objet au chargement.

Aller sur l'erreur

Si une erreur a été détectée pendant l'exécution du programme, alors ce bouton poussoir permet de placer le curseur à l'endroit qui a provoqué l'erreur.

Exemples IRIS 2D

Le nom des fichiers d'exemple fait référence au sous répertoire « Exemples » du répertoire où a été installé AUTOMGEN.

Exemple d'objet composé

Cet exemple va vous permettre de comprendre comment créer un objet « Clavier décimal » composé de touches : « 0 » à « 9 » plus une touche [ENTER] pour valider.

Créez un objet « Pupitre », puis à partir du menu du pupitre créez un objet « Bouton Voyant ». Nous allons paramétrer cet objet puis nous le dupliquerons pour obtenir les autres touches. Ensuite nous retoucherons les propriétés des touches dupliquées pour les personnaliser : texte affiché sur la touche et action. Nous aurons ainsi un clavier dont l'aspect des touches sera homogène.

Le lien avec l'application sera réalisé par l'intermédiaire d'un mot.

Lorsqu'une touche sera enfoncée elle écrira son code (0 à 9 ou 13 pour la touche de validation) dans ce mot.

Pour spécifier ce mot nous pourrions donner son numéro sous la rubrique action des propriétés de chacun des objets. L'inconvénient est que lorsque nous réutiliserons l'objet « Clavier décimal » et si nous voulons utiliser un autre mot il faudra modifier les propriétés des 11 objets « Bouton Voyant ».

Pour contourner ce problème nous allons utiliser la possibilité donnée aux objets enfants d'accéder à un paramètre défini dans les propriétés d'un pupitre parent. L'onglet « Liens » de la fenêtre des propriétés du pupitre permet de définir le paramètre. Ecrivez dans la zone d'édition sur une seule ligne : « CLAVIER=M200 ». Cette ligne signifie que le paramètre clavier est égal à M200.

Les touches du clavier feront référence au paramètre « CLAVIER » et non directement au mot M200. Ainsi pour changer le mot utilisé il suffira de modifier la définition du paramètre dans les propriétés du pupitre.

Revenons à l'aspect de notre clavier ...

Pour que l'aspect du clavier soit satisfaisant nous allons définir une grille pour aligner les touches. Dans la fenêtre des propriétés du pupitre et l'onglet « Options » écrivez la valeur « 10 » sous les deux rubriques « Grilles ». Ainsi la fonction déplacer du menu du pupitre utilisera une grille de 10 pixels. Nous allons également fixer les dimensions de la première touche. Nous pourrions directement modifier les dimensions de la touche en la saisissant par un de ses bords, cependant pour plus de précision nous allons modifier directement les dimensions sous la rubrique « Taille de l'objet en pixel » de l'onglet « Aspect » de la fenêtre des propriétés de l'objet « Bouton Voyant ».

Entrez par exemple « 30 » pour la largeur et la hauteur.

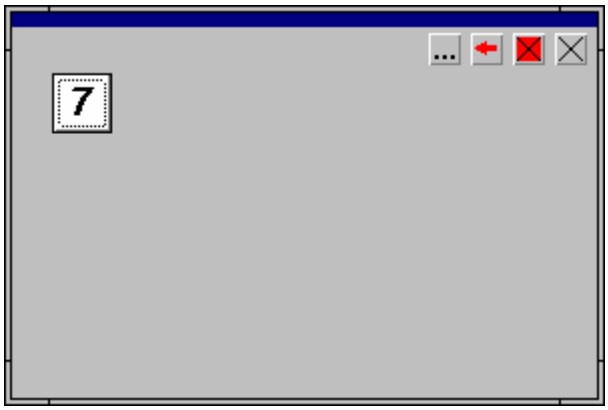
A ce stade, vous pouvez également personnaliser le style de la touche : la couleur, la fonte utilisée pour le marquage, etc ...

Nous placerons cette première touche en haut et à gauche du clavier (c'est un choix arbitraire). Le clavier que nous allons créer ressemblera au pavé numérique d'un clavier d'ordinateur. Nous allons donc marquer cette touche avec le texte « 7 » sous la rubrique « Texte » de l'onglet « Aspect ».

Nous allons également paramétrer l'aspect fonctionnel de la touche : sous la rubrique « Action lorsque le bouton est enfoncé » de l'onglet « Liens » nous allons écrire: « PARENTPARAM(CLAVIER)=7 ». Ce qui signifie que lorsque le bouton poussoir sera enfoncé le mot désigné par le paramètre « CLAVIER » du pupitre parent recevra la valeur 7. Effacez ce qui se trouve sous la rubrique « Action lorsque le bouton est relâché ».

Nous pouvons également affecter à l'objet « Bouton Voyant » une touche du clavier de l'ordinateur. Ainsi il sera possible d'utiliser le clavier avec la souris ou le clavier de l'ordinateur. Pour affecter une touche à l'objet « Bouton Voyant », utilisez la rubrique « Touche » de l'onglet « Options ». Entrez par exemple « 7 » pour associer la touche « 7 » du clavier de l'ordinateur à l'objet.

Placez ensuite la touche « 7 » en haut et à gauche du clavier, comme ceci :



Pour déplacer cette touche sélectionnez d'abord l'objet (touche [SHIFT] enfoncée puis cliquez avec le bouton gauche de la souris sur l'objet), puis utilisez la fonction « Déplacer » du menu du pupitre. Cette fonction est la seule qui utilise la grille contrairement au déplacement en saisissant la barre des objets enfants.

Pour créer les autres touches, dupliquez la touche existante :

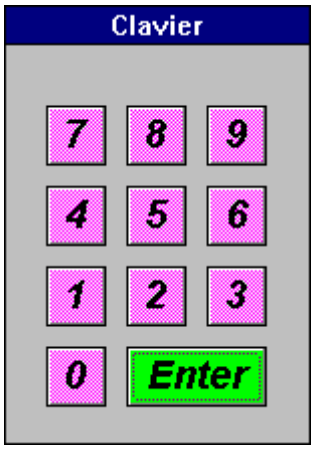
- sélectionnez la première touche,
- choisissez « Copier » dans le menu du pupitre, puis « Coller »,
- déplacez la touche ainsi collée,
- paramétrez la nouvelle touche : (texte, liens et touche du clavier de l'ordinateur).

Lorsque vous avez terminé la rangée du haut (touches « 7 », « 8 » et « 9 ») vous pouvez sélectionner ensemble ces trois touches et les dupliquer.

Vous pouvez créer une touche de validation (plus large pour remplir la surface du clavier).

Pour terminer, redimensionnez le pupitre, passez les objets en mode « Exploitation ».

Le résultat final doit ressembler à ceci :



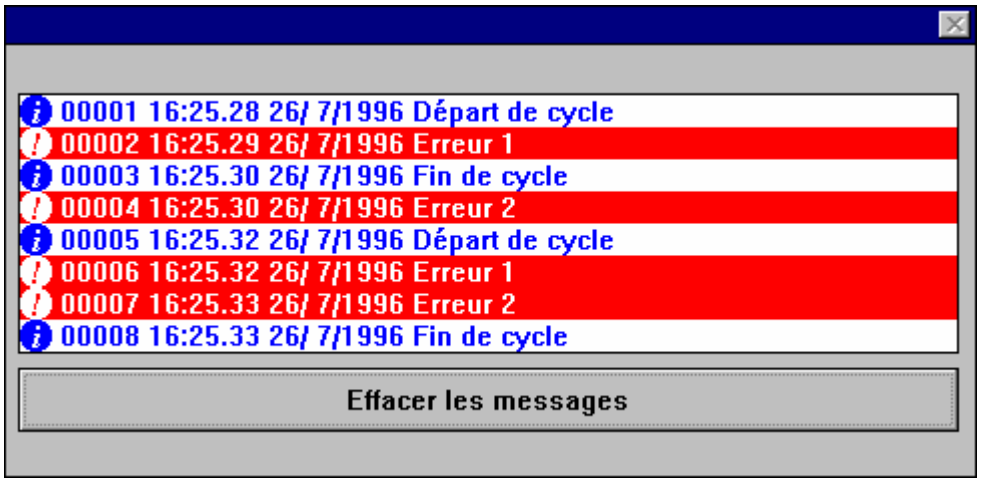
« Exemples\IRIS2D\clavier.agn »

Exemple d'utilisation de l'objet « Ecran, clavier, liste à messages » comme liste à messages

Cahier des charges :

- l'objet doit afficher quatre messages différents suivant l'état de quatre entrées (i0 à i3),
- pour l'entrée 0 : un message d'information « Départ de cycle »,
- pour l'entrée 1 : un message d'information « Fin de cycle »,
- pour l'entrée 2 : un message d'erreur « Erreur 1 »,
- pour l'entrée 3 : un message d'erreur « Erreur 2 »,
- les messages doivent être affichés à l'apparition du front montant des entrées,
- un historique de 50 messages sera conservé dans l'objet et sauvegardé sur disque,
- les messages seront dupliqués sur une imprimante connectée sur « LPT1 : »,
- un bouton poussoir doit permettre d'effacer les messages.

Solution :



🖫 « Exemples\IRIS2D\écran clavier.agn »

Variante :

L'appui sur le bouton poussoir « Effacer les messages » provoque l'ouverture d'une boîte de dialogue « Voulez vous effacer les messages » avec un choix OUI ou NON.

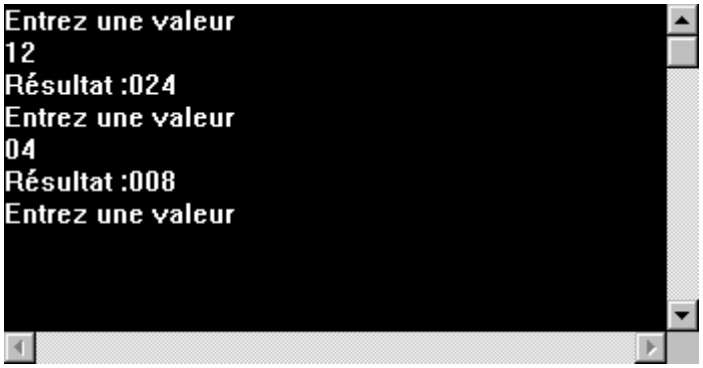
Solution :

🖫 « Exemples\IRIS2D\Ecran clavier 2.agn »

Exemple d'utilisation de l'objet ECRANCLA comme terminal

Cahier des charges :
Afficher un message « Entrez une valeur », attendre une valeur décimale tapée au clavier (deux caractères) puis afficher cette valeur multipliée par deux derrière le texte « Résultat : ».

Solution :



🖫 « Exemples\IRIS2D\terminal 1.agn »

Variante :
Les messages affichés sont stockés dans l'objet et non plus dans l'application d'automatisme.

Solution :
🖫 « Exemples\IRIS2D\terminal 2.agn »

Exemple d'application composée de plusieurs pages

Cet exemple va vous permettre de comprendre comment créer une application composée de plusieurs éléments : dans ce cas particulier un menu qui permet d'accéder à deux pages différentes.

🖫 « Exemples\IRIS2D\menu.agn »

Exemple d'utilisation de l'objet OBJET

Simulation d'un vérin.

Cahier des charges :

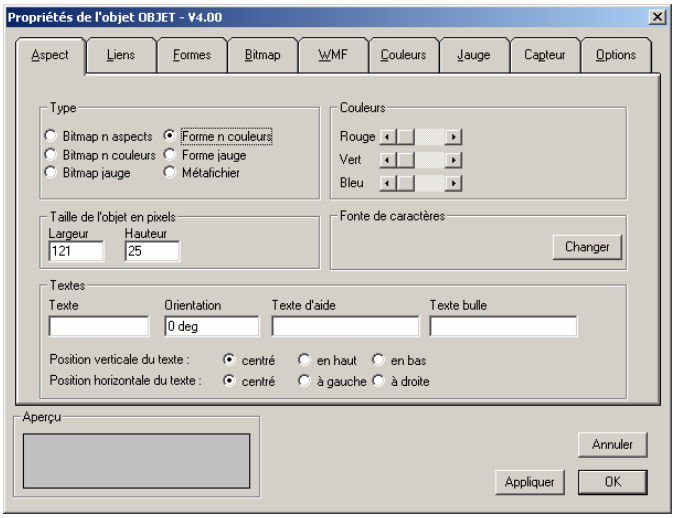
- vérin piloté par deux sorties o0 (sortir le vérin) et o1 (rentrer le vérin),
- deux entrées de fin de course i0 (vérin rentré) et i1 (vérin sorti).

Trois objets seront utilisés :

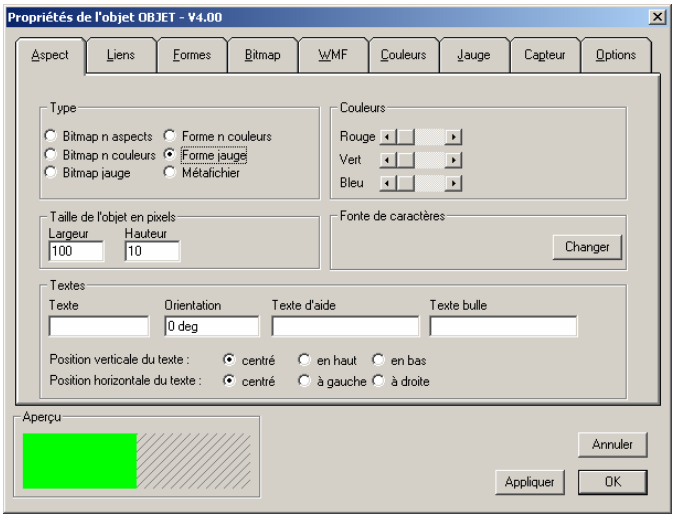
- un objet « Pupitre » servant de support,
- un objet « Objet » servant de corps de vérin,
- un objet « Objet » servant de tige de vérin.

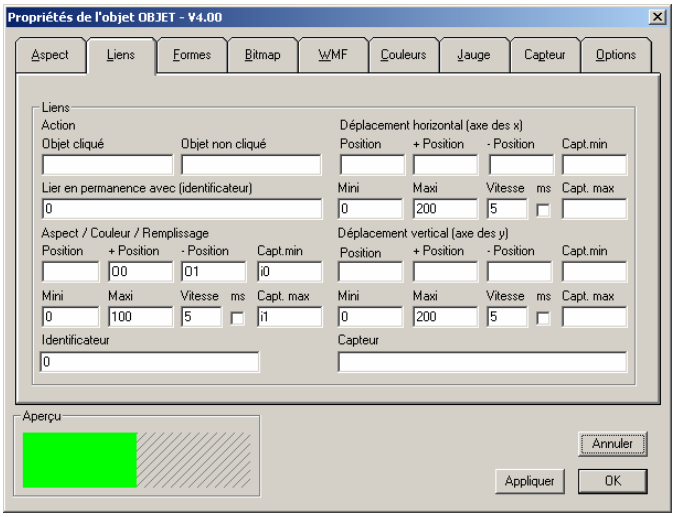
Solution :

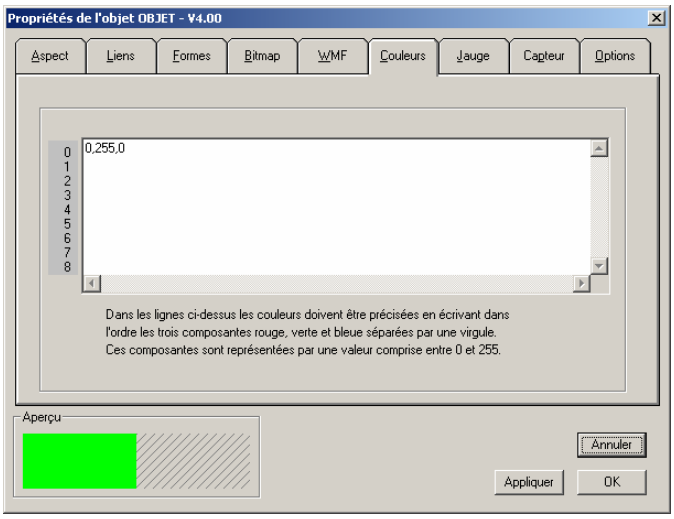
Le corps du vérin est un objet OBJET qui reste statique, seul son aspect est configuré :

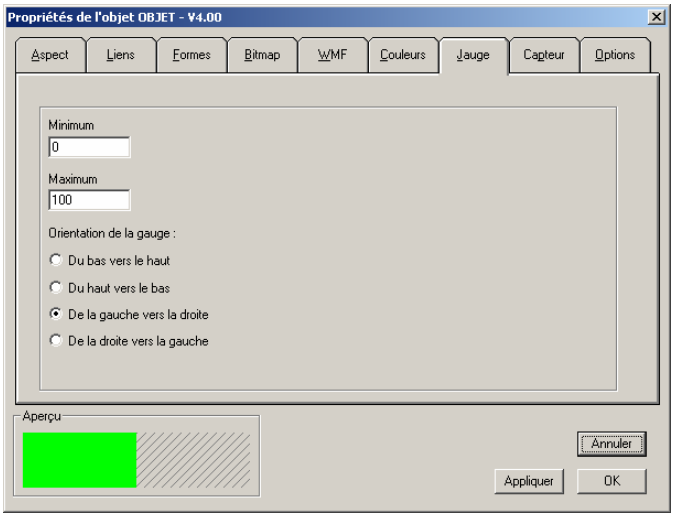


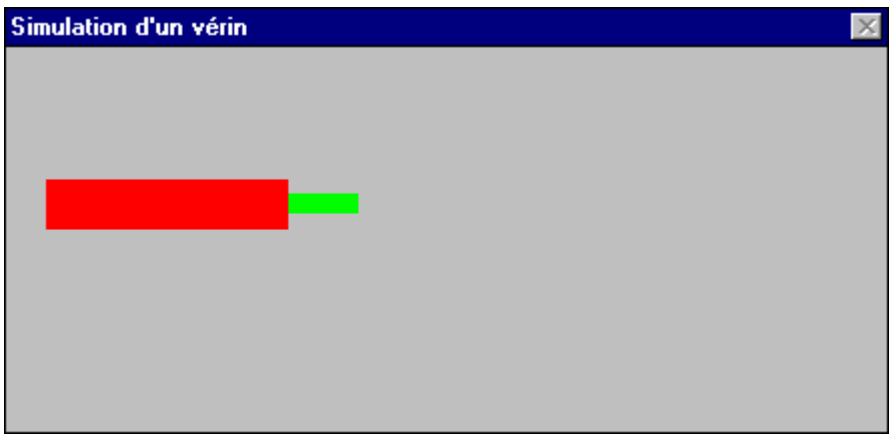
La tige du vérin est un objet OBJET configuré de la façon suivante :











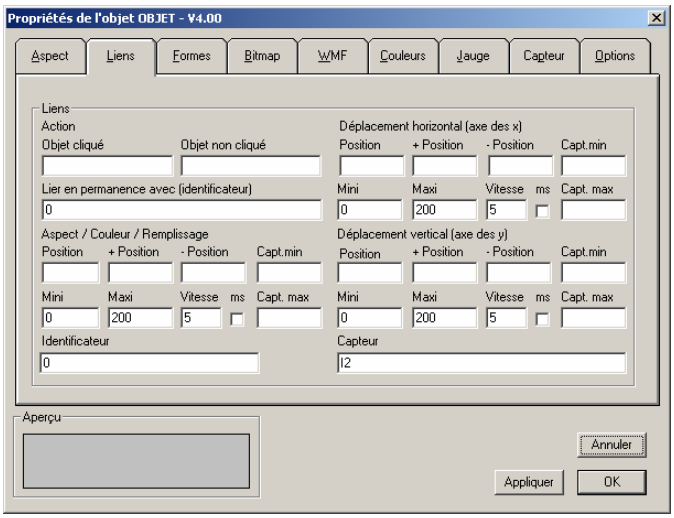
« Exemples\Simulation PO\2D\tutorial1.agn »

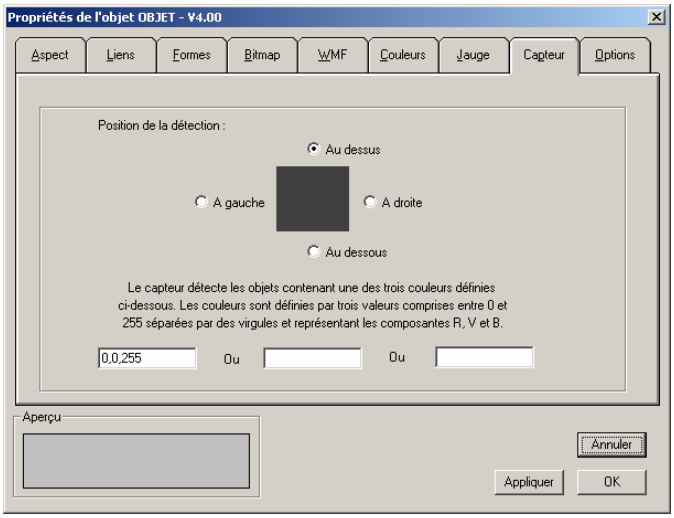
Variante :

Une position intermédiaire doit être ajoutée sur le vérin. Pour cela nous allons utiliser deux objets supplémentaires : une pièce fixée sur la tige du vérin qui actionnera un capteur et un capteur.

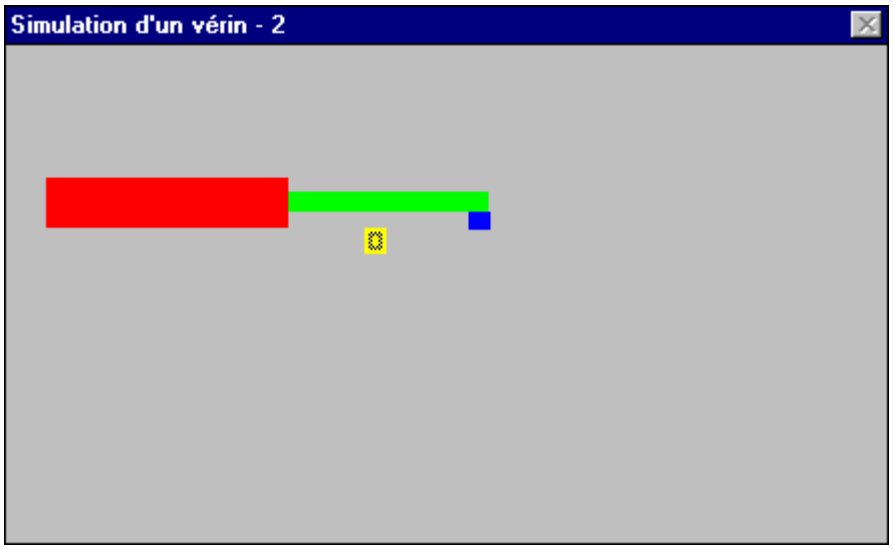
Pour lier la pièce actionnant le capteur à la tige du vérin il faut associer à la tige du vérin un identificateur : sous la rubrique « Identificateur » de l'onglet « Liens » écrire « 100 ». Pour lier la pièce à la tige, écrire sous la rubrique « Déplacement horizontal, Position » de l'onglet « Liens » : « SISTERPARAM(100,STATE) ». Ceci a pour effet de lier la pièce avec l'état de la tige du vérin.

L'objet utilisé comme capteur est paramétré comme suit :



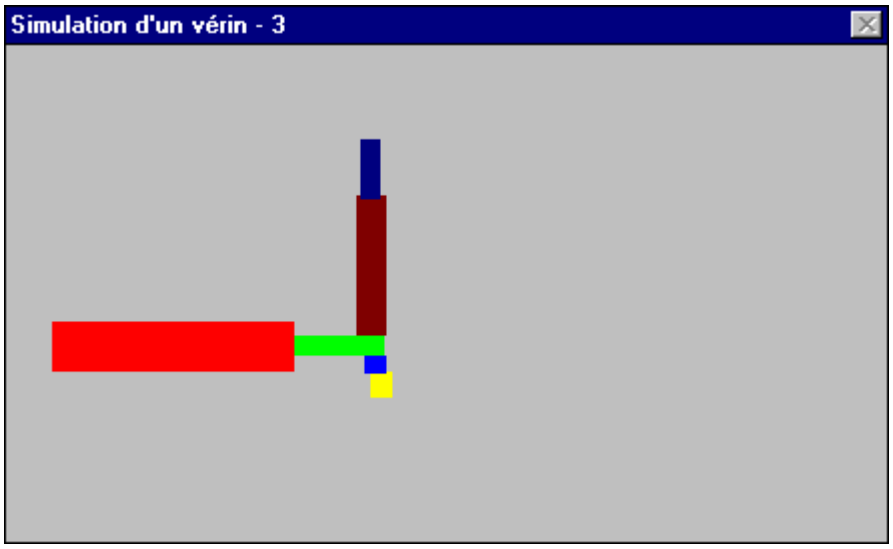


Le résultat est le suivant :



« Exemples\Simulation PO\2D\tutorial2.agn »

Deuxième variante :
Un vérin vertical fixé sur la tige du vérin horizontal est ajouté. Ce vérin est actionné par une seule sortie (O2=1 pour sortir le vérin, O2=0 pour le rentrer). Deux fins de course sont associés i3 et i4.
Le résultat est le suivant :



« Exemples\Simulation PO\2D\tutorial3.agn »

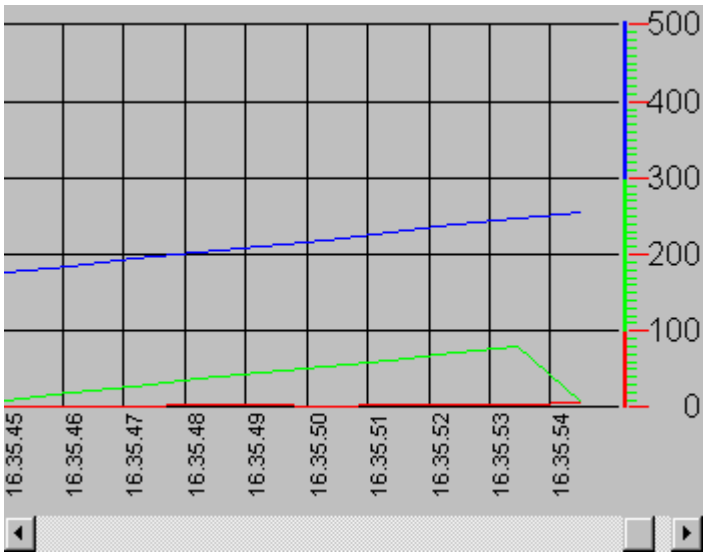
Deux objets "Objets" sont ajoutés : un pour le corps du vérin et un pour la tige.

Exemples d'utilisation de l'objet ARCHIVE

Cahier des charges :

- archiver l'état de 3 mots de l'application d'automatisme (m31 à m33) toutes les secondes,
- l'état des 4 mots sera affiché sur une courbe laissant apparaître 10 secondes d'acquisition,
- 1000 valeurs seront mémorisées dans l'objet,
- les acquisitions seront archivées dans un fichier « data.txt » au format texte.

Solution :



« Exemples\IRIS2D\archivage »

Exemple d'utilisation de l'objet PROG

Cahier des charges :

- l'appui sur un bouton poussoir doit provoquer l'inversion de l'état des sorties O0 à O99.

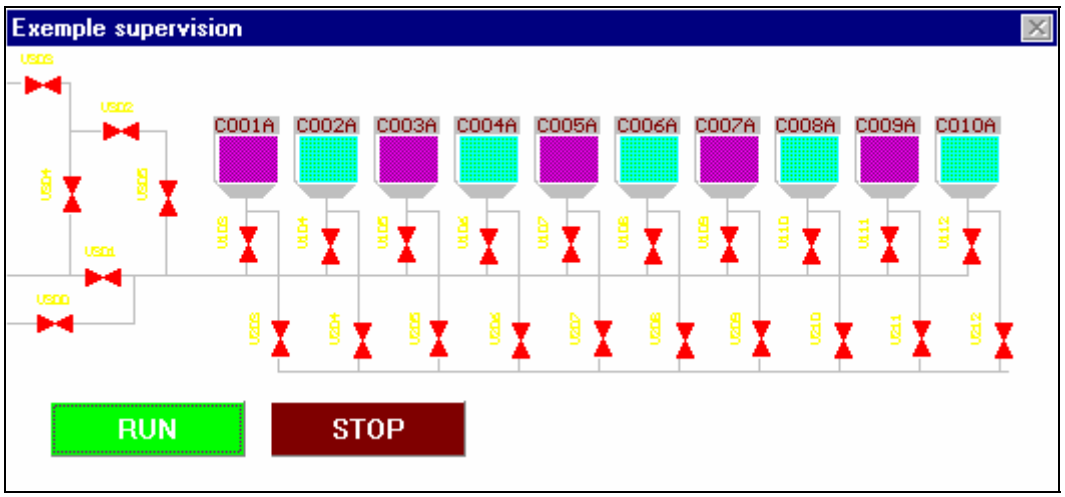
Solution :

« Exemples\IRIS2D\programme.agn »

Exemples d'application de supervision 1

L'exemple suivant illustre la création d'une application de supervision. L'application de supervision affichera l'état de vannes et de niveaux de cuves. L'action de l'utilisateur sur les vannes aura pour effet d'inverser l'état de la vanne (ouverte ou fermée). L'état RUN/STOP de l'application d'automatisme sera également affiché et deux boutons poussoirs permettront de passer en RUN ou en STOP.

Le résultat est le suivant :



« Exemples\IRIS2D\supervision 1 »

Des objets "Objet" seront utilisés pour représentés les vannes. Un fichier bitmap est créé pour représenter les vannes : état ouvert (couleur verte) et état fermée (couleur rouge) :

Exemple d'application de supervision 2

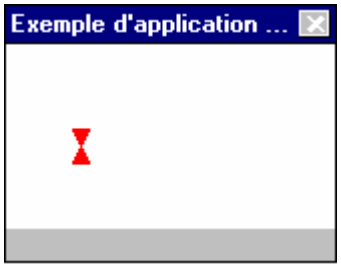
Cet exemple illustre une utilisation plus évoluée d'un objet OBJET. L'application affiche l'état d'une vanne qui peut être :

- vanne ouverte (ouverture commandée et capteur vanne ouverte vrai) : couleur verte,
- vanne fermée (fermeture commandée et capteur vanne fermée vrai) : couleur rouge,

- vanne en cours d'ouverture (ouverture commandée et capteur vanne ouverte faux) : couleur bleue,
- vanne en cours de fermeture (fermeture commandée et capteur vanne fermée faux) : couleur violette.

L'utilisateur peut inverser l'état de la vanne en cliquant dessus.

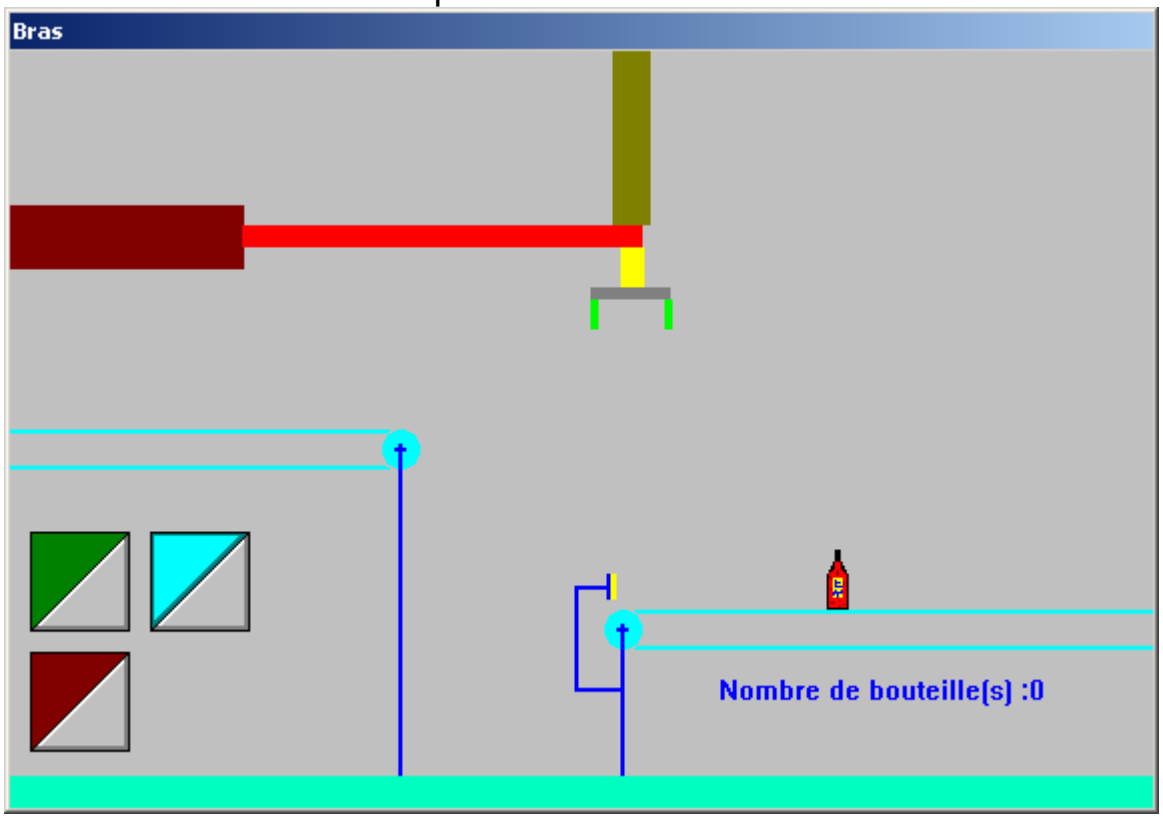
L'application d'automatisme gère l'état de la vanne.



« Exemples\IRIS2D\supervision 2.agn »

Exemple de simulation d'une partie opérative 1

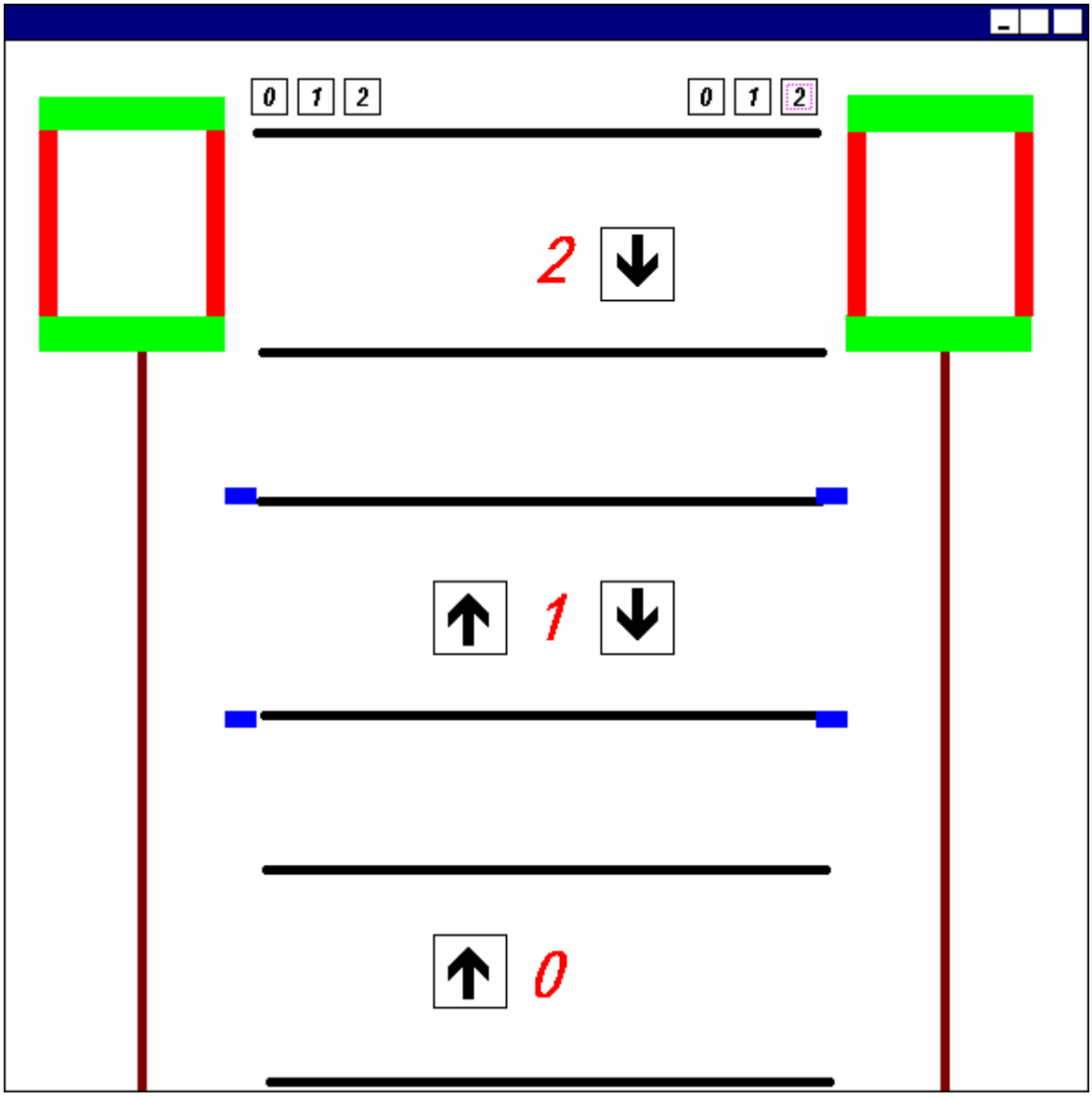
Simulation d'un bras manipulateur



« Exemples\Simulation PO\2D\bras manipulateur.agn »

Exemple de simulation d'une partie opérative 2

Simulation d'un ascenseur



« Exemples\Simulation PO\2D\ascenseur.agn »

Les objets IRIS 2D permettent de créer des applications de supervision et de simulation de parties opératives 2D.

Références IRIS 3D

IRIS 3D vous permet de créer des applications de simulation de parties opératives 3D. Le moteur physique TOKAMAK est intégré à IRIS3D pour permettre une simulation physique réaliste : pesanteur, interactions entre objets.

IRIS 3D permet d'animer des objets 3D créés dans des modeleurs standards : 3D STUDIO, SOLIDWORKS, SOLIDCONCEPTER, etc …

IRIS 3D permet également d'importer et d'exporter des objets « évolués » composés de dessin 3D et de comportement. Ces fichiers portent l'extension « .i3d ». Des fichiers standards (un vérin par exemple) sont présents dans le sous répertoire « i3d » du répertoire d'installation d'AUTOMGEN.

Le format natif des fichiers traités par IRIS 3D est celui des fichiers « .x » défini par DIRECTX 8 de Microsoft.

Un convertisseur du format « .3ds » vers « .x » et « .vrl » (format VRML 1.0) est intégré à l'environnement.

L'utilitaire CROSSROADS fourni sur le CD-ROM d'installation d'AUTOMGEN ou téléchargeable sur www.irai.com permet de convertir un nombre important de format de fichier 3D vers le format « .3ds ».

IRIS 3D se présente sous la forme d'une fenêtre encapsulée dans un pupitre IRIS 2D. Sur ce pupitre viendront s'animer les objets 3D.

Chaque fichier 3D représentera un objet dans IRIS 3D. Les éléments d'une partie opérative devant avoir un mouvement propre, devront être représentés par des fichiers séparés. Par exemple, pour un vérin composé d'un corps et d'une tige, il faut créer un fichier pour le corps du vérin et un pour la tige du vérin.

Pour créer l'animation des objets dans le monde 3D, un ou plusieurs comportements peuvent être appliqués à chacun des objets. Un comportement se compose d'une modification de l'objet (déplacement, changement de couleur, etc …) et d'un lien avec les variables de l'application d'automatisme pour conditionner cette modification. Par

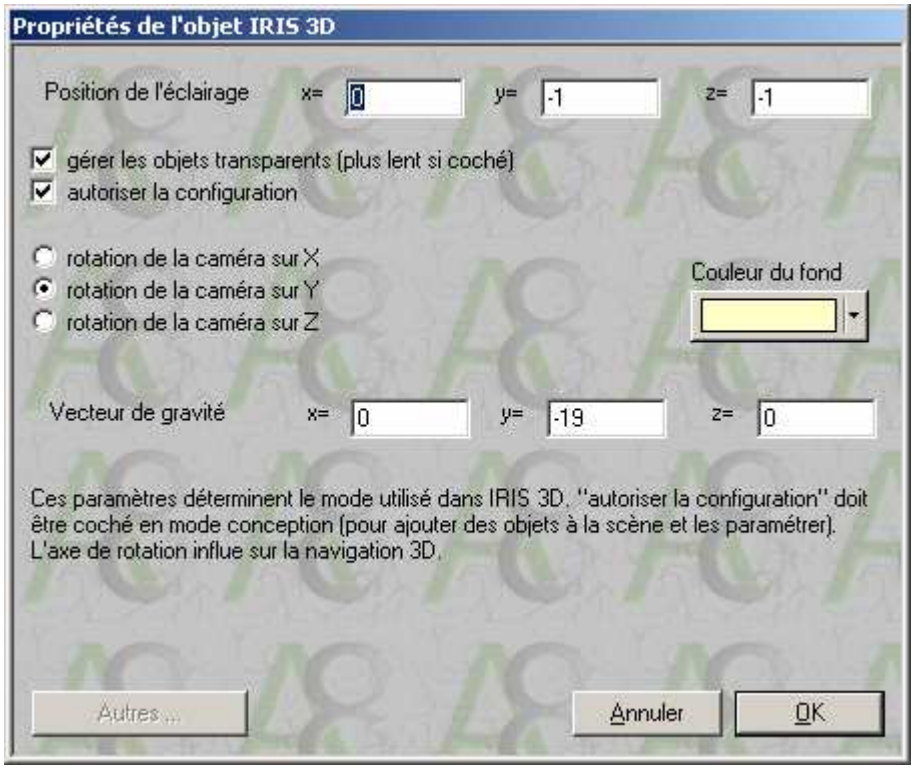
exemple : sortir la tige du vérin si la sortie 1 de l'application d'automatisme est vraie.

Tutorial

Un fichier WORD contenant un tutorial consacré à la création de parties opératives 3D se trouve dans les sous répertoires « exemples\simulation PO\3d\tutorial 2 » du répertoire d'installation d'AUTOMGEN.

Créer un pupitre IRIS 3D

Cliquez avec le bouton droit de la souris sur l'élément « Iris » dans le navigateur et choisissez l'option « Ajouter un pupitre IRIS 3D ».



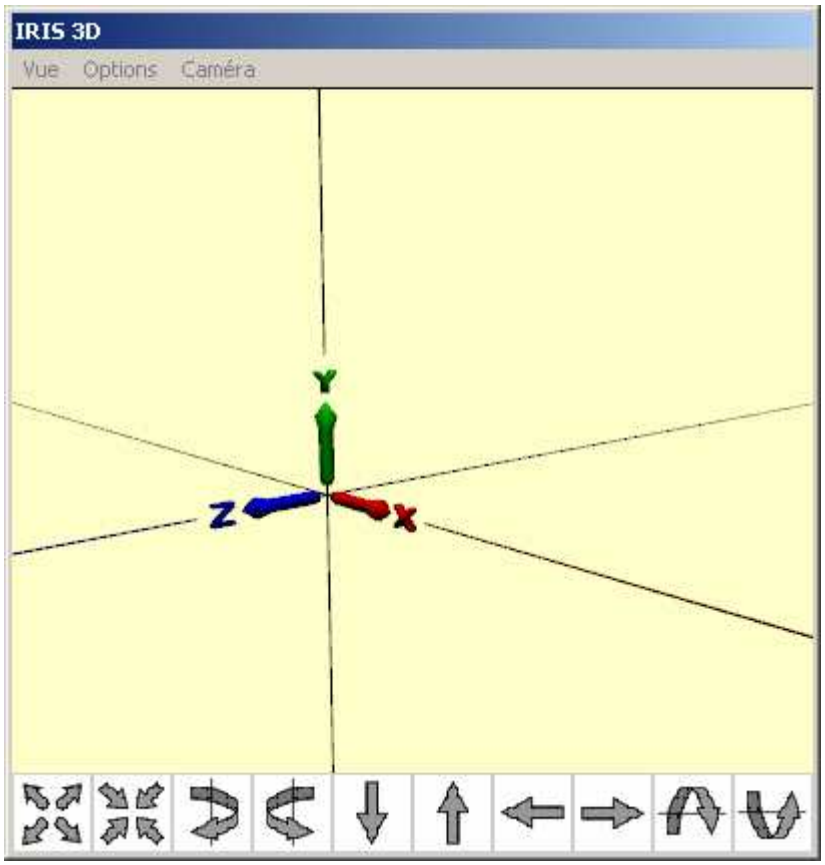
Création d'un pupitre IRIS 3D

La position de l'éclairage peut être modifiée ainsi que la gestion ou non des objets transparents, le verrouillage de la configuration, la couleur du fond, l'axe de rotation par défaut du point de vue et la gravité pour le moteur physique.

Ajouter des fichiers 3D au projet

Cliquez avec le bouton droit de la souris sur l'élément « Ressources » dans le navigateur et choisissez « Importer un ou des fichiers 3D » dans le menu. Sélectionnez un ou plusieurs fichiers « .3DS » ou « .VRL ». (si

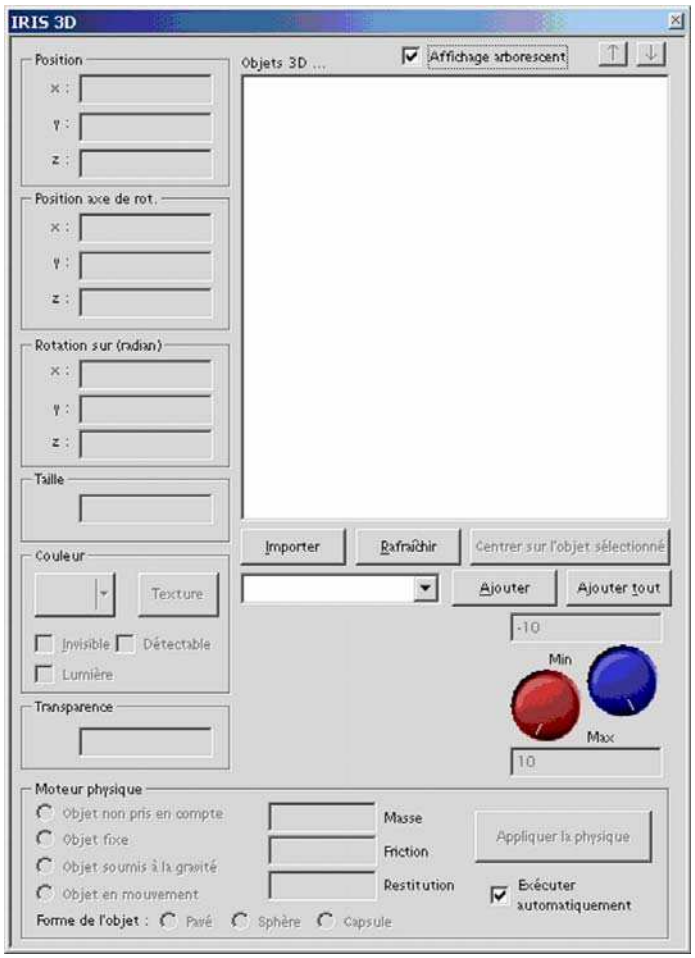
vos fichiers ne sont ni au format « .3DS », ni au format « .VRL » utilisez l'utilitaire « CROSSROAD » pour les convertir).



Le pupitre IRIS 3D

Configurer les objets

Choisissez « Ouvrir la fenêtre de configuration » dans le menu « Options » de la fenêtre IRIS 3D.

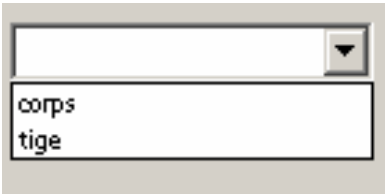


La fenêtre de configuration d'IRIS 3D

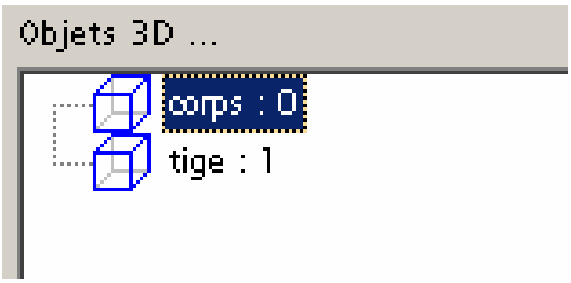
La liste des objets apparaît dans la liste. Les objets liés à un objet apparaissent comme sous éléments si la case à cocher « Affichage arborescent » est cochée.

Ajouter des objets dans le monde 3D

En cliquant sur l'élément  vous accédez à la liste des objets 3D présents dans les ressources. Par exemple :



En sélectionnant un objet dans cette liste et en cliquant sur « Ajouter » vous ajouter l'objet sélectionné au monde 3D. En cliquant sur « Ajouter tout » vous ajouter l'ensemble des objets de la liste au monde 3D. Les objets ainsi ajoutés apparaissent dans la liste de la fenêtre de configuration.



Enlever un fichier 3D des ressources

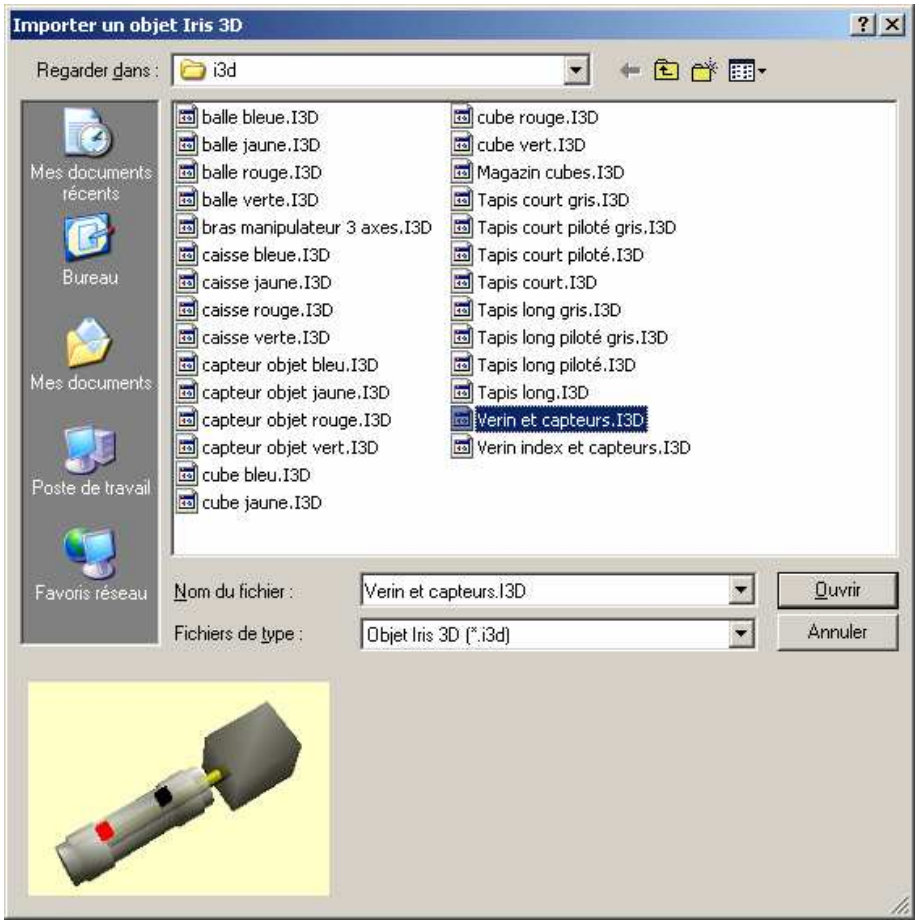
Cliquez avec le bouton droit de la souris sur le fichier 3D dans le navigateur et choisissez « Effacer ». L'objet doit également être effacé du monde 3D.

Enlever un objet du monde 3D

Cliquez avec le bouton droit de la souris sur l'objet dans la fenêtre de configuration d'IRIS 3D et choisissez « Effacer » dans le menu.

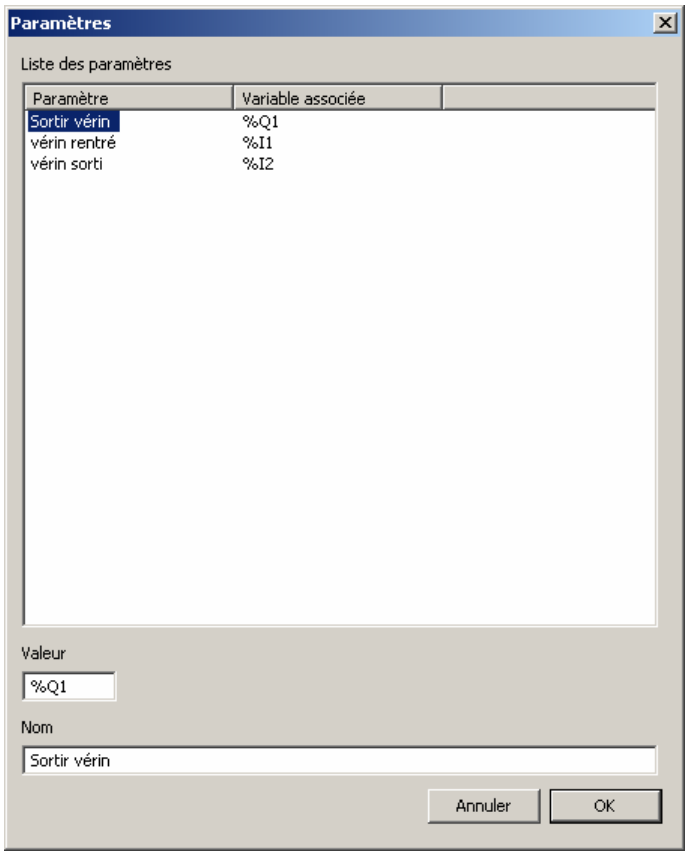
Importer un objet « évolué »

Cliquez sur le bouton « Importer ». Un navigateur vous permet de sélectionner l'objet à importer.



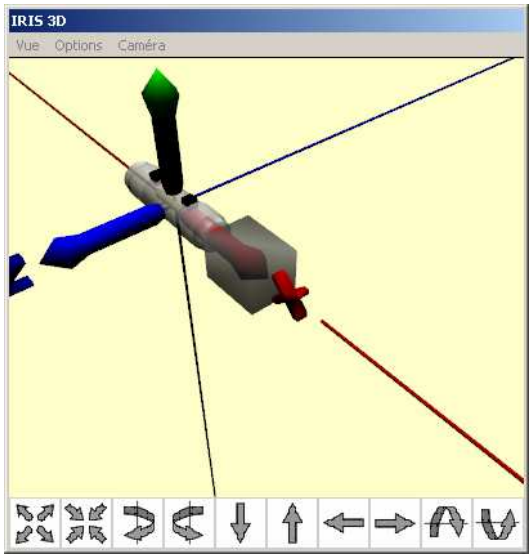
Le navigateur de sélection des objets « évolués »

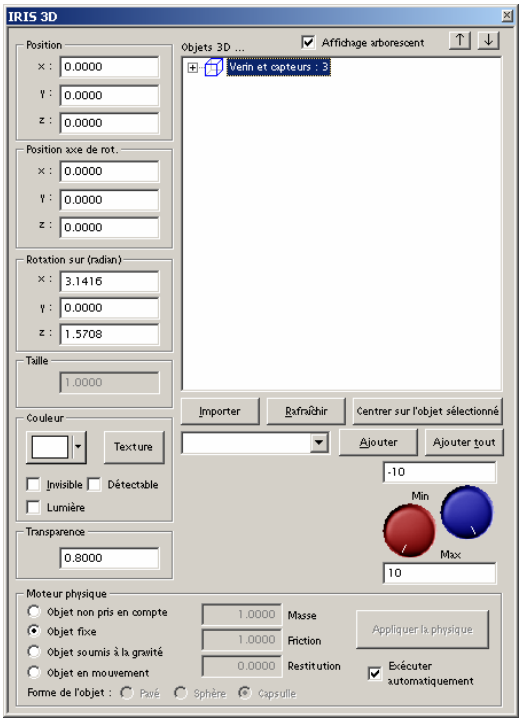
Une fois l'objet sélectionné, cliqué sur « Ouvrir ». Une fenêtre de paramétrage vous permet ensuite de définir les variables qui seront en relation avec l'objet.



La fenêtre de paramétrage de l'objet

Dans cet exemple (pour le vérin), la variable de pilotage du vérin et les deux fins de course sont à paramétrer. L'objet apparaît ensuite dans le monde 3D et dans la liste des objets.

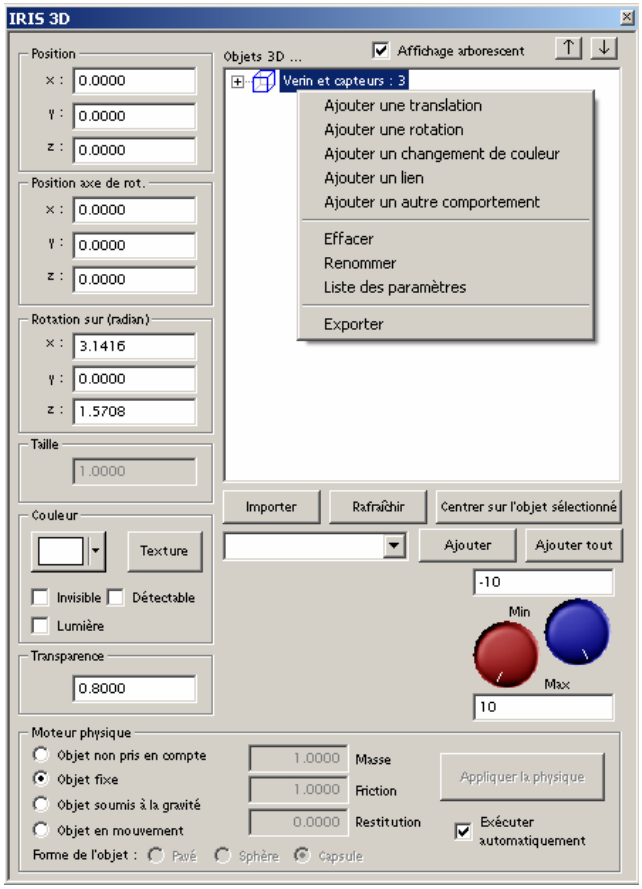




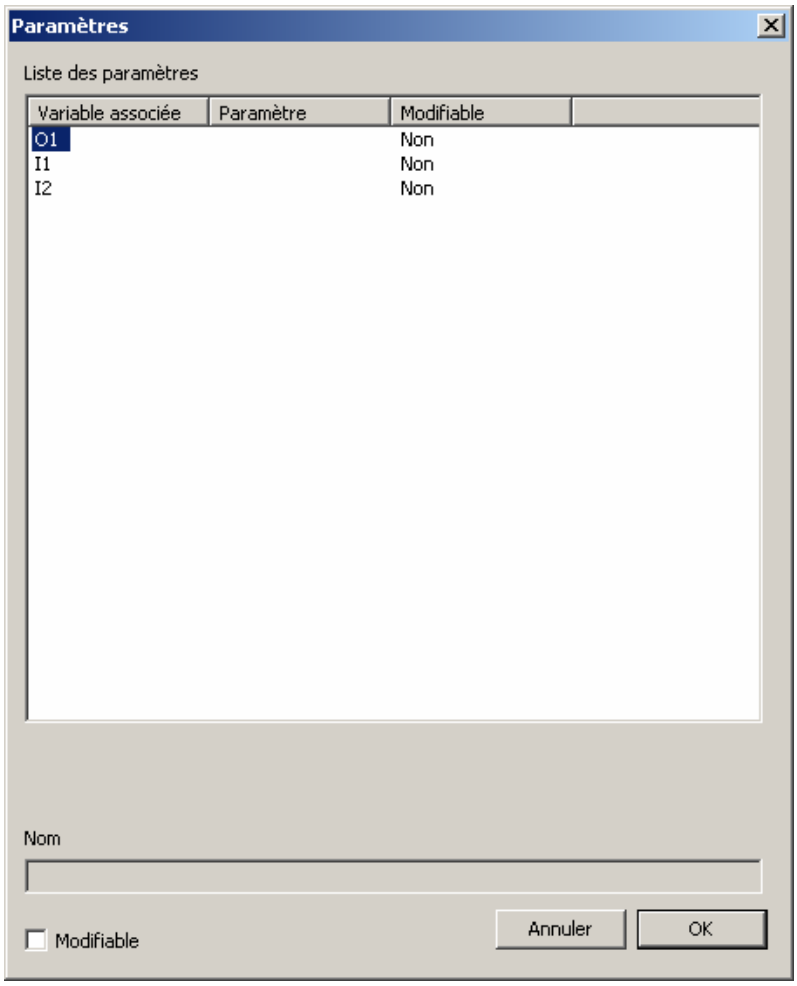
La position et l'orientation de l'objet peuvent être modifiées.

Exporter un objet « Evolué »

Pour exporter un objet, cliquez avec le bouton droit de la souris sur l'objet et choisissez « Exporter ». Les objets liés et l'ensemble des comportements sont sauvegardés.

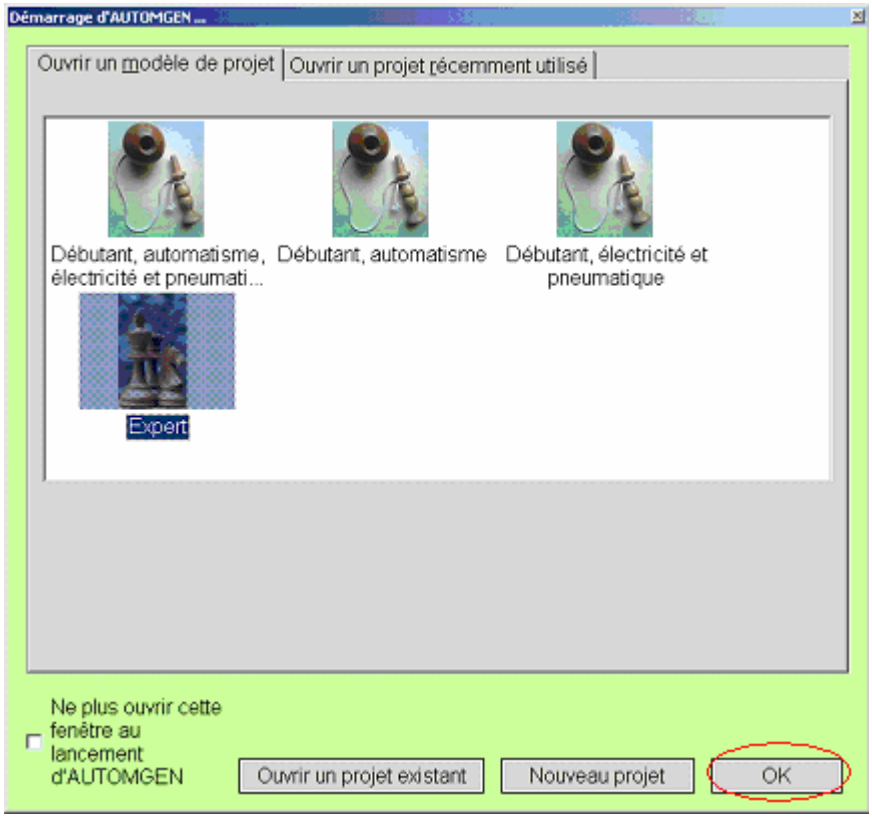


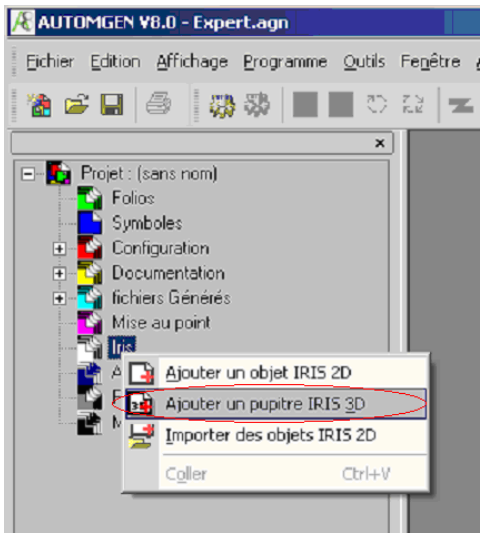
Après avoir entré un nom pour le fichier, une boîte de dialogue permet d'assigner un nom à chaque variable utilisée dans les comportements et de définir si ce paramètre sera modifiable ou non lors de la relecture.

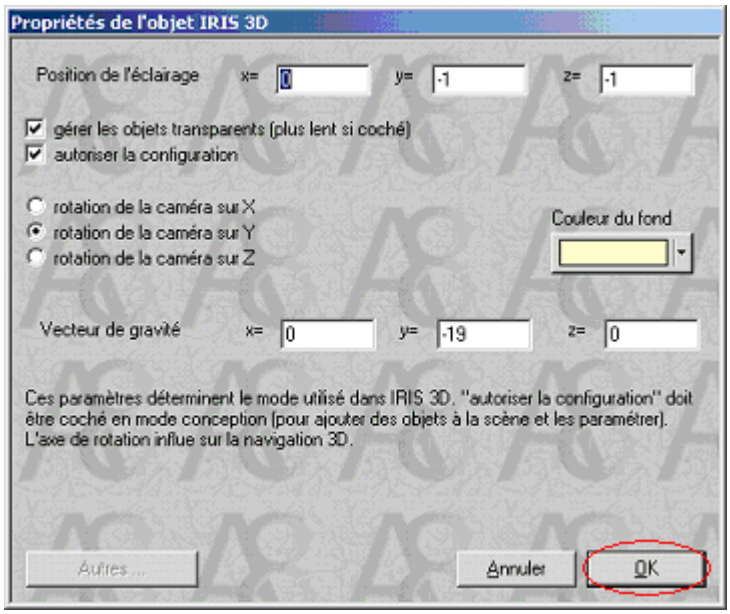


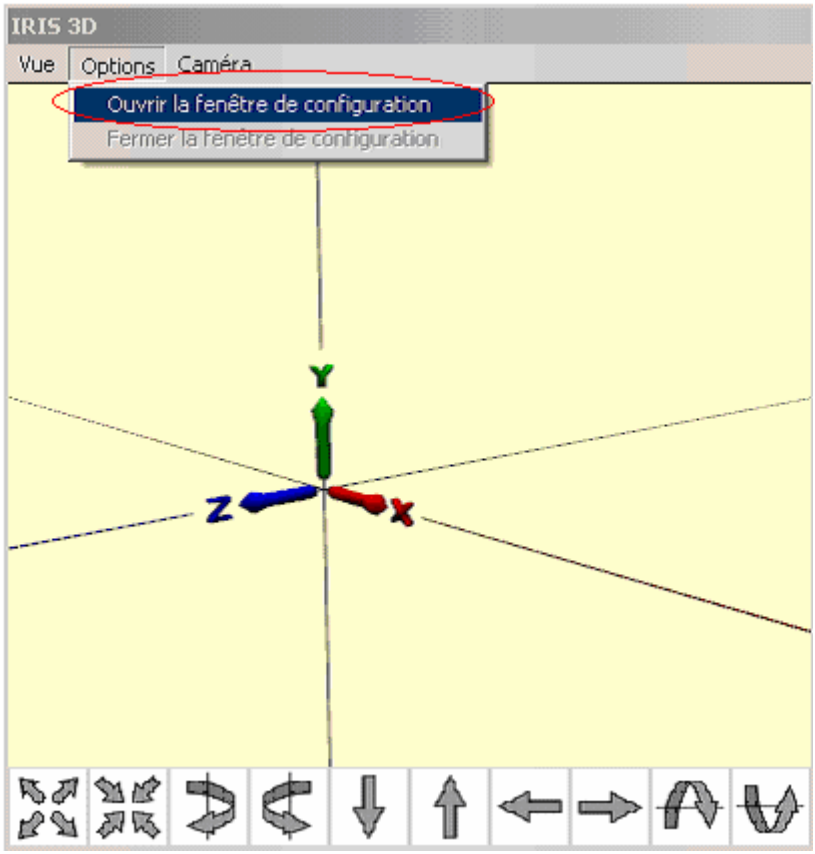
Exemple de création d'une simulation 3D à base d'objets évolués

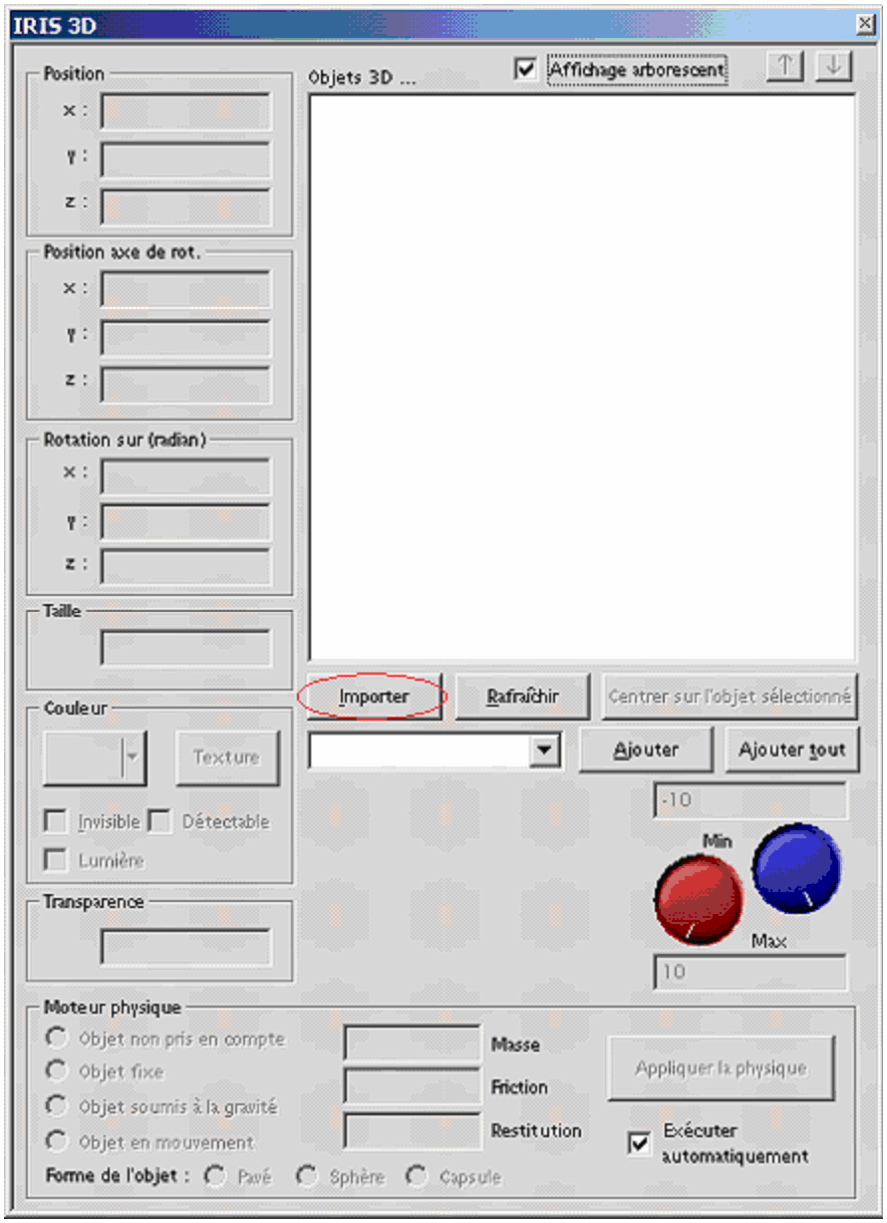
Créons en quelques clics une simulation de partie opérative : un déstockeur de pièces.

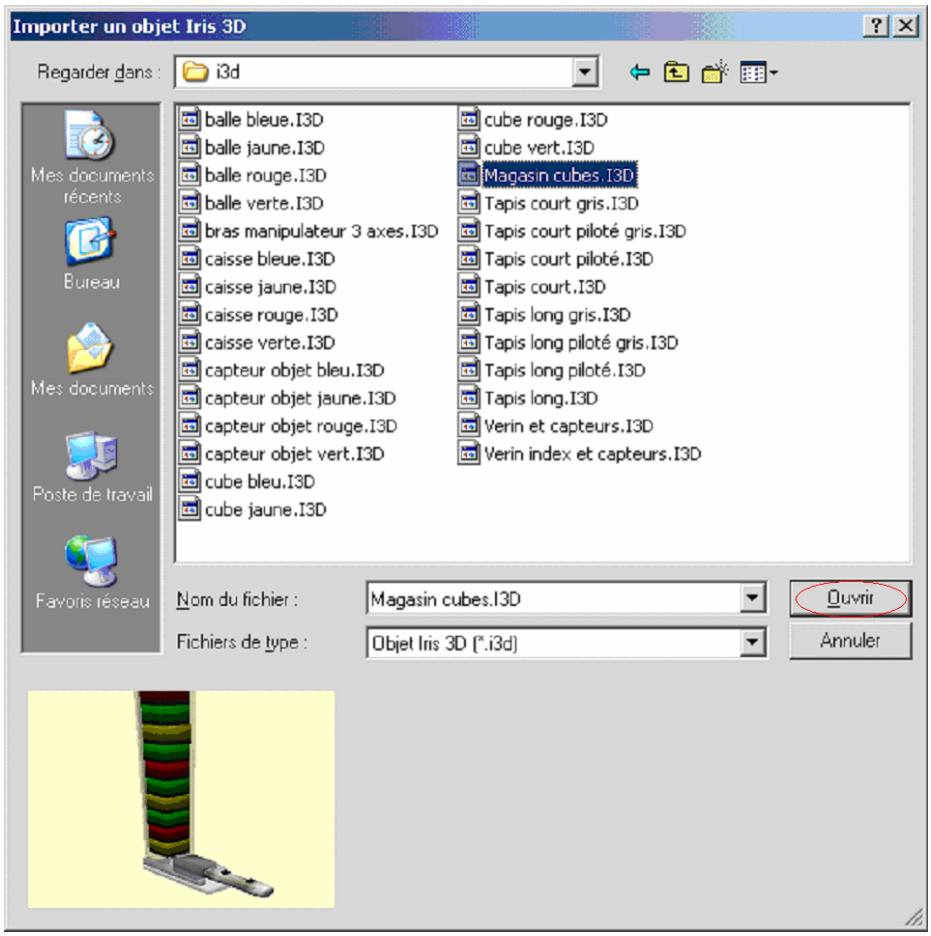




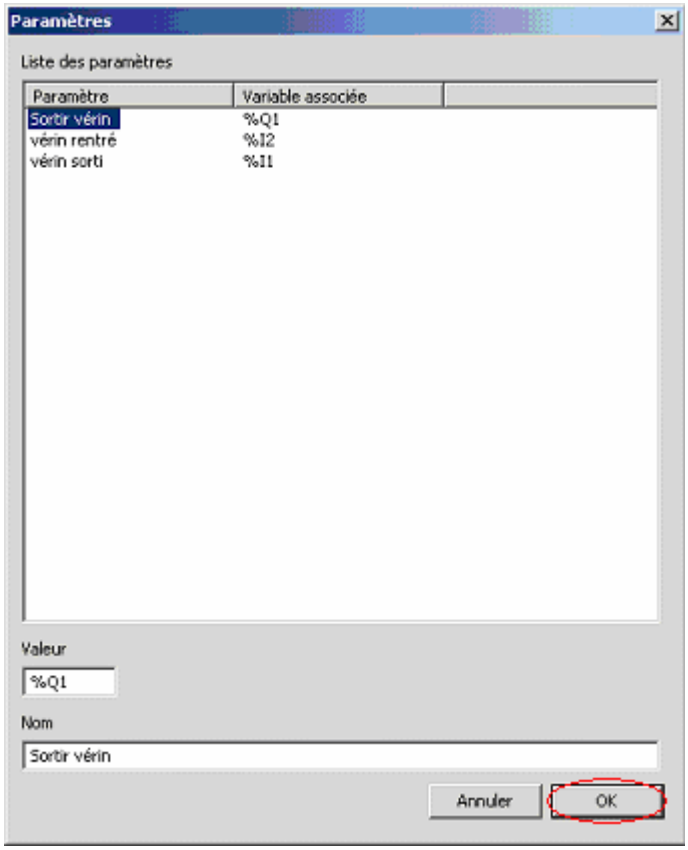

Propriétés de l'objet IRIS 3D
Position de l'éclairage
x= 0
y= -1
z= -1
gérer les objets transparents (plus lent si coché)
autoriser la configuration
rotation de la caméra sur X
rotation de la caméra sur Y
rotation de la caméra sur Z
Couleur du fond
Vecteur de gravité
x= 0
y= -19
z= 0
Ces paramètres déterminent le mode utilisé dans IRIS 3D. "autoriser la configuration" doit être coché en mode conception (pour ajouter des objets à la scène et les paramétrer). L'axe de rotation influe sur la navigation 3D.
Autres...
Annuler
OK


IRIS 3D
Vue
Options
Caméra
Ouvrir la fenêtre de configuration
Fermer la fenêtre de configuration
Y
Z
X

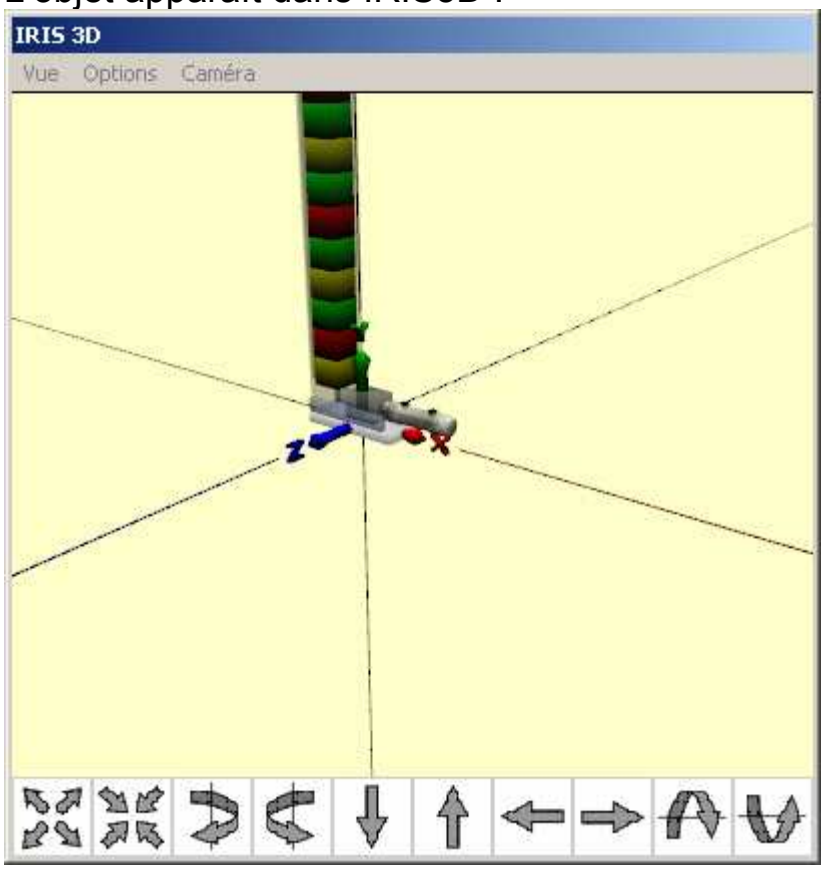


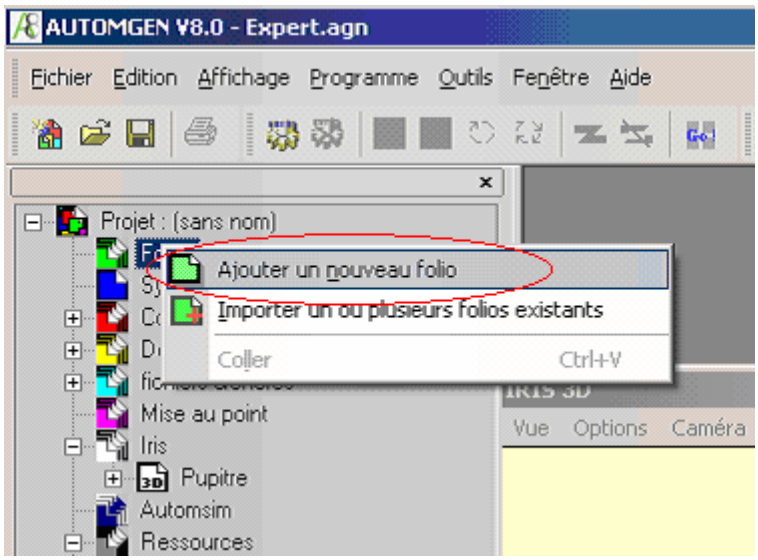


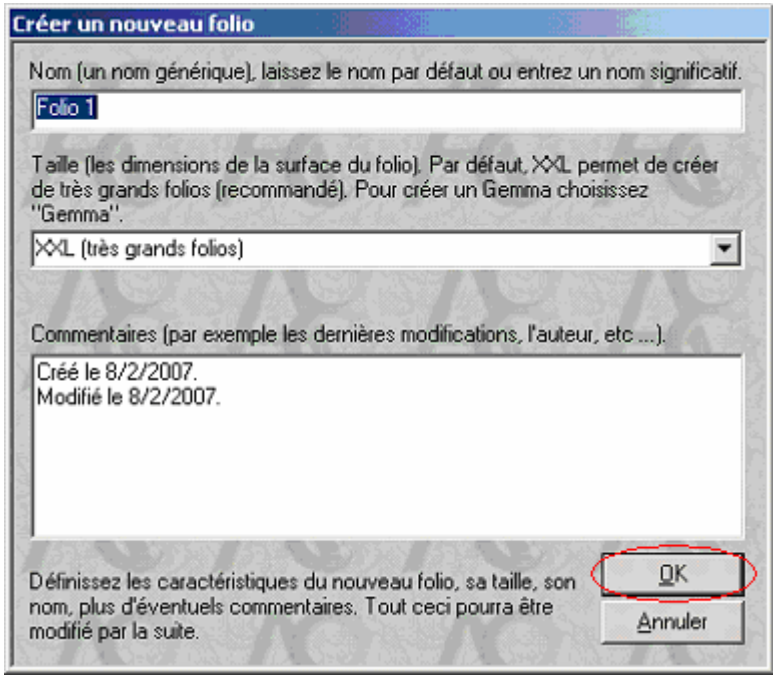
Les objets prédéfinis se trouvent dans le sous répertoire « i3d » du répertoire d'installation d'AUTOMGEN.

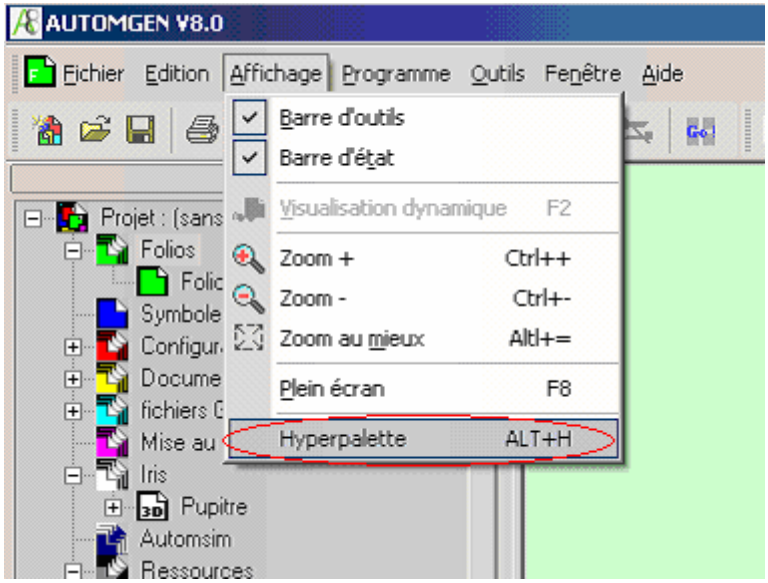


L'objet apparaît dans IRIS3D :

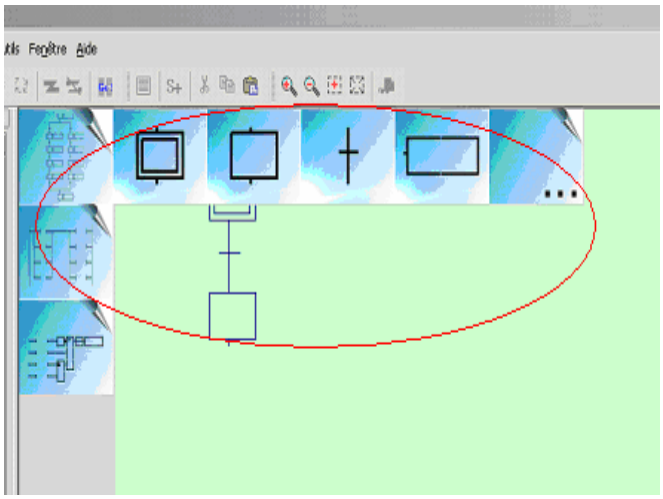




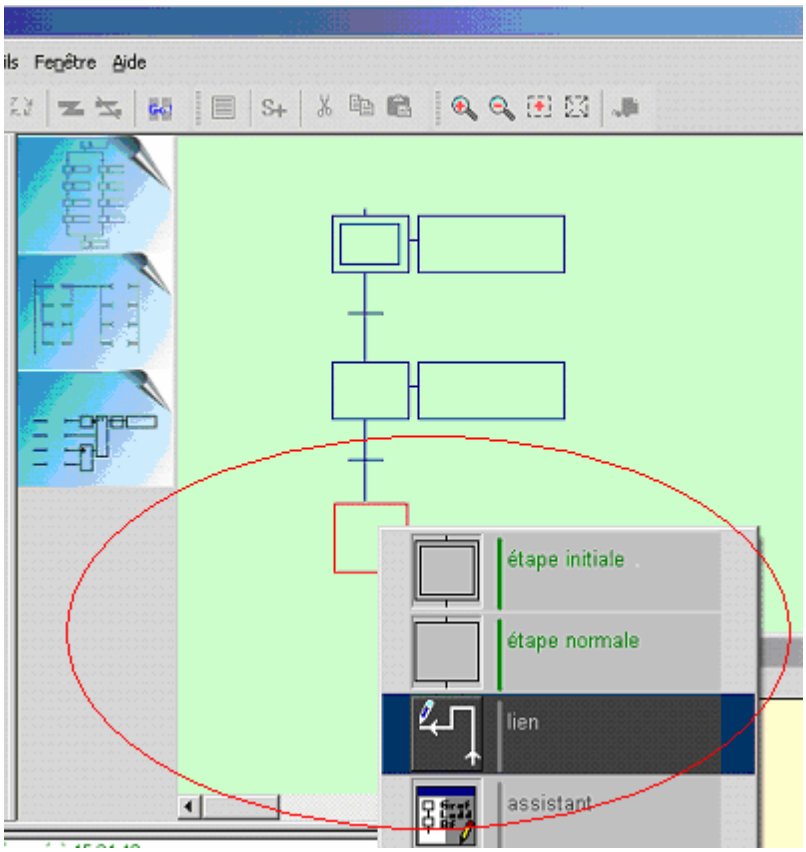




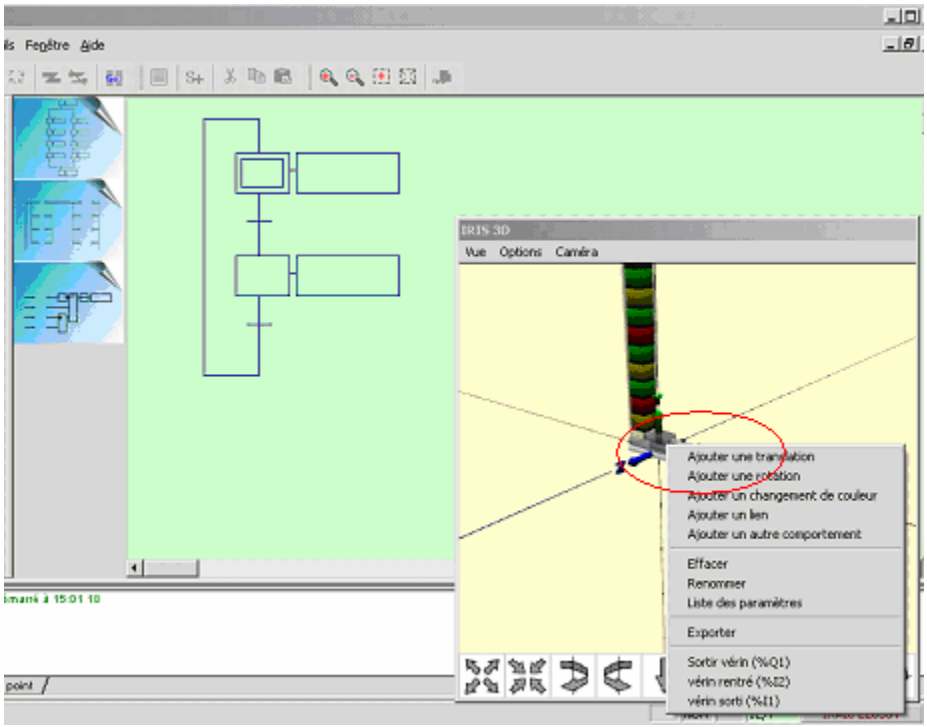
Ceci nous permet de voir comment accéder aux palettes simplifiées à partir du mode « Expert ».

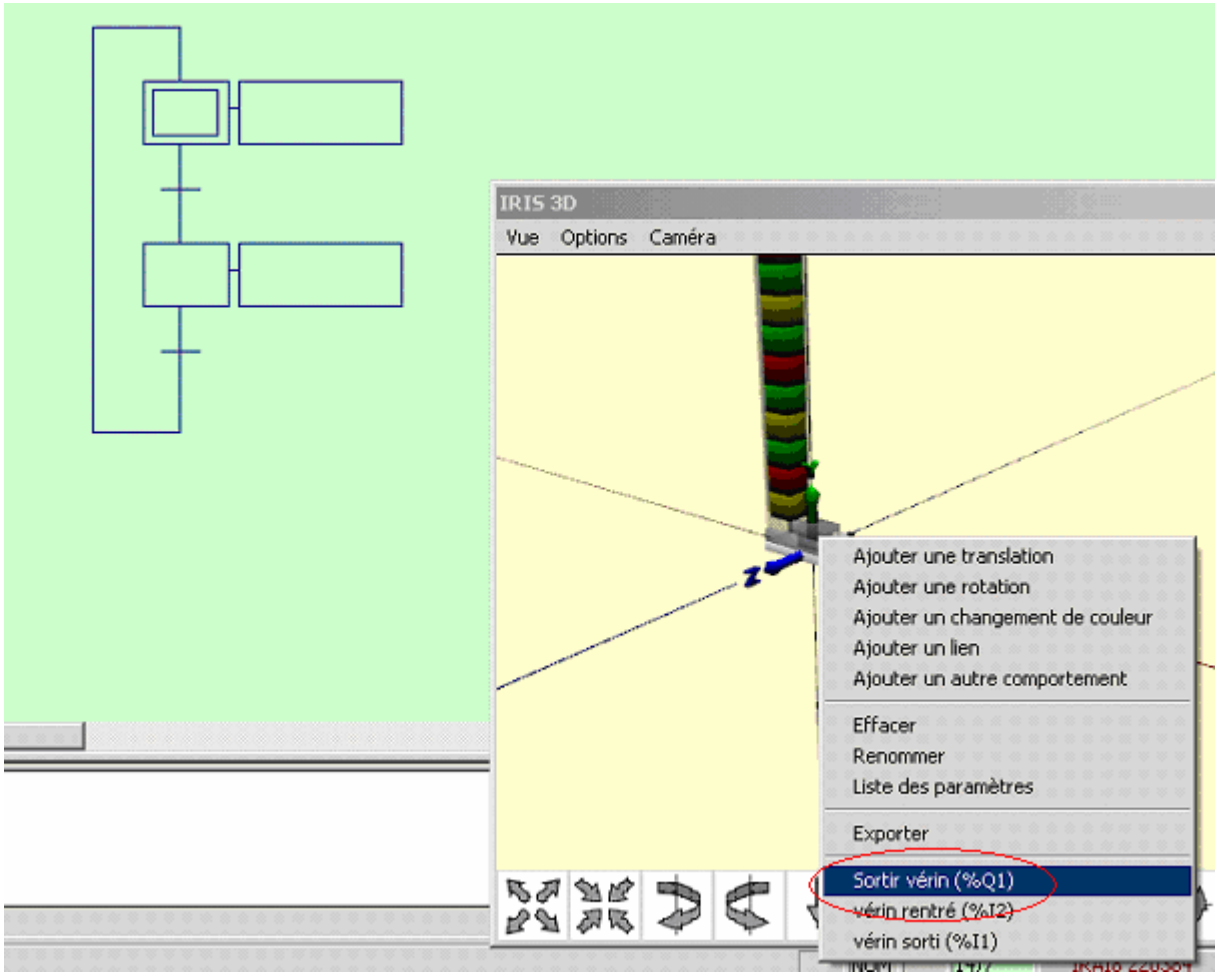


Dessinez un Grafcet à deux étapes avec la palette. Un clic droit sur le folio vous permet d'accéder à la fonction de tracé de lien pour reboucler le Grafcet.

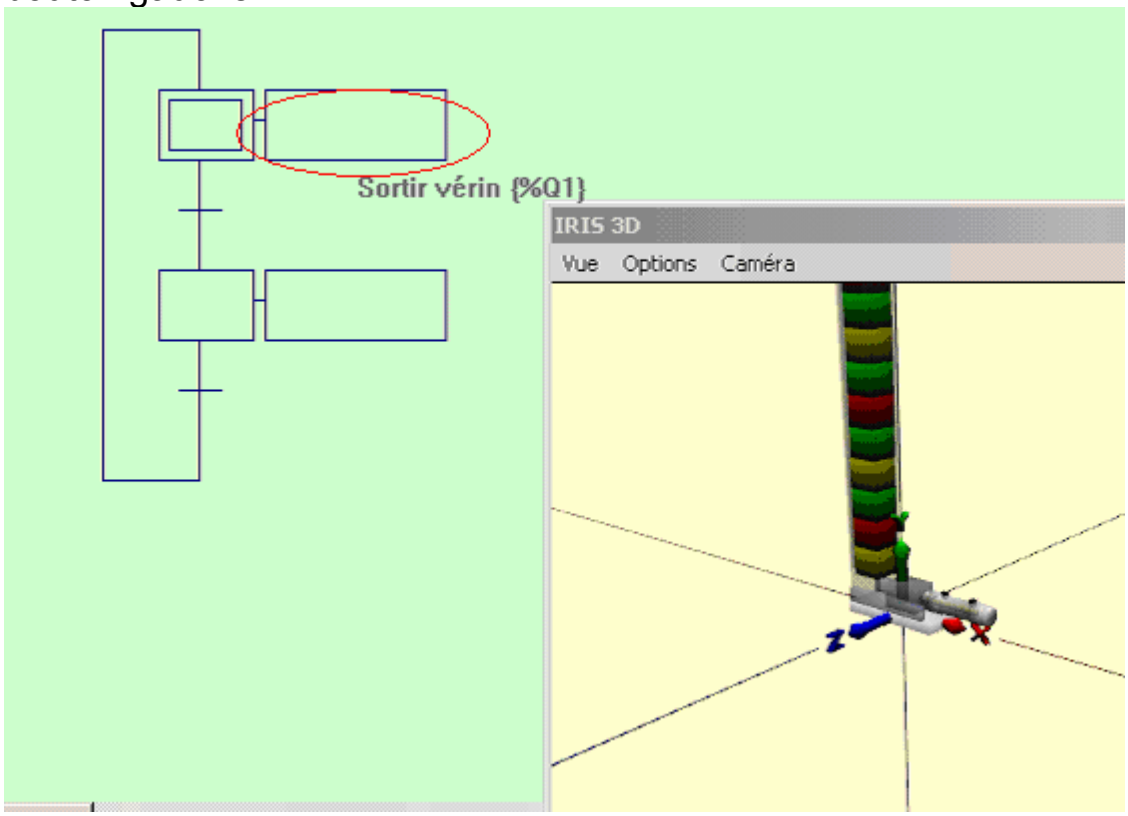


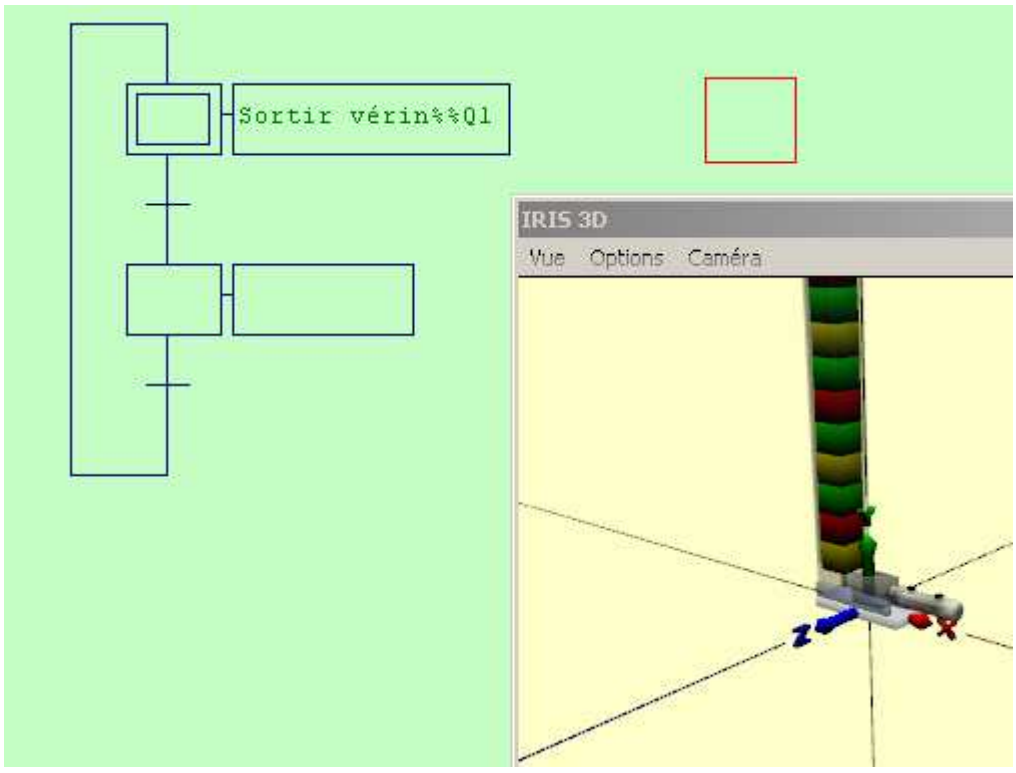
Un clic droit sur le magasin de cubes permet d'accéder à la liste des variables.





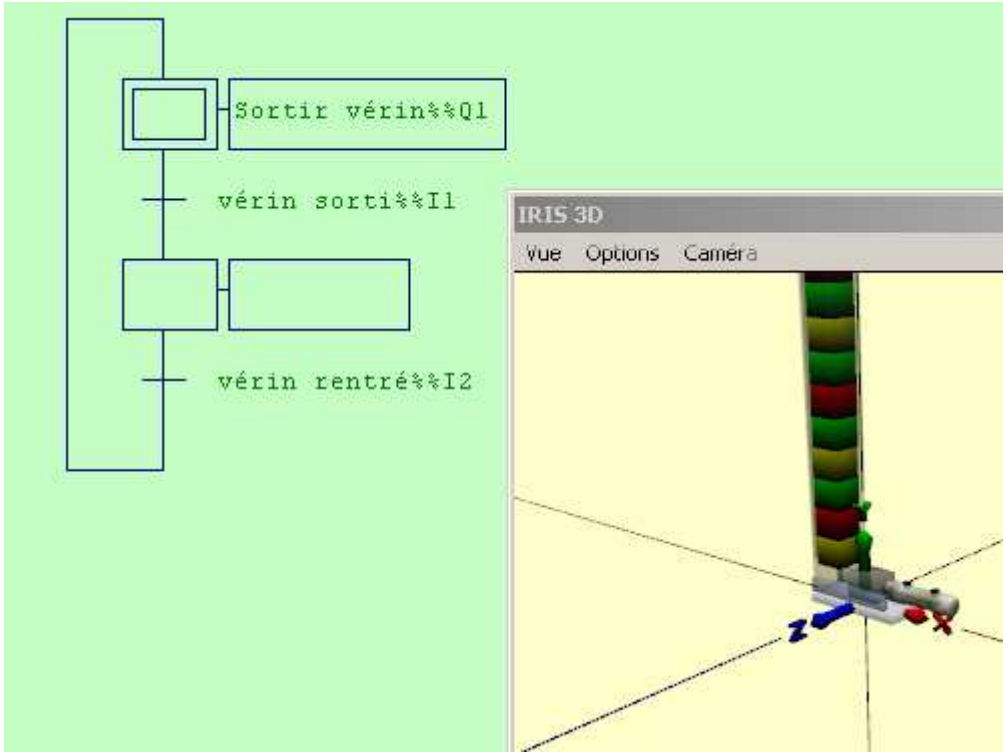
Déplacez le curseur jusque sur le rectangle d'action et cliquez avec le bouton gauche.





Recommencez cette opération pour placer l'élément « vérin sorti » sur la première transition et « vérin rentré sur la deuxième ».

Le résultat final est celui-ci :



Vous pouvez maintenant cliquer sur le bouton « GO » de la barre d'outils pour lancer l'application.

Cet exemple complet se trouve dans le sous répertoire « exemples\simulation PO\3D\moteur physique » et se nomme « déstockeur.agn ».

Appliquer un comportement à un objet

Cliquez avec le bouton droit de la souris sur l'objet dans la fenêtre de configuration d'IRIS 3D et choisissez « Ajouter … » dans le menu.

Nom des variables AUTOMGEN

Les noms de variables AUTOMGEN utilisables dans les comportements sont limités aux syntaxes suivantes :

Accès aux variables booléennes

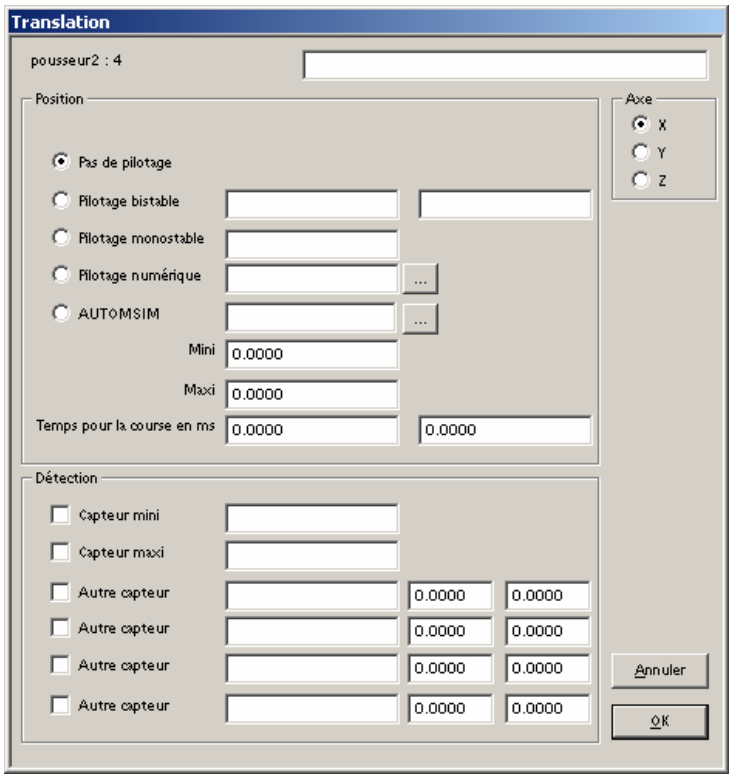
On : sortie « n », par exemple O0, O10,
/On : complément de la sortie « n », par exemple /O1, /O15,
In : entrée « n », par exemple I0, I4,
/In : complément de l'entrée « n », par exemple /I4, /I56,
Bn : bit « n », par exemple B100, B200,
/Bn : complément du bit « n », par exemple /B800, /B100,

L'accès aux bits B est limité à une table de bits linéaire, une directive #B doit être utilisée pour réserver des bits (voir le manuel de référence langage).

Accès aux variables numériques

Mn : mot « n », par exemple : M200, M300
Fn : flottant « n », par exemple : F200,F400

Ajouter une translation



Propriétés d'une translation

Nom

La première zone permet d'entrer un nom générique pour la translation. Ce nom apparaîtra dans la liste de la fenêtre de configuration d'IRIS 3D, il sert uniquement de commentaire et peut être laissé vierge.

Axe

Détermine dans quelle dimension va s'appliquer la translation.

Type

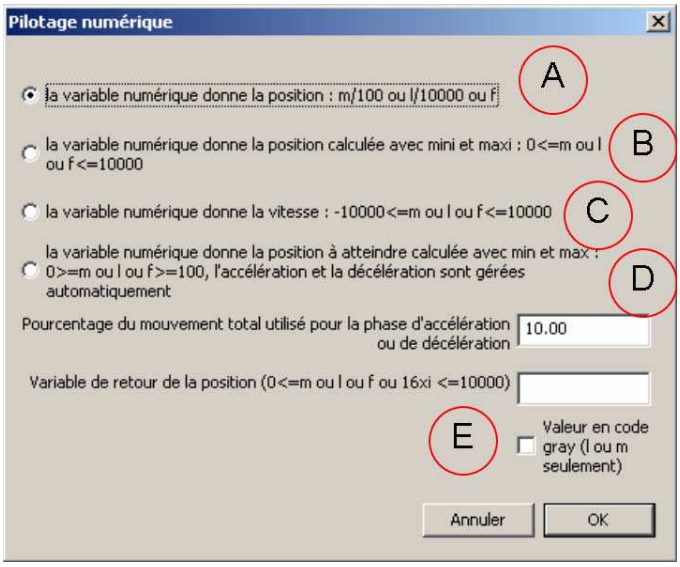
- pas de pilotage : aucune translation, ceci permet de rendre inopérante une translation sans avoir besoin de l'effacer (pour faire des essais par exemple).
- pilotage bistable : deux variables booléennes : la translation est pilotée par deux variables booléennes : la première pilote la translation dans un sens (du mini vers le maxi), la deuxième dans l'autre sens (du maxi vers le mini).

Etat de la première variable	Etat de la deuxième variable	Objet
0	0	Immobile
1	0	Translation du mini vers le maxi
0	1	Translation du maxi vers le mini
1	1	Immobile

- pilotage monostable : une variable booléenne pilote la translation, si la variable est vraie

Etat de la variable	Objet
1	Translation du mini vers le maxi
0	Translation du maxi vers le mini

- pilotage numérique : la position de l'objet sur l'axe désigné est égale au contenu de la variable numérique spécifiée. Le bouton « … » permet de définir un mode « évolué » pour ce type de lien :



A

Le contenu de la variable numérique détermine la position de l'objet. Si c'est un mot la position sera fixée à la valeur divisée par 100, si c'est un long à la valeur divisée par 10000, si c'est un flottant à la valeur contenue dans le flottant. Mini et maxi déterminent les bornes pour ces valeurs.

B

Le contenu de la variable détermine la position entre les valeurs min et max. 0 = position mini, 10000=position maxi.

C

Le contenu de la variable donne une vitesse de déplacement comprise entre -10000 et 10000.

D

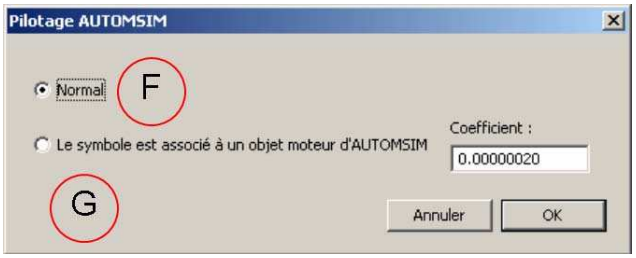
Le contenu de la variable donne une position à atteindre en pourcentage de la course entre mini et maxi : 0=position mini, 100=position maxi. L'accélération et la décélération sont calculées automatiquement. Le paramètre « pourcentage du mouvement utilisé … » détermine la longueur de la phase d'accélération et de décélération.

E

Permet de déterminer une variable qui recevra à chaque instant la position de l'objet. La variable peut être un mot, un long, un flottant ou une entrée (dans ce cas, cette entrée et les 16 suivantes reçoivent la position à la manière d'un codeur absolu connecté sur des entrées). La case à cocher « valeur en code gray » permet d'obtenir cette valeur à la manière d'un codeur gray.

Le sous-répertoire « exemples\Simulation PO\3D\pilotages numériques » contient des exemples illustrant ces différents modes.

- AUTOMSIM : la position de l'objet sur l'axe désigné est donnée par le contenu d'une variable gérée par un objet AUTOMSIM. Le bouton « … » permet de définir un mode évolué pour ce type de lien :



F

La variable associée à un objet AUTOMSIM détermine la position entre mini et maxi.

G

La variable associée à un objet « moteur » d'AUTOMSIM modifie la position en fonction du coefficient (il permet de déterminer le rapport entre la vitesse de rotation du moteur AUTOMSIM et la vitesse de variation de la position).

Le sous-répertoire « exemples\Simulation PO\3D\automsim » contient des exemples illustrant ces deux modes.

Amplitude et origine

Les zones « Mini » et « Maxi » déterminent l'amplitude et l'origine de la translation.

Vitesse

Le temps pour la course détermine la vitesse pour aller du point mini au point maxi (elle est identique à la vitesse de retour).

Détection

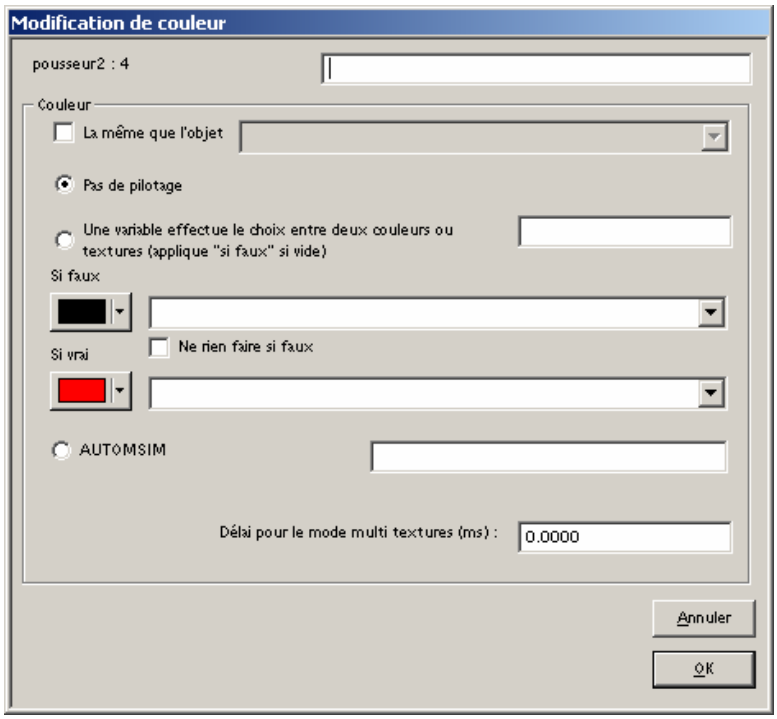
Permet de définir des capteurs pour la translation. Les capteurs mini et maxi gèrent les fins de courses, les 4 autres capteurs possibles peuvent être utilisés pour créer des positions intermédiaires.

Ajouter une rotation

Les paramètres sont en tout point semblables à la translation voir chapitre Ajouter une translation. Les angles sont exprimés en radians.

Le centre de rotation de l'objet doit être réglé pour chaque objet dans la fenêtre de configuration d'IRIS 3D.

Ajouter un changement de couleur ou de texture



Changement de couleur

La case à cocher « La même que l'objet… » permet d'appliquer la même couleur que celle d'un autre objet.

Le pilotage d'une couleur par une variable doit faire référence à une variable booléenne.

Le pilotage de la couleur peut également être réalisé avec une variable AUTOMSIM (associée à un objet voyant d'AUTOMSIM par exemple).

Si la case « ne rien faire si faux » est cochée, aucune couleur n'est appliquée si la variable est à l'état faux. Ceci permet d'associer plusieurs changements de couleur à un même objet si plus de 2 couleurs sont nécessaires.

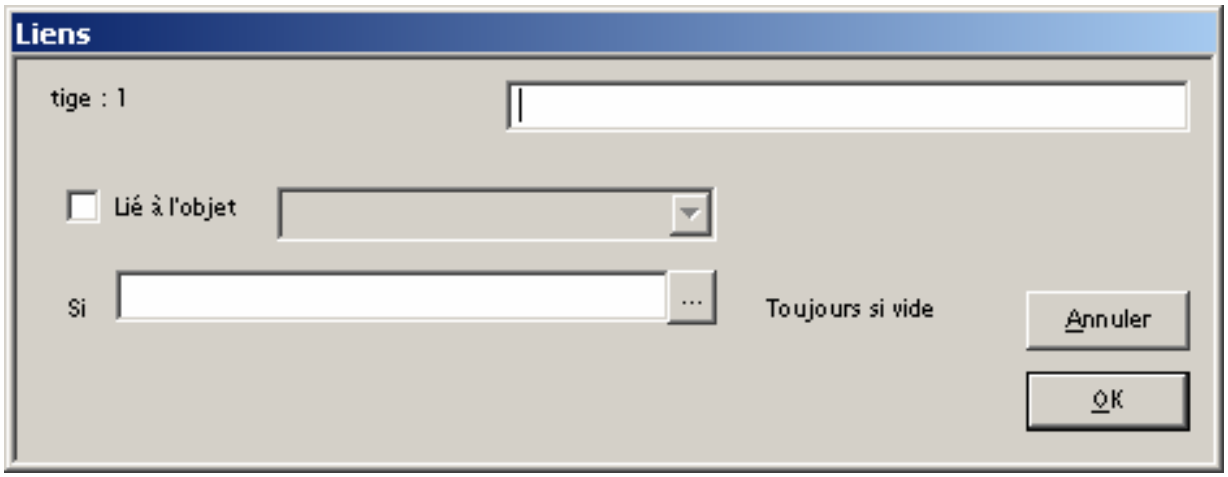
Les listes déroulantes permettent de sélectionner une texture à la place d'une couleur. Pour faire apparaître une texture dans la liste déroulante, placez la (fichier « .bmp » ou « .jpg ») dans les ressources du projet AUTOMGEN.

Textures multiples

Il est possible d'associer plusieurs textures qui seront appliquées automatiquement. Pour ceci, associez plusieurs comportements de type « Modification de couleur » à un même objet et documentez la zone « Délai pour le mode multi texture » avec le temps au bout duquel la texture suivante sera automatiquement appliquée. L'objet prédéfini « Tapis roulant » exploite cette technique.

Ajouter un lien

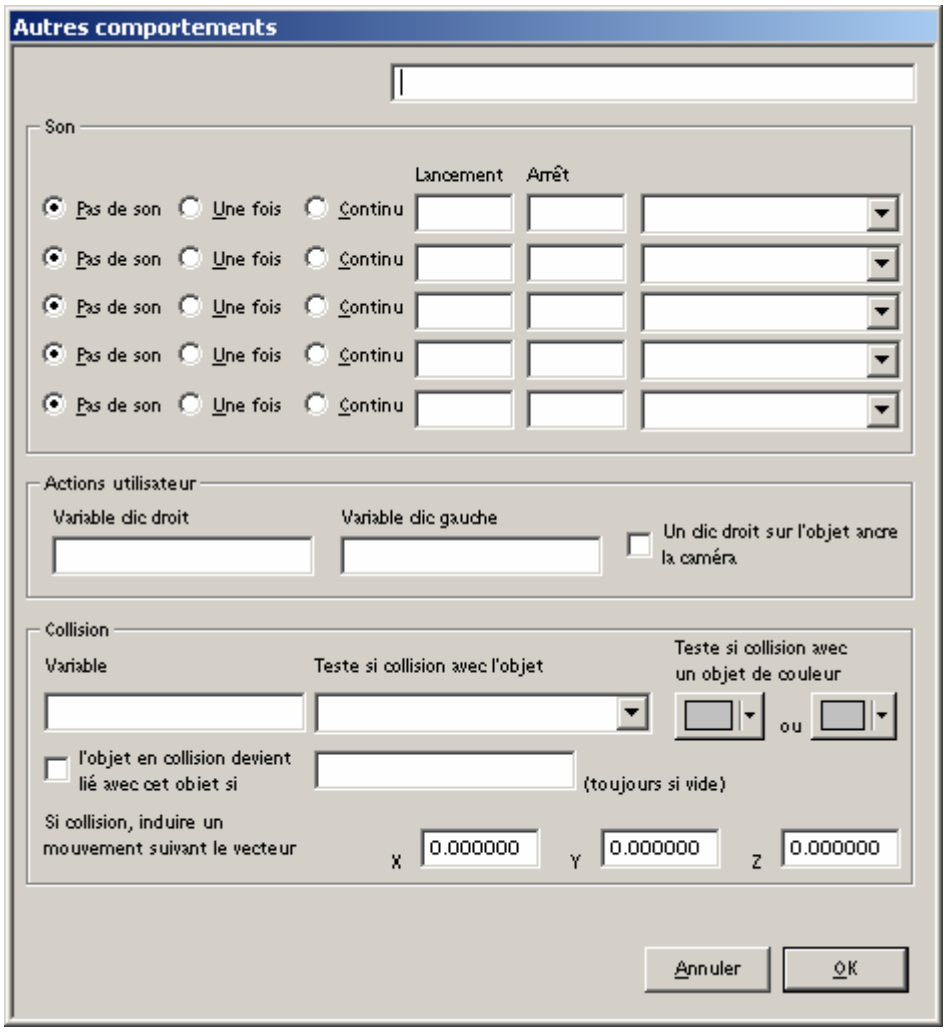
Un lien force l'objet auquel s'applique ce comportement à suivre les déplacements d'un autre objet.



Liens entre objets

La condition du lien peut être une variable booléenne. Le lien est inconditionnel (objet toujours lié) si la condition est laissée vierge.

Ajouter un autre comportement



Autres comportements

Les éléments du groupe « Son » permettent de jouer un son associé à une condition.

Les élément du groupe « Actions utilisateurs » permettent de faire passer à 1 une variable booléenne lorsque l'utilisateur clique avec le bouton droit ou gauche de la souris sur l'objet auquel s'applique le comportement. Le case à cocher « Un clic droit sur l'objet ancre la caméra » permet d'accrocher la caméra (qui définit le point de vue d'affichage dans la fenêtre IRIS 3D) sur l'objet auquel se comportement est appliqué.

Les éléments du groupe « Collision » permettent de définir un test de collision :

- soit avec un objet en particulier,
- soit avec des objets possédant une couleur particulière (choix possible de 2 couleurs).

La zone « Variable » peut être documentée avec un nom de variable booléenne qui passera à l'état vrai si le test de collision est vrai.

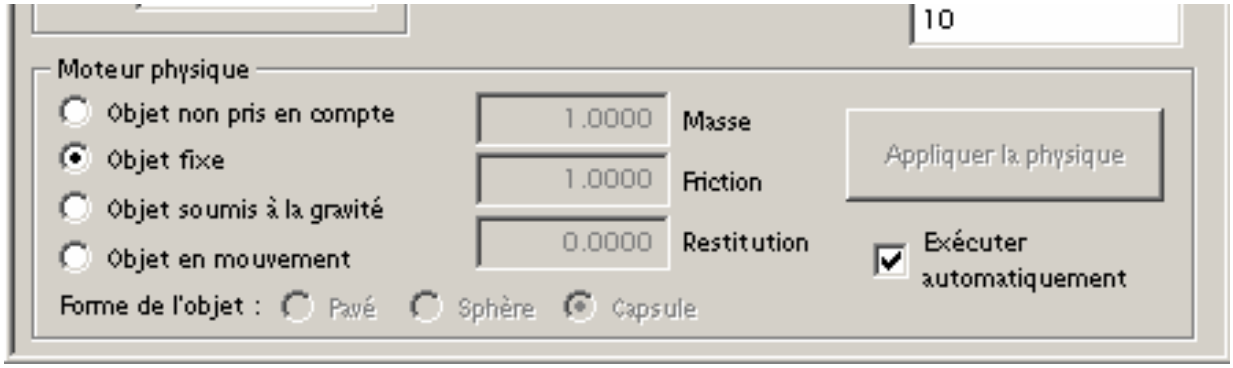
La case « L'objet en collision devient lié avec l'objet si » permet de lier l'objet qui entre en collision avec l'objet auquel est appliqué le comportement. Une variable peut conditionner ce lien. Cette technique permet de gérer facilement la simulation d'une pince ou d'une ventouse.

Le vecteur permet de donner une vitesse à un objet qui entre en collision avec l'objet auquel est appliqué le comportement. L'objet prédéfini « Tapis roulant » utilise cette technique.

Moteur physique

Le moteur physique permet de gérer la pesanteur et les interactions entre objet afin d'obtenir une simulation très réaliste. Pour les objets soumis à la pesanteur, le moteur physique gère uniquement des formes de type pavé, sphère ou capsule.

Pour chaque objet on peut définir un type de gestion utilisé par le moteur physique :



« Objet non pris en compte » : l'objet n'est pas géré par le moteur physique : pas soumis à la pesanteur, pas d'interaction avec les autres objets.

« Objet fixe » : objet géré par le moteur physique ne se déplaçant pas mais ayant des interactions avec les autres objets : le bâti d'une machine par exemple.

« Objet soumis à la gravité » : objets en mouvement géré par le moteur physique, soumis à la pesanteur et ayant des interactions avec les autres objets : une boîte se déplaçant sur un tapis roulant par exemple. Pour ce type d'objet, la masse, les coefficients de frictions et de restitutions ainsi que la forme primaire de l'objet (pavé, sphère ou capsule) sont à déterminer.

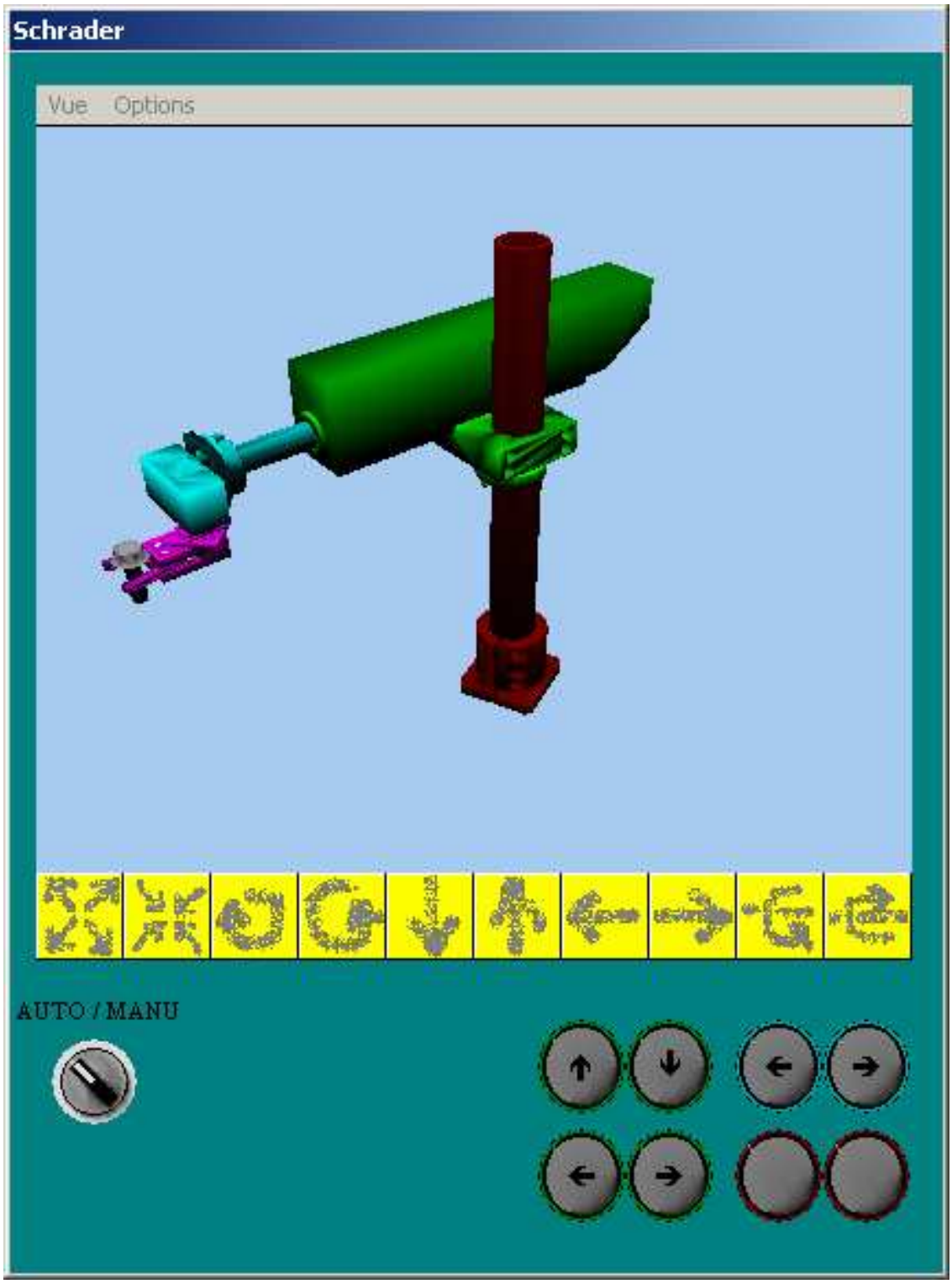
« Objet en mouvement » : objet en mouvement géré par le moteur physique non soumis à la pesanteur et ayant des interactions avec les autres objets : une tige de vérin poussant des objets par exemple. Pour ce type d'objet, les coefficients de frictions et de restitutions ainsi que la forme primaire de l'objet (pavé, sphère ou capsule) sont à déterminer.

Le bouton « Appliquer la physique » permet de lancer le moteur physique. La case à cocher « Exécution automatique » lance automatiquement le moteur physique lorsque l'exécuteur PC d'AUTOMGEN est installé.

Le sous répertoire « exemples\simulation PO\3d\moteur physique » contient des exemples illustrant l'utilisation du moteur physique.

Exemple IRIS 3D

Le sous répertoire « exemples\simulation PO\3d » contient de nombreux exemples. Par exemple :



« Exemples\Simulation PO\3D\Scharder.agn »

Résumé IRIS 3D

IRIS 3D permet de concevoir des applications de simulation de parties opératives en 3D. Les objets peuvent être créés dans un modeleur 3D standard et importés dans les ressources du projet AUTOMGEN, des comportements sont ensuite appliqués aux objets pour créer les animations 3D. Des objets 3D évolués peuvent également être utilisés.

Langages

Eléments communs

Ce chapitre détaille les éléments communs à tous les langages utilisables dans AUTOMGEN. AUTOMGEN utilise les éléments des normes CEI1131-3 et 60848. Des éléments de syntaxe assurent également une compatibilité avec les anciennes versions d'AUTOMGEN. Enfin, des extensions à la norme nommées Grafcet++ donnent accès à des fonctions évoluées et sont une exclusivité proposée dans AUTOMGEN.

Les variables

Les types de variables suivants existent :

⇨ le type booléen : la variable peut prendre la valeur vrai (1) ou faux (0).

⇨ le type numérique : la variable peut prendre une valeur numérique, différents sous types existent : variables 16 bits, 32 bits et virgule flottante.

⇨ le type temporisation : type structuré, il est une combinaison du type booléen et numérique.

La syntaxe des noms de variables peut être celle propre à AUTOMGEN ou la syntaxe de la norme CEI 1131-3.

Les variables booléennes

Le tableau suivant donne la liste exhaustive des variables booléennes utilisables.

Type	Syntaxe AUTOMGEN	Syntaxe CEI 1131-3	Commentaire
Entrées	I0 à I9999	%I0 à %I9999	Peut correspondre ou non à des entrées physiques (dépend de la configuration des E/S de la cible).
Sorties	O0 à O9999	%Q0 à %Q9999	Peut correspondre ou non à des sorties physiques (dépend de la configuration des E/S de la cible).
Bits Système	U0 à U99	%M0 à %M99	Voir le manuel consacré à l'environnement pour le détail des bits Système.
Bits Utilisateur	U100 à U9999	%M100 à %M9999	Bits internes à usage général.

Etapes Grafcet	X0 à X9999	%X0 à %X9999	Bits d'étapes Grafcet.
Bits de mots	M0#0 à M9999#15	%MW0 :X0 à %MW9999 :X15	Bits de mots : le numéro du bit est précisé en décimal et est compris entre 0 (bit de poids faible) et 15 (bit de poids fort).

Les variables numériques

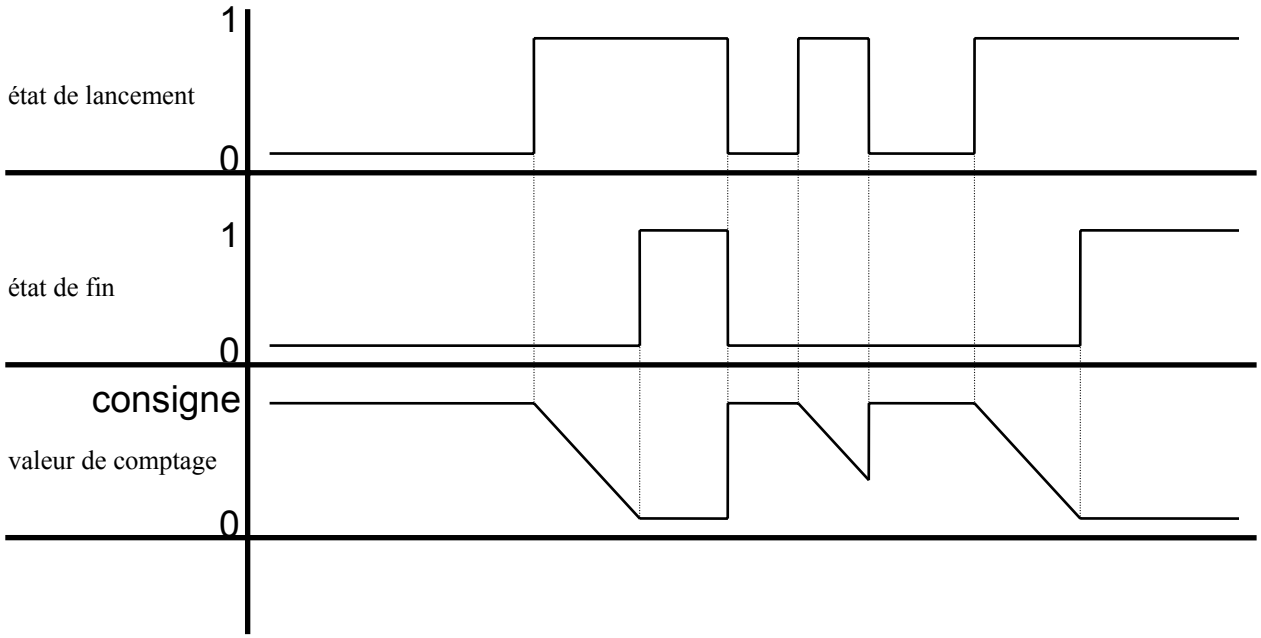
Le tableau suivant donne la liste exhaustive des variables numériques.

Type	Syntaxe AUTOMGEN	*Syntaxe CEI 1131-3*	Commentaire
Compteurs	C0 à C9999	%C0 à %C9999	Compteur de 16 bits, peut être initialisé, incrémenté, décrémenté et testé avec les langages booléens sans utiliser le langage littéral.
Mots Système	M0 à M199	%MW0 à %MW199	Voir le manuel consacré à l'environnement pour le détail des mots Système.
Mots Utilisateur	M200 à M9999	%MW200 à %MW9999	Mot de 16 bits à usage général.
Longs	L100 à L4998	%MD100 à %MD4998	Valeur entière sur 32 bits.
Flottants	F100 à F4998	%MF100 à %MF4998	Valeur réelle sur 32 bits (format IEEE).

Les temporisations

La temporisation est un type composé qui regroupe deux variables booléennes (état de lancement, état de fin) et deux variables numériques sur 32 bits (la consigne et le compteur).

Le schéma suivant donne le chronogramme de fonctionnement d'une temporisation :



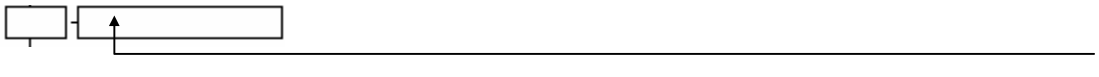
La valeur de consigne d'une temporisation est comprise entre 0 ms et 4294967295 ms (soit un peu plus de 49 jours)

La consigne de la temporisation peut être modifiée par programme, voir STASTASTASTAchapitres : Le langage littéral bas niveau STASTA(instruction STA)
Ecriture de la consigne d'une temporisation
Le compteur de la temporisation peut être lu par programme, voir chapitres :
(instruction LDA)
Lecture du compteur d'une temporisation

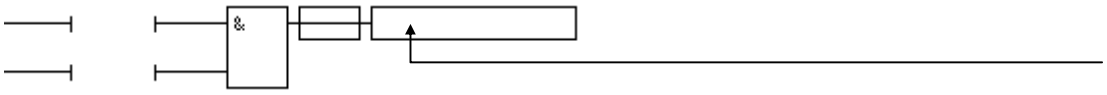
Les actions

Les actions sont utilisées dans :

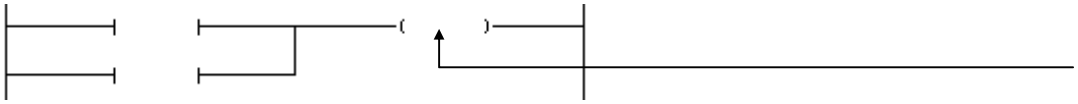
⇨ les rectangles d'action du langage Grafcet,



⇨ les rectangles d'action du langage logigramme,



⇨ les bobines du langage ladder.



Affectation d'une variable booléenne

La syntaxe de l'action « Affectation » est :
«variable booléenne»
Fonctionnement :

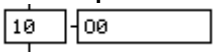
⇨ si la commande du rectangle d'action ou de la bobine est à l'état vrai alors la variable est mise à 1 (état vrai),

⇨ si la commande du rectangle d'action ou de la bobine est à l'état faux alors la variable est mise à 0 (état faux).

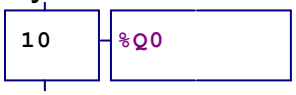
Table de vérité :

Commande	Etat de la variable (résultat)
0	0
1	1

Exemple :



Syntaxe CEI1131-3 :

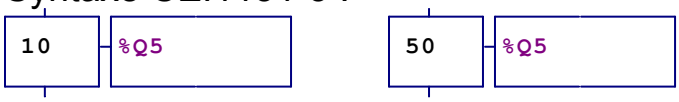


Si l'étape 10 est active alors O0 prend la valeur 1, sinon O0 prend la valeur 0.

Plusieurs actions « Affectation » peuvent être utilisées pour une même variable au sein d'un programme. Dans ce cas, les différentes commandes sont combinées en « Ou » logique.

Exemple :

Syntaxe CEI1131-3 :



Etat de X10	Etat de X50	Etat de O5
0	0	0
1	0	1
0	1	1
1	1	1

Affectation complémentée d'une variable booléenne

La syntaxe de l'action « Affectation complémentée » est :
«N variable booléenne»
Fonctionnement :

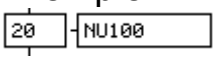
⇨ si la commande du rectangle d'action ou de la bobine est à l'état vrai alors la variable est mise à 0 (état faux),

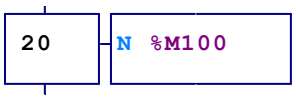
⇨ si la commande du rectangle d'action ou de la bobine est à l'état faux alors la variable est mise à 1 (état vrai).

Table de vérité :

Commande	Etat de la variable (résultat)
0	1
1	0

Exemple :





Si l'étape 20 est active, alors U100 prend la valeur 0, sinon U100 prend la valeur 1.
Plusieurs actions « Affectation complémentée» peuvent être utilisées pour une même variable au sein d'un programme. Dans ce cas, les différentes commandes sont combinées en « Ou » logique.

Exemple :

Syntaxe CEI1131-3 :

Etat de X100	Etat de X110	Etat de O20
0	0	1
1	0	0
0	1	0
1	1	0

Mise à un d'une variable booléenne

La syntaxe de l'action « Mise à un » est :
«S variable booléenne» ou « variable booléenne :=1 »
Fonctionnement :

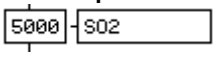
⇨ si la commande du rectangle d'action ou de la bobine est à l'état vrai alors la variable est mise à 1 (état vrai),

⇨ si la commande du rectangle d'action ou de la bobine est à l'état faux alors l'état de la variable n'est pas modifié.

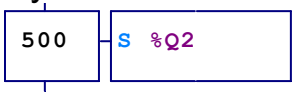
Table de vérité :

Commande	Etat de la variable (résultat)
0	inchangé
1	1

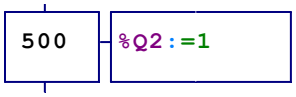
Exemple :



Syntaxe CEI1131-3 :



ou



Si l'étape 5000 est active alors O2 prend la valeur 1, sinon O2 garde son état.

Mise à zéro d'une variable booléenne

La syntaxe de l'action « Mise à zéro » est :
«R variable booléenne» ou « variable booléenne :=0 »
Fonctionnement :

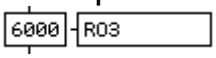
⇨ si la commande du rectangle d'action ou de la bobine est à l'état vrai alors la variable est mise à 0 (état faux),

⇨ si la commande du rectangle d'action ou de la bobine est à l'état faux alors l'état de la variable n'est pas modifié.

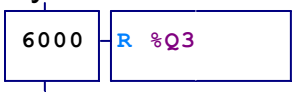
Table de vérité :

Commande	Etat de la variable (résultat)
0	inchangé
1	0

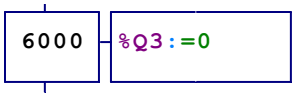
Exemple :



Syntaxe CEI1131-3 :



ou



Si l'étape 6000 est active alors O3 prend la valeur 0, sinon O3 garde son état.

Inversion d'une variable booléenne

La syntaxe de l'action « Inversion » est :
«I variable booléenne»
Fonctionnement :

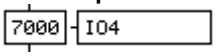
⇨ si la commande du rectangle d'action ou de la bobine est à l'état vrai alors l'état de la variable est inversé à chaque cycle d'exécution,

⇨ si la commande du rectangle d'action ou de la bobine est à l'état faux alors l'état de la variable n'est pas modifié.

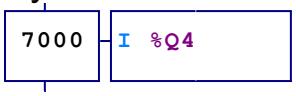
Table de vérité :

Commande	Etat de la variable (résultat)
0	inchangé
1	inversé

Exemple :



Syntaxe CEI1131-3 :



Si l'étape 7000 est active alors l'état de O4 est inversé, sinon O4 garde son état.

Mise à zéro d'un compteur, d'un mot ou d'un long

La syntaxe de l'action « Mise à zéro d'un compteur, d'un mot ou d'un long» est :
«R compteur, mot ou long»
Fonctionnement :

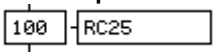
⇨ si la commande du rectangle d'action ou de la bobine est à l'état vrai alors le compteur, le mot ou le long est remis à zéro,

⇨ si la commande du rectangle d'action ou de la bobine est à l'état faux la valeur du compteur, du mot, ou du long n'est pas modifiée.

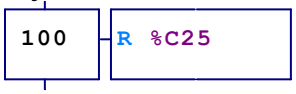
Table de vérité :

Commande	Valeur du compteur du mot ou du long (résultat)
0	Inchangé
1	0

Exemple :



Syntaxe CEI1131-3 :



Si l'étape 100 est active alors le compteur 25 est remis à zéro, sinon C25 garde sa valeur.

Incrémentation d'un compteur, d'un mot ou d'un long

La syntaxe de l'action « Incrémentation d'un compteur » est :
«+ compteur, mot ou long»
Fonctionnement :

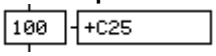
⇨ si la commande du rectangle d'action ou de la bobine est à l'état vrai alors le compteur, le mot ou le long est incrémenté à chaque cycle d'exécution,

⇨ si la commande du rectangle d'action ou de la bobine est à l'état faux la valeur du compteur n'est pas modifiée.

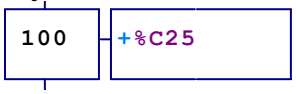
Table de vérité :

Commande	Valeur du compteur, du mot ou du long (résultat)
0	Inchangé
1	valeur actuelle +1

Exemple :

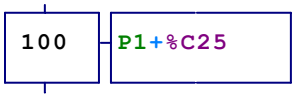


Syntaxe CEI1131-3 :



Si l'étape 100 est active alors le compteur 25 est incrémenté, sinon C25 garde sa valeur. L'incrémentation est réalisée à chaque cycle tant

que l'étape est active. Pour réaliser l'incrémentation une seule fois sur activation de l'étape (ce qui est généralement le cas le plus fréquent), le modificateur d'action P1 peut être utilisé. Par exemple :



Décrémentation d'un compteur, d'un mot ou d'un long

La syntaxe de l'action « Décrémentation d'un compteur » est :
«- compteur, mot ou long»
Fonctionnement :

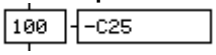
⇨ si la commande du rectangle d'action ou de la bobine est à l'état vrai alors le compteur, le mot ou le long est décrémenté à chaque cycle d'exécution,

⇨ si la commande du rectangle d'action ou de la bobine est à l'état faux la valeur du compteur n'est pas modifiée.

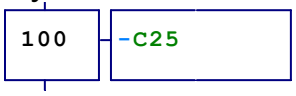
Table de vérité :

Commande	Valeur du compteur, du mot ou du long (résultat)
0	inchangé
1	valeur actuelle -1

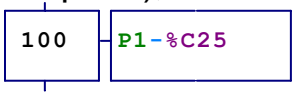
Exemple :



Syntaxe CEI1131-3 :



Si l'étape 100 est active alors le compteur 25 est decrémenté, sinon C25 garde sa valeur. La décrémentation est réalisée à chaque cycle tant que l'étape est active. Pour réaliser la décrémentation une seule fois sur activation de l'étape (ce qui est généralement le cas le plus fréquent), le modificateur d'action P1 peut être utilisé. Par exemple :



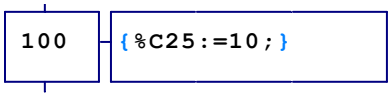
Placer une constante dans un compteur, un mot ou un long

La syntaxe de l'action « Placer une constante dans un compteur, un mot ou un long» est :
« {compteur, mot ou long :=constante ;}»
Fonctionnement :

⇨ si la commande du rectangle d'action ou de la bobine est à l'état vrai alors le compteur, le mot ou le long reçoit la constante,

⇨ si la commande du rectangle d'action ou de la bobine est à l'état faux la valeur du compteur, du mot, ou du long n'est pas modifiée.

Exemple :



Temporisations

Les temporisations sont considérées comme des variables booléennes et sont utilisables avec les actions « Affectation », « Affectation complémentée », « Mise à un », « Mise à zéro », et « Inversion ». La consigne de la temporisation peut être écrite à la suite de l'action. La syntaxe est :
« temporisation(durée) »
La durée est par défaut exprimée en dixièmes de seconde. Le caractère « S » placé à la fin de la durée indique qu'elle est exprimée en secondes.

Exemples :

Syntaxe CEI1131-3 :

L'étape 10 lance une temporisation de 2 secondes qui restera active tant que l'étape le sera. L'étape 20 arme une temporisation de 6 secondes qui restera active même si l'étape 20 est désactivée.
Une même temporisation peut être utilisée à plusieurs endroits avec une même consigne et à des instants différents. La consigne de la temporisation doit dans ce cas être indiquée une seule fois.

Remarque : d'autres syntaxes existent pour les temporisations. Voir chapitre Temporisations

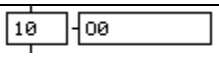
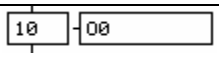
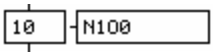
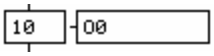
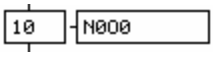
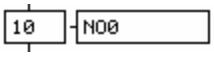
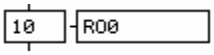
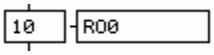
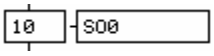
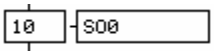
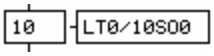
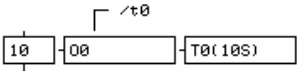
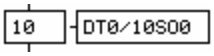
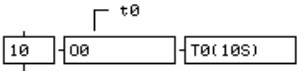
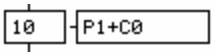
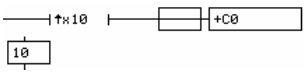
Interférences entre les actions

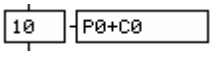
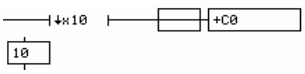
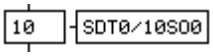
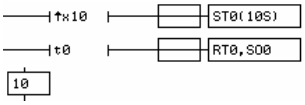
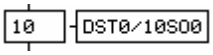
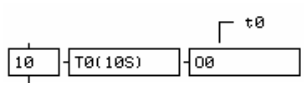
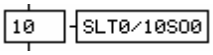
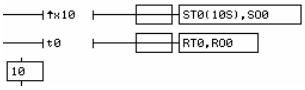
Certains types d'action ne peuvent être utilisés simultanément sur une variable. Le tableau ci-dessous résume les combinaisons interdites.

	Affectation	**Affectation complémentée**	**Mise à un**	**Mise à zéro**	**Inversion**
Affectation	OUI	NON	NON	NON	NON
Affectation complémentée	NON	OUI	NON	NON	NON
Mise à un	NON	NON	OUI	OUI	OUI
Mise à zéro	NON	NON	OUI	OUI	OUI
Inversion	NON	NON	OUI	OUI	OUI

Actions de la norme CEI 1131-3

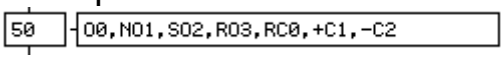
Le tableau ci-dessous donne la liste des actions de la norme CEI 1131-3 utilisables dans AUTOMGEN.

Nom	*Syntaxe CEI 1131-3*	*Exemple*	*Exemple avec Equivalent syntaxe AUTOMGEN*
Non mémorisé	Néant		
Non mémorisé	N1		
Non mémorisé complémenté	N0		
Mise à zéro	R		
Mise à 1	S		
Limité dans le temps	LTn/durée		
Temporisé	DTn/durée		
Impulsion sur front montant	P1		

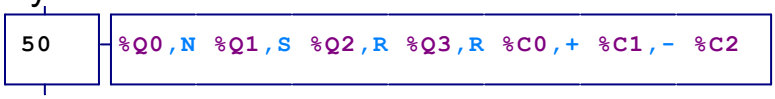
Impulsion sur front descendant	P0		
Mémorisé et temporisé	SDTn/durée		
Temporisé et mémorisé	DSTn/durée		
Mémorisé et limité dans le temps	SLTn/durée		

Actions multiples

Au sein d'un même rectangle d'action ou d'une bobine, plusieurs actions peuvent être écrites en les séparant par le caractère « , » (virgule).
Exemple :



Syntaxe CEI1131-3 :



Plusieurs rectangles d'action (Grafcet et logigramme) ou bobines (ladder) peuvent être juxtaposés. Reportez-vous aux chapitres correspondant à ces langages pour plus de détails.

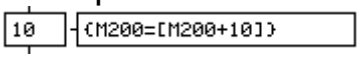
Code littéral

Du code littéral peut être inséré dans un rectangle d'action ou une bobine.

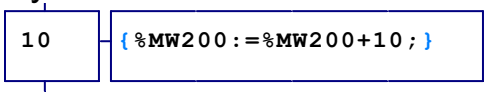
La syntaxe est :
« { code littéral } »

Plusieurs lignes de langage littéral peuvent être écrites entre les accolades. Le séparateur est ici aussi le caractère « , » (virgule).

Exemple :



Syntaxe CEI1131-3 :

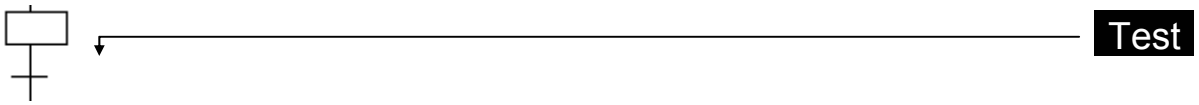


Pour plus de détails, consultez les chapitres « Langage littéral bas niveau », « Langage littéral étendu » et « Langage littéral ST ».

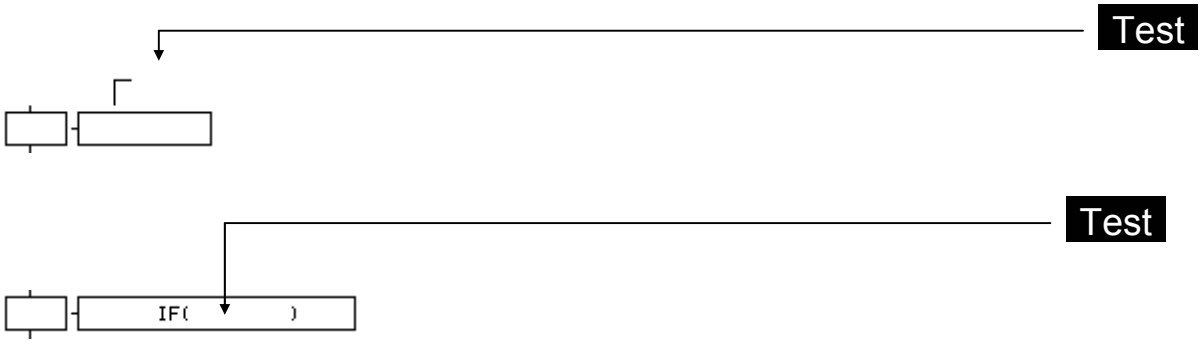
Les tests

Les tests sont utilisés dans :

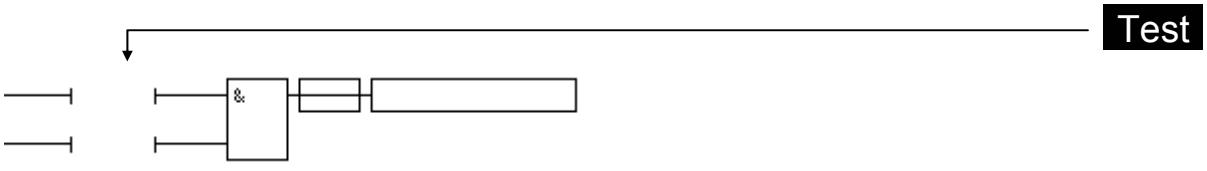
⇨ les transitions du langage Grafcet,



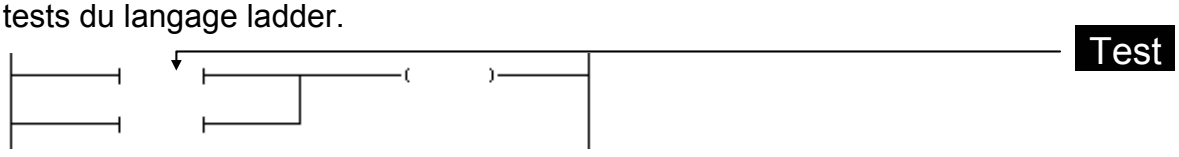
⇨ les conditions sur action du langage Grafcet,



⇨ les tests du langage logigramme,



⇨ les tests du langage ladder.



Forme générale

Un test est une équation booléenne composée de une ou de n variables séparées par des opérateurs « + » (ou) ou « . » (et).

Exemple de test :

```
i0 (test l'entrée 0)
i0+i2 (test l'entrée 0 « ou » l'entrée 2)
i10.i11 (test l'entrée 10 « et » l'entrée 11)
```


Modificateur de test

Par défaut, si seul le nom d'une variable est spécifié, le test est « si égal à un» (si vrai). Des modificateurs permettent de tester l'état complémenté, le front montant et le front descendant :

⇨ le caractère « / » placé devant une variable teste l'état complémenté,

⇨ le caractère « u » ou le caractère « ↑* » placé devant une variable teste le front montant,

⇨ le caractère « d » ou le caractère « ↓** » placé devant une variable teste le front descendant.

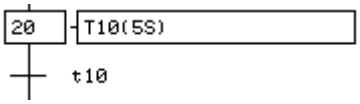
Les modificateurs de tests peuvent s'appliquer à une variable ou à une expression entre parenthèses.

Exemples :

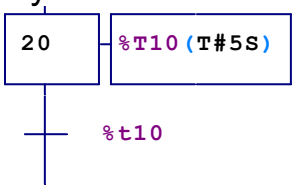
```
↑ i0
/i1
/(i2+i3)
↓(i2+(i4./i5))
```

Temporisations

Quatre syntaxes sont disponibles pour les temporisations.
Dans la première, on active la temporisation dans l'action et on mentionne simplement la variable temporisation dans un test pour vérifier l'état de fin :



Syntaxe CEI1131-3 :



Pour les autres, tout est écrit dans le test. La forme générale est :
« temporisation / variable de lancement / durée »
ou
« durée / variable de lancement / temporisation »
ou
« durée / variable de lancement »

---

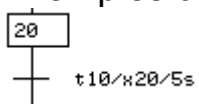
* Pour obtenir ce caractère pendant l'édition d'un test pressez sur la touche [↑].

** Pour obtenir ce caractère pendant l'édition d'un test pressez sur la touche [↓].

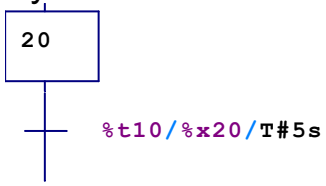
Dans ce cas, une temporisation est automatiquement attribuée. La plage d'attribution est celle des symboles automatiques voir chapitre Symboles automatiques.

La durée est par défaut exprimée en dixièmes de seconde.
La durée peut être exprimée en jours, heures, minutes, secondes et millisecondes avec les opérateurs « d », « h », « m », « s » et « ms ». Par exemple : 1d30s = 1 jour et 30 secondes.

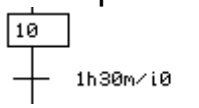
Exemples utilisant la deuxième syntaxe :



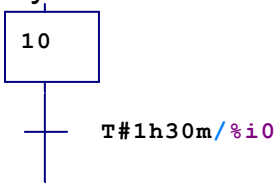
Syntaxe CEI1131-3 :



Exemple utilisant la syntaxe normalisée :



Syntaxe CEI1131-3 :



Priorité des opérateurs booléens

Par défaut l'opérateur booléen « . » (ET) a une priorité supérieure à l'opérateur « + » (OU). Des parenthèses peuvent être utilisées pour définir une autre priorité.
Exemples :

```
i0.(i1+i2)
((i0+i1).i2)+i5
```

Test toujours vrai

La syntaxe du test toujours vrai est :
« » (néant) ou « =1 »

Test sur variable numérique

Les tests sur variable numérique doivent utiliser la syntaxe suivante :

« variable numérique » « type de test » « constante ou variable numérique »

ou
[(« variable numérique » « type de test » « constante ou variable numérique »]

Le type de test peut être :

⇨ « = » égal,

⇨ « ! » ou « <> » différent,

⇨ « < » inférieur (non signé),

⇨ « > » supérieur (non signé),

⇨ « << » inférieur (signé),

⇨ « >> » supérieur (signé),

⇨ « <= » inférieur ou égal (non signé),

⇨ « >= » supérieur ou égal (non signé),

⇨ « <<= » inférieur ou égal (signé),

⇨ « >>= » supérieur ou égal (signé).

Un flottant ne peut être comparé qu'avec un autre flottant ou une constante réelle.
Un long ne peut être comparé qu'avec un autre long ou une constante longue.
Un mot ou un compteur ne peut être comparé qu'avec un mot, un compteur ou une constante 16 bits.
Les constantes réelles doivent être suivies du caractère « R ».
Les constantes longues (32 bits) doivent être suivies du caractère « L ».
Les constantes entières 16 ou 32 bits sont écrites en décimal par défaut. Elles peuvent être écrites en hexadécimal (suffixe « $ » ou « 16# ») ou en binaire (suffixe « % » ou « 2# »).
Les tests sur variables numériques sont utilisés dans les équations comme les tests sur variables booléennes. Ils peuvent être utilisés avec les modificateurs de test à condition d'être encadrés par des parenthèses.

Exemples :

```
m200=100
%mw1000=16#abcd
c10>20.c10<100
f200=f201
[m200=m203]
```



```
%md100=%md102
f200=3.14r
l200=$12345678L
m200<<-100
m200>>1000
[%mw500<=12]
/(m200=4)
🡻(m200=100)
/(l200=100000+l200=-100000)
```

Transitions sur plusieurs lignes

Le texte des transitions peut être étendu sur plusieurs lignes. La fin d'une ligne de transition doit obligatoirement être un opérateur « . » ou « + ». Les combinaisons de touche [CTRL] + [🡻] et [CTRL] + [🡹] permettent de déplacer le curseur d'une ligne à l'autre.

Utilisation de symboles

Les symboles permettent d'associer un texte à une variable.
Les symboles peuvent être utilisés avec tous les langages.
Un symbole doit être associé à une et une seule variable.

Syntaxe des symboles

Les symboles sont composés de :

⇨ un caractère « _ » optionnel (souligné, généralement associé à la touche [8] sur les claviers) qui marque le début du symbole,

⇨ le nom du symbole,

⇨ un caractère « _ » optionnel (souligné) qui marque la fin du symbole.

Les caractères « _ » encadrant les noms de symboles sont optionnels. Il doivent être utilisés si le symbole commence par un chiffre ou un opérateur (+,-, etc…).

Symboles automatiques

Il est parfois fastidieux de devoir définir l'attribution entre chaque symbole et une variable, notamment si l'attribution précise d'un numéro de variable importe peu. Les symboles automatiques sont une solution à ce problème, ils permettent de laisser le soin au compilateur de générer automatiquement l'attribution d'un symbole à un numéro de variable. Le type de variable à utiliser est, quant à lui, fourni dans le nom du symbole.

Syntaxe des symboles automatiques

La syntaxe des symboles automatiques est la suivante :

```
_« nom du symbole » %« type de variable »_
```


« type de variable » peut être :
I , O ou Q, U ou M, T, C, M ou MW, L ou MD, F ou MF.

Comment le compilateur gère-t-il les symboles automatiques ?

Au début de la compilation d'une application, le compilateur efface tous les symboles automatiques qui se trouvent dans le fichier « .SYM » de l'application. A chaque fois que le compilateur rencontre un symbole automatique, il crée une attribution unique pour ce symbole en fonction du type de variable spécifié dans le nom du symbole. Le symbole ainsi généré, est écrit dans le fichier « .SYM ». Si un même symbole automatique est présent plusieurs fois dans une application, il fera référence à la même variable.

Plage d'attribution des variables

Par défaut, une plage d'attribution est définie pour chaque type de variables :

Type	Début	Fin
I ou %I	0	9999
O ou %Q	0	9999
U ou %M	100	9999
T ou %T	0	9999
C ou %C	0	9999
M ou %MW	200	9999
L ou %MD	100	4998
F ou %MF	100	4998

La plage d'attribution est modifiable pour chaque type de variables en utilisant la directive de compilation #SR« type »=« début », « fin »
« type » désigne le type de variable, début et fin, les nouvelles bornes à utiliser.
Cette directive modifie l'attribution des variables automatiques pour la totalité du folio où elle est écrite et jusqu'à une prochaine directive « #SR ».

Symboles à adresse fixe

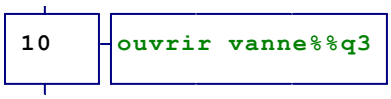
La syntaxe des symboles automatiques est la suivante :

```
_« nom du symbole » %« nom de variable »_
```

Par exemple :

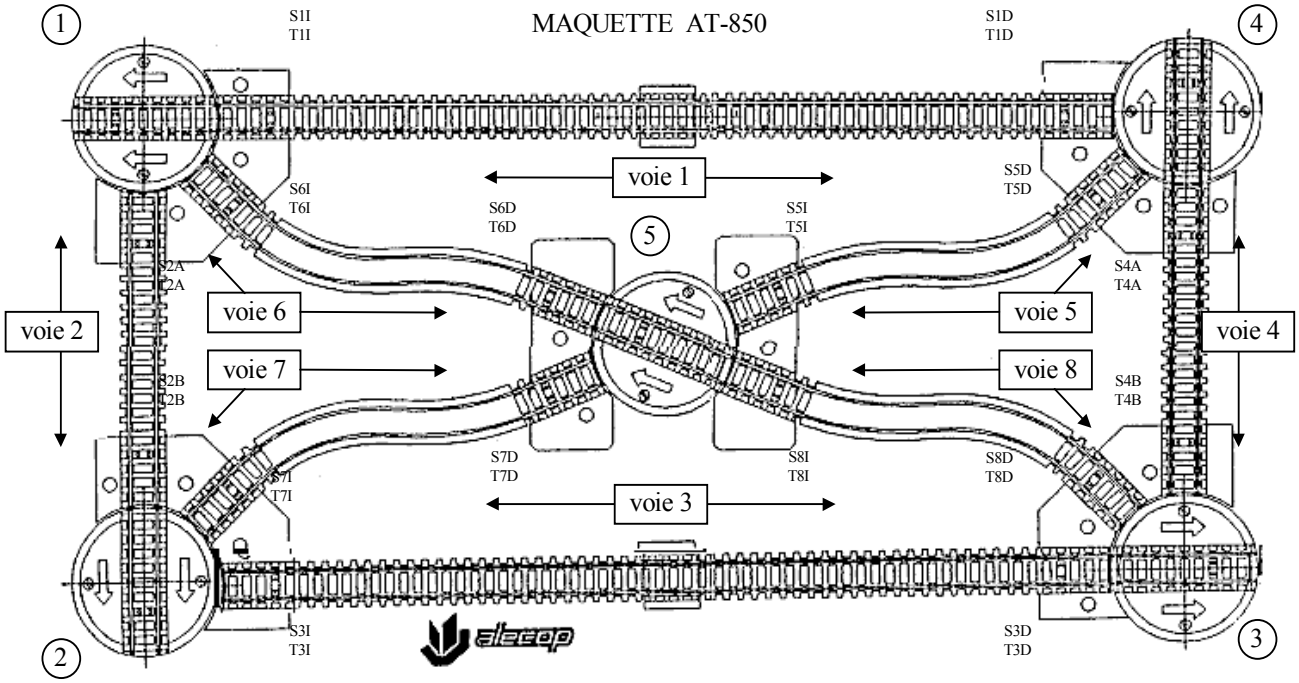
```
ouvrir vanne%%q3
```

Désigne un symbole qui sera associé à la variable %Q3.



A propos des exemples

Pour mieux illustrer ce manuel, nous avons développé des exemples fonctionnant avec une maquette de train dont voici le schéma :



Nous avons utilisé des cartes d'E/S sur PC pour piloter cette maquette. Les symboles définis par le constructeur de la maquette ont été conservés.

Le fichier de symboles suivant a été créé :

AV1	O0	alimentation voie 1
AV2	O1	alimentation voie 2
AV3	O2	alimentation voie 3
AV4	O3	alimentation voie 4
AV5	O4	alimentation voie 5
AV6	O5	alimentation voie 6
AV7	O6	alimentation voie 7
AV8	O7	alimentation voie 8
AP1	O8	alimentation plateforme 1
AP2	O9	alimentation plateforme 2
AP3	O10	alimentation plateforme 3
AP4	O11	alimentation plateforme 4
AP5	O12	alimentation plateforme 5
IP1	O13	rotation plateforme 1
IP2	O14	rotation plateforme 2
IP3	O15	rotation plateforme 3
IP4	O16	rotation plateforme 4
IP5	O17	rotation plateforme 5
ZP1	O18	initialisation plateforme 1
ZP2	O19	initialisation plateforme 2
ZP3	O20	initialisation plateforme 3
ZP4	O21	initialisation plateforme 4
ZP5	O22	initialisation plateforme 5
DV1	O23	direction voie 1
DV2	O24	direction voie 2
DV3	O25	direction voie 3
DV4	O26	direction voie 4
DV5	O27	direction voie 5
DV6	O28	direction voie 6
DV7	O29	direction voie 7
DV8	O30	direction voie 8
S1D	O31	feu droit voie 1
S1I	O32	feu gauche voie 1
S2A	O33	feu haut voie 2
S2B	O34	feu bas voie 2
S3D	O35	feu droit voie 3
S3I	O36	feu gauche voie 3
S4A	O37	feu haut voie 4
S4B	O38	feu bas voie 4
S5D	O39	feu droit voie 5
S5I	O40	feu gauche voie 5
S6D	O41	feu droit voie 6
S6I	O42	feu gauche voie 6
S7D	O43	feu droit voie 7
S7I	O44	feu gauche voie 7
S8D	O45	feu droit voie 8
S8I	O46	feu gauche voie 8
T1D	i0	train droit voie 1
T1I	i1	train gauche voie 1
T2A	i2	train haut voie 2
T2B	i3	train bas voie 2
T3D	i4	train droit voie 3
T3I	i5	train gauche voie 3
T4A	i6	train haut voie 4
T4B	i7	train bas voie 4
T5D	i8	train droit voie 5

T5I	i9	train gauche voie 5
T6D	i10	train droit voie 6
T6I	i11	train gauche voie 6
T7D	i12	train droit voie 7
T7I	i13	train gauche voie 7
T8D	i14	train droit voie 8
T8I	i15	train gauche voie 8
TP1	i16	train plateforme 1
TP2	i17	train plateforme 2
TP3	i18	train plateforme 3
TP4	i19	train plateforme 4
TP5	i20	train plateforme 5
P1P	i21	index plateforme 1
P2P	i22	index plateforme 2
P3P	i23	index plateforme 3
P4P	i24	index plateforme 4
P5P	i25	index plateforme 5
P1Z	i26	init plateforme 1
P2Z	i27	init plateforme 2
P3Z	i28	init plateforme 3
P4Z	i29	init plateforme 4
P5Z	i30	init plateforme 5
ERR	i31	court-circuit

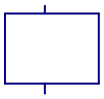
Grafcet

AUTOMGEN supporte les éléments suivants :

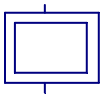
- ⇨ divergences et convergences en « Et » et en « Ou »,
- ⇨ étapes puits et sources,
- ⇨ transitions puits et sources,
- ⇨ synchronisation,
  - ⇨ forçages de Grafcets,
  - ⇨ mémorisation de Grafcets,
  - ⇨ figeage,
  - ⇨ macro-étapes,
  - ⇨ étapes encapsulantes.

Etapes Grafcets

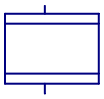
Les différentes formes d'étapes sont :



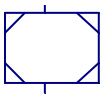
étape normale



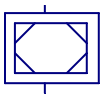
étape initiale



macro-étape

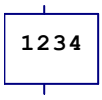


étape encapsulante



étape encapsulante initiale

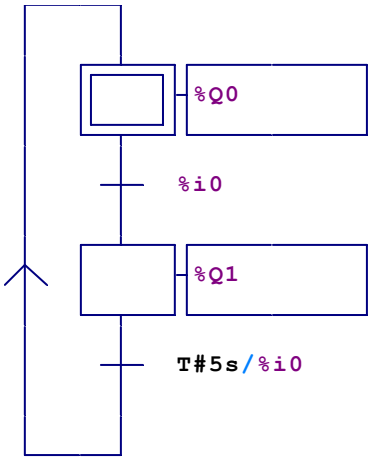
Les étapes doivent être documentées avec un numéro allant de 0 à 9999.



Par exemple :

Un symbole commençant par une lettre et composé d'au plus 4 caractères peut être utilisé. Par exemple :

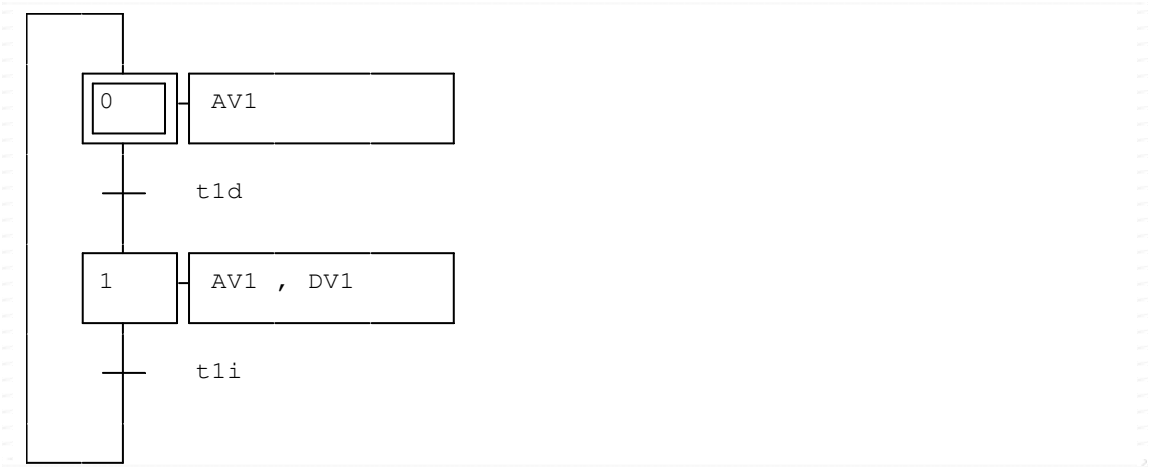
GRAFCET++ : les étapes peuvent ne pas être numérotées. Par exemple :



Grafcet simple

L'écriture de Grafcet en ligne se résume à la juxtaposition d'étapes et de transitions.
Illustrons un Grafcet en ligne avec l'exemple suivant :

Cahier des charges :
La locomotive doit partir sur la voie 1 vers la droite, jusqu'au bout de la voie. Elle revient ensuite dans le sens inverse jusqu'à l'autre bout et recommence.
Solution 1 :



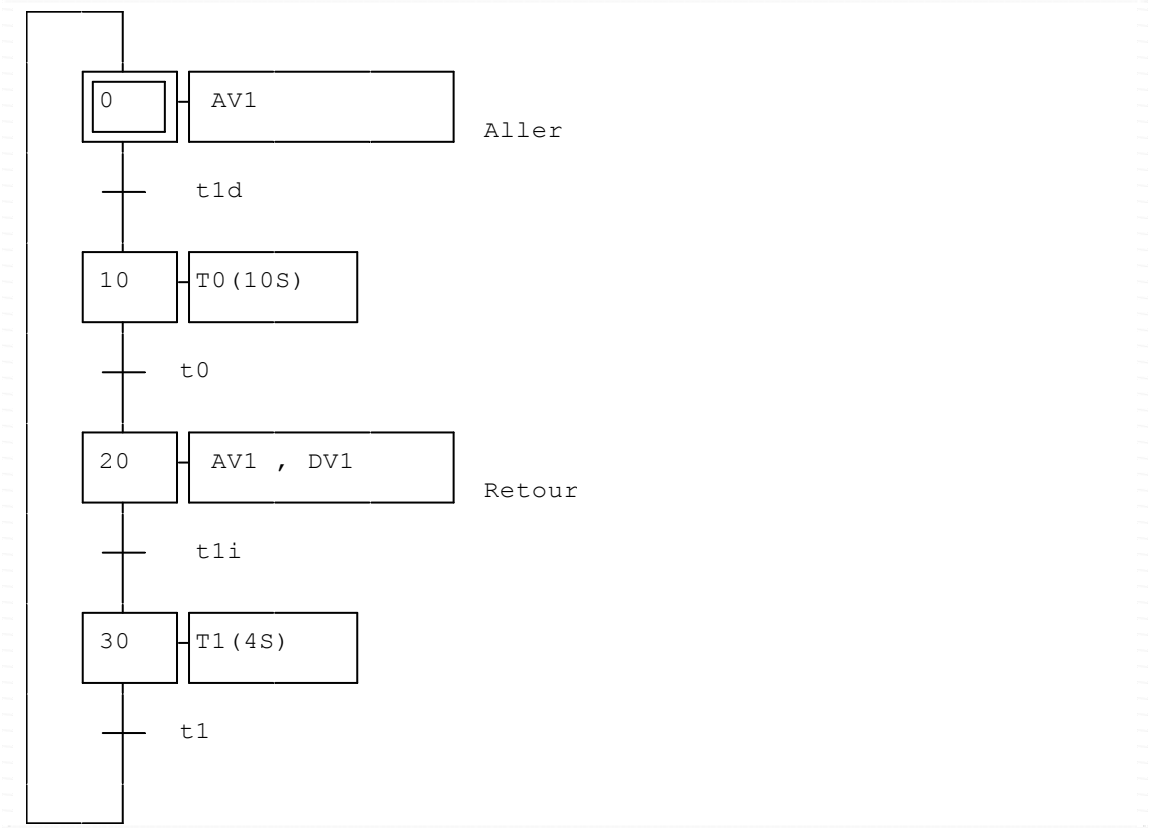
exemples\grafcet\simple1.agn

Solution 2 :

exemples\grafcet\simple2.agn

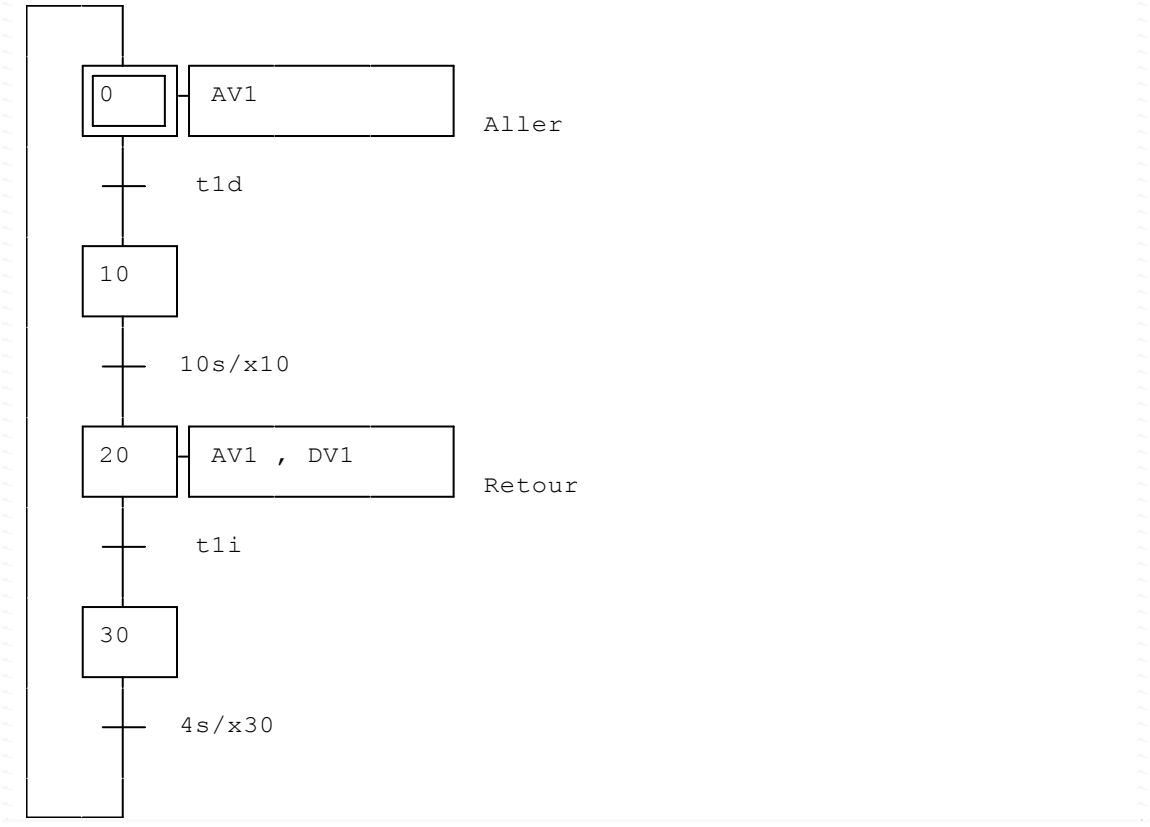
La différence entre ces deux solutions réside dans l'utilisation des actions « Affectation » pour le premier exemple et des actions « Mise à un » et « Mise à zéro » pour le deuxième.
Modifions notre cahier des charges en imposant une attente de 10 secondes lorsque la locomotive arrive à droite de la voie 1 et une attente de 4 secondes lorsque la locomotive arrive à gauche de la voie 1.

Solution 1 :



exemples\grafcet\simple3.agn

Solution 2 :

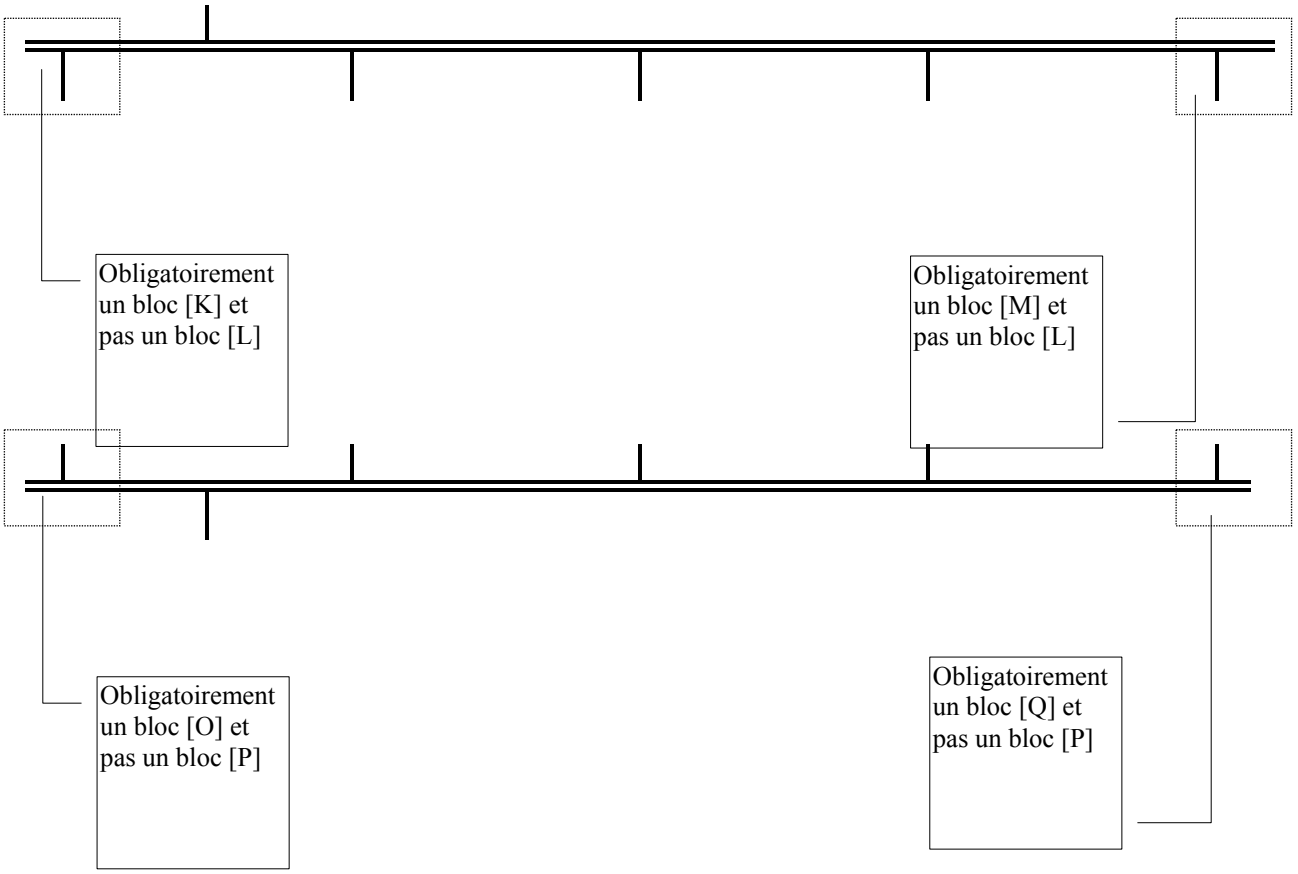


exemples\grafcet\simple4.agn

La différence entre les exemples 3 et 4 réside dans le choix de la syntaxe utilisée pour définir les temporisations. Le résultat au niveau du fonctionnement est identique.

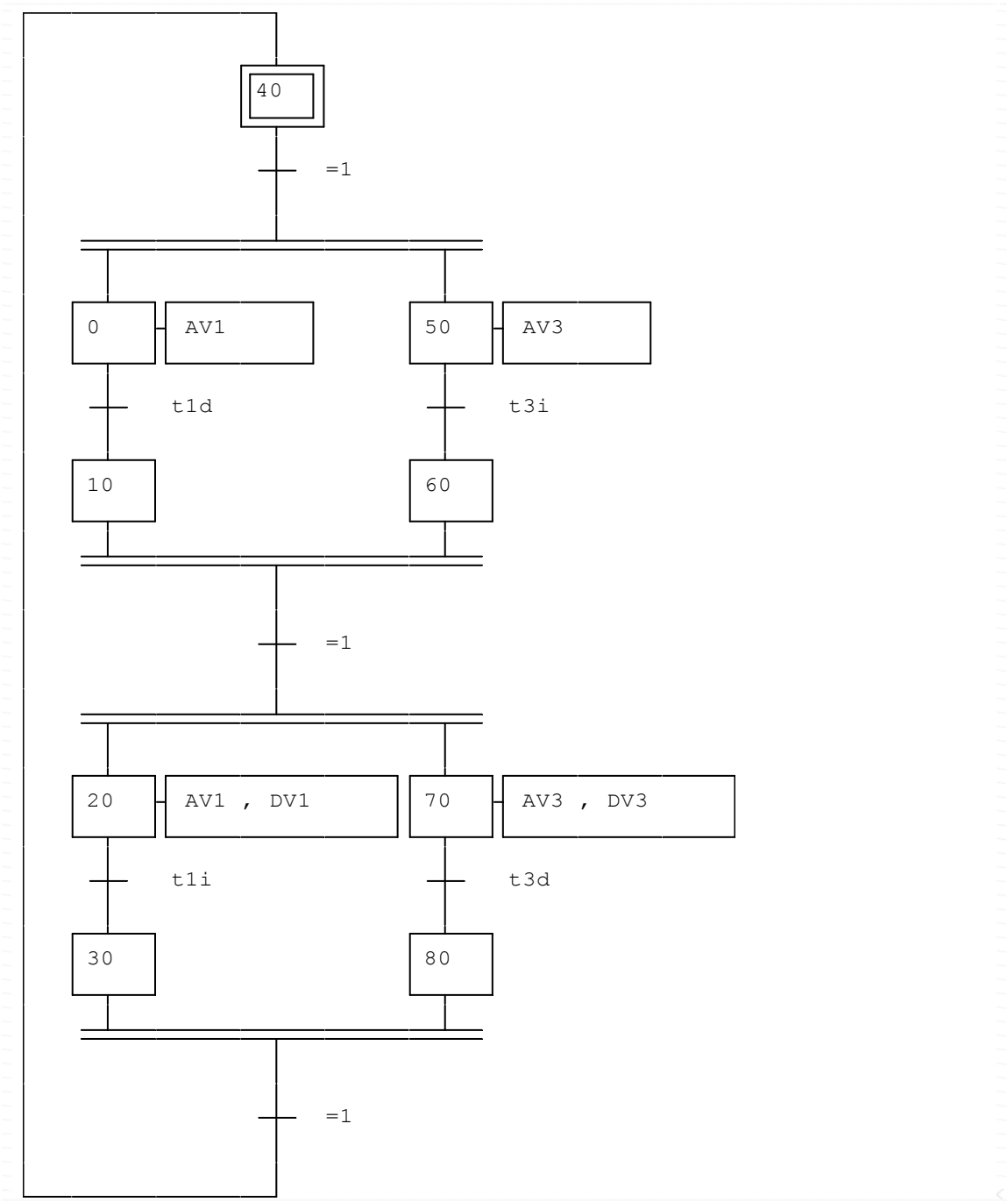
Divergence et convergence en « Et »

Les divergences en « Et » peuvent avoir n branches. L'important est de respecter l'utilisation des blocs de fonction :



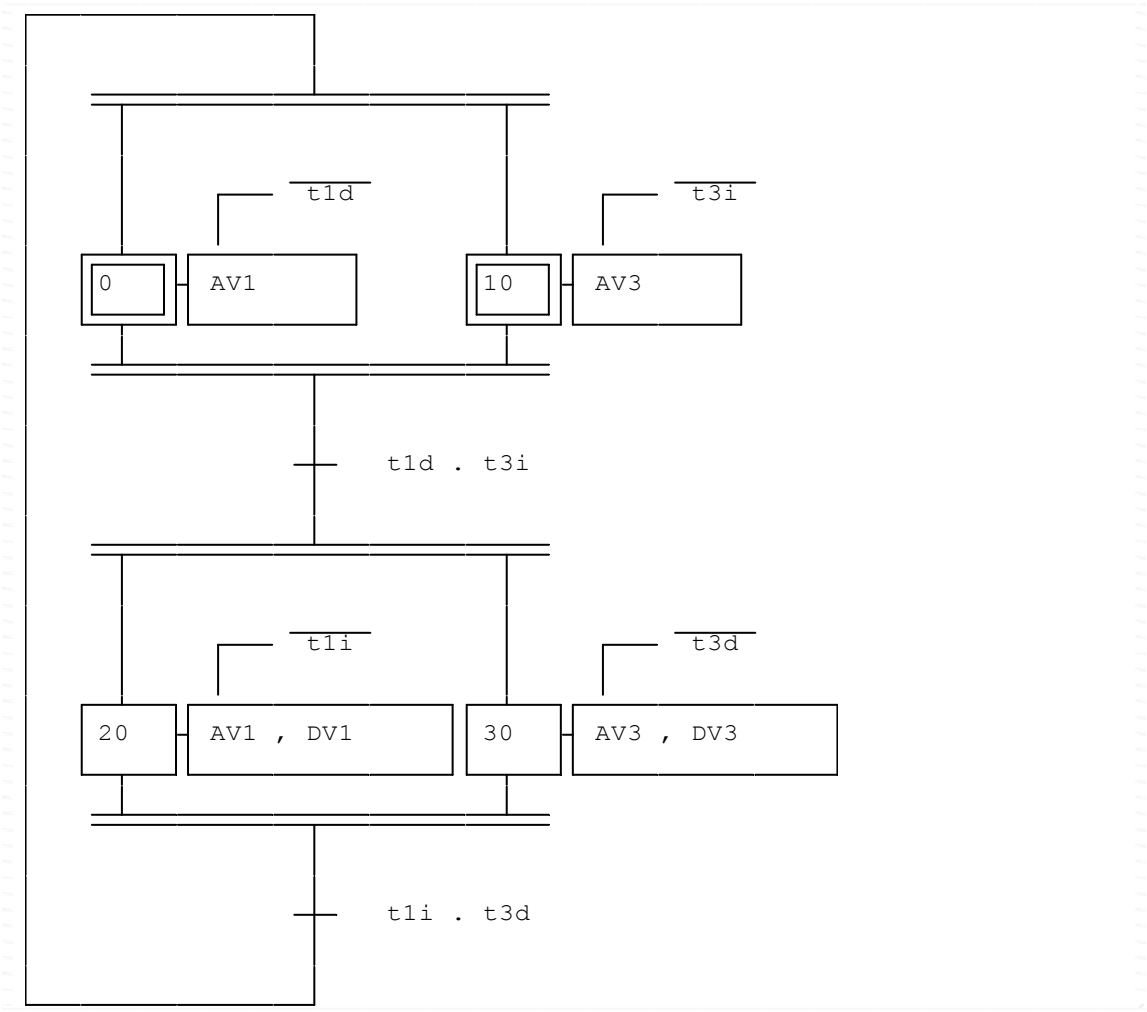
Illustrons l'utilisation des divergences et convergences en « Et ».
Cahier des charges :
Nous allons utiliser deux locomotives : la première effectuera des allers et retours sur la voie 1, la seconde sur la voie 3. Les deux locomotives seront synchronisées (elles s'attendront en bout de voie).

Solution 1 :



exemples\grafcet\divergence et 1.agn

Solution 2 :

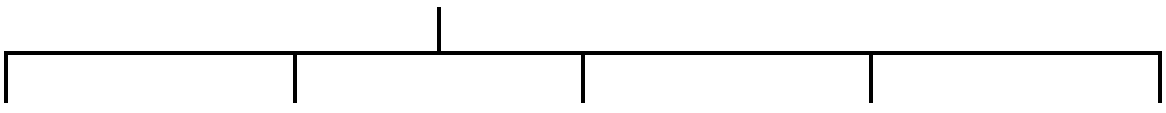


exemples\grafcet\divergence et 2.agn

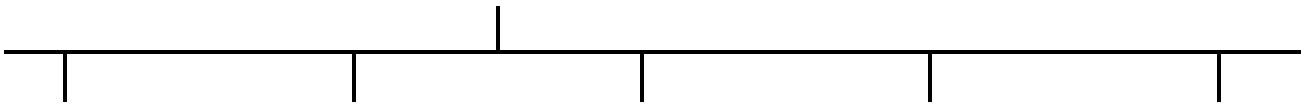
Ces deux solutions sont équivalentes au niveau du fonctionnement. La deuxième est une version plus compacte qui utilise des actions conditionnées.

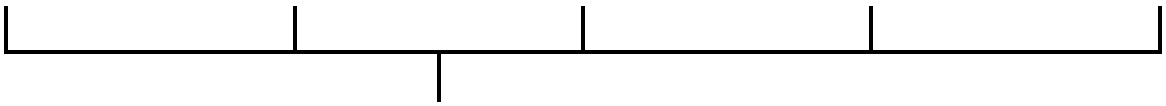
Divergence et convergence en « Ou »

Les divergences en « Ou » peuvent avoir n branches. L'important est de respecter l'utilisation des blocs de fonction :

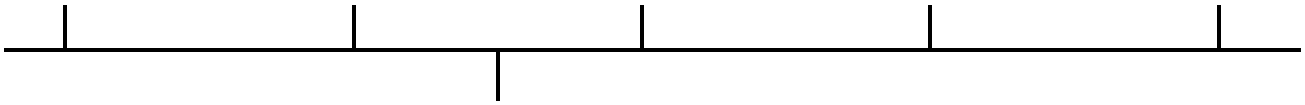


ou

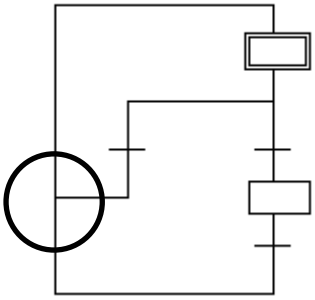




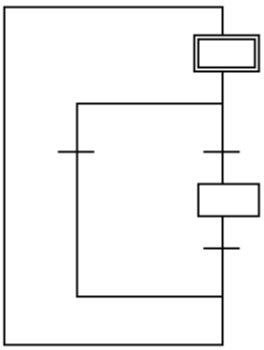
ou



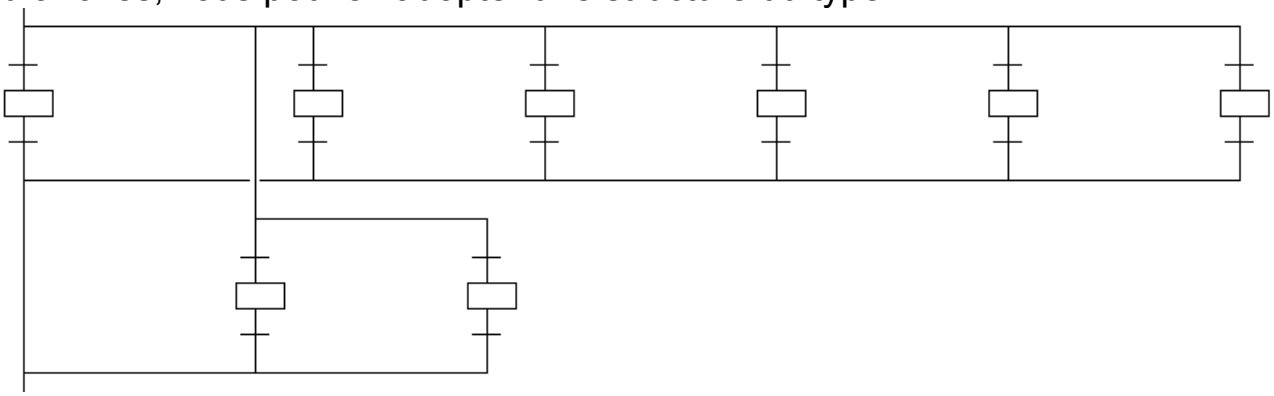
Les divergences en « Ou » doivent obligatoirement se brancher sur des liaisons descendantes. Par exemple :



incorrect, le bon dessin est :



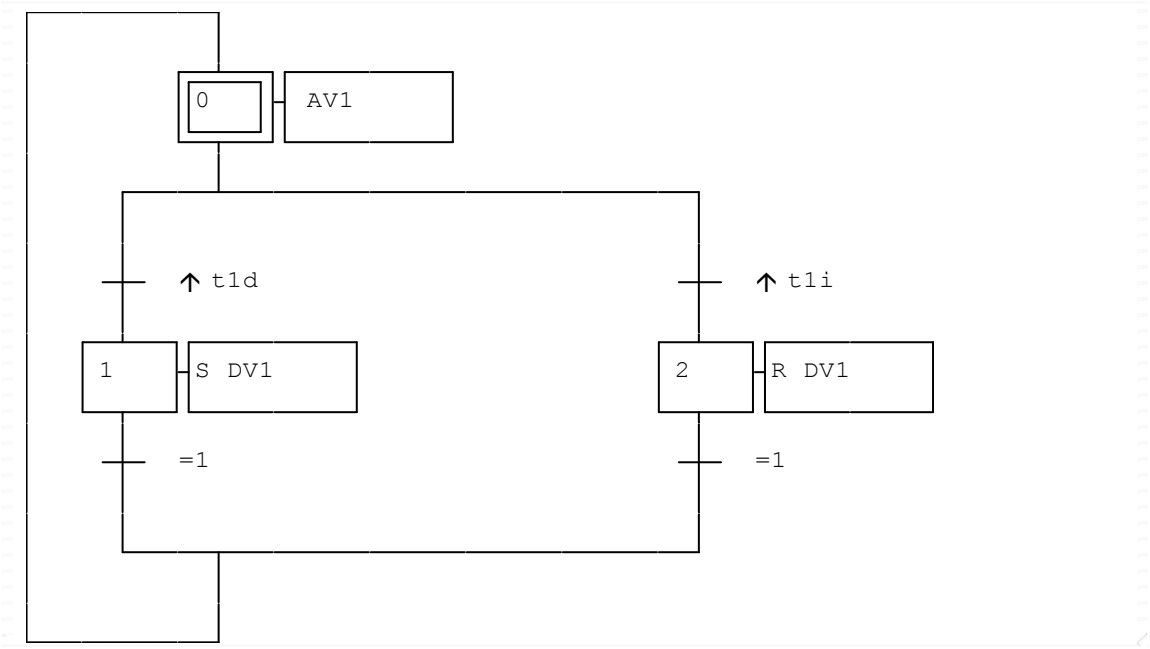
Si la largeur de la page vous interdit l'écriture d'un grand nombre de branches, vous pouvez adopter une structure du type:



Voyons un exemple pour illustrer l'utilisation des divergences et convergences en « Ou » :

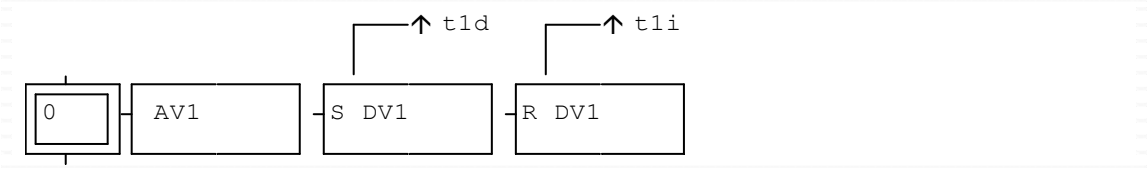
Cahier des charges :
Reprenons le cahier des charges du premier exemple de chapitre : aller et retour d'une locomotive sur la voie 1.

Solution :



exemples\grafcet\divergence ou.agn

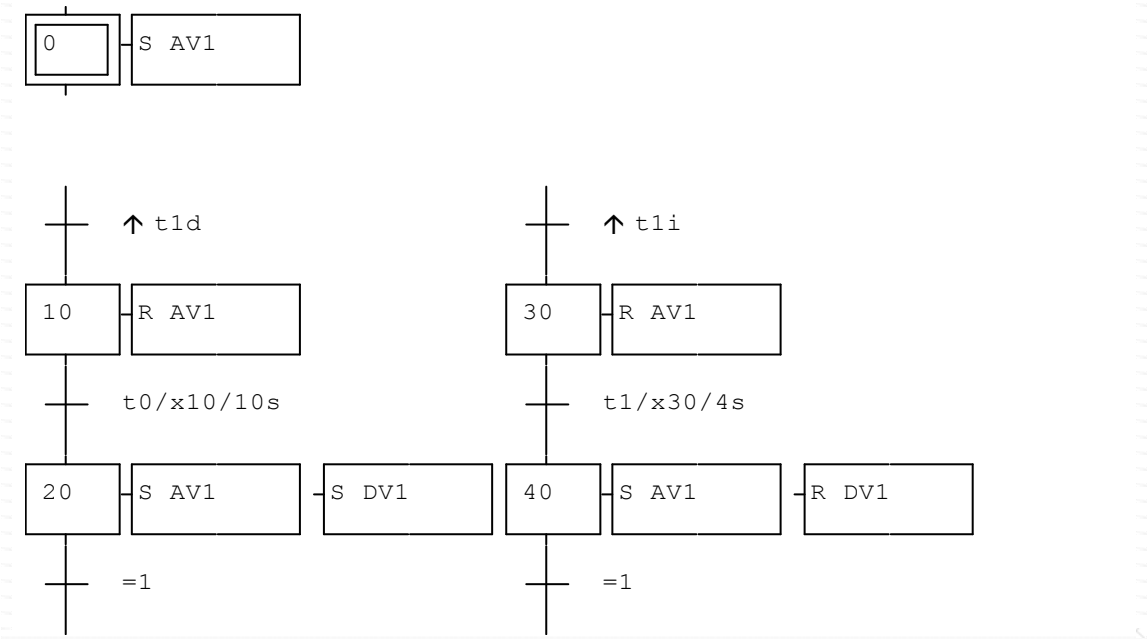
Ce Grafcet pourrait se résumer à une étape en utilisant des actions conditionnées, comme dans cet exemple :



exemples\grafcet\action conditionnée.agn

Etapes puits et sources, transitions puits et sources

Illustrons ces principes par des exemples :
Cahier des charges :
Traitons à nouveau le second exemple de ce chapitre : aller et retour d'une locomotive sur la voie 1 avec attente en fin de voie.
Solution :



exemples\grafcet\étapes puits et sources.agn

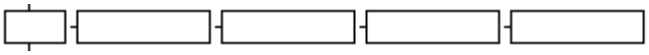
Actions multiples

Nous avons déjà utilisé dans ce chapitre des actions multiples et des actions conditionnées. Détaillons ici ces deux principes.

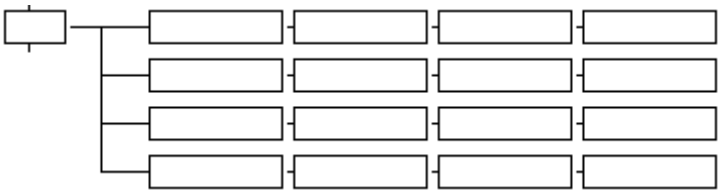
Comme il est dit dans le chapitre consacré au compilateur, plusieurs actions peuvent être écrites dans un même rectangle, le caractère « , » (virgule) sert de délimiteur dans ce cas.

Lorsqu'une condition est ajoutée sur un rectangle d'action, c'est l'ensemble des actions contenues dans ce rectangle qui sont conditionnées.

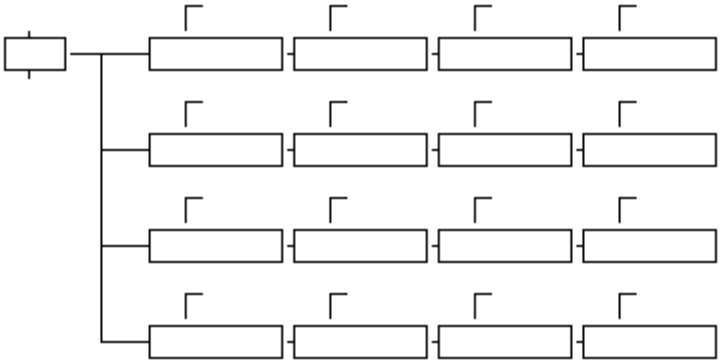
Plusieurs rectangles d'action peuvent être associés à une étape :



autre possibilité :



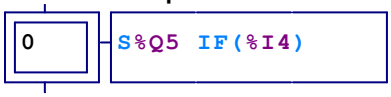
Chacun des rectangles peut recevoir une condition différente :



Actions conditionnées, actions évènementielles

Pour dessiner une action conditionnée ou évènementielle, placez le curseur sur le rectangle d'action, cliquez sur le bouton droit de la souris et choisissez « Action conditionnelle » ou action évènementielle dans le menu. Pour documenter la condition sur action, cliquez sur l'élément  ou ou  .

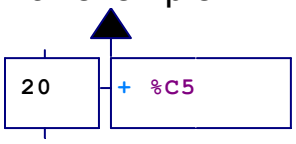
La syntaxe IF(condition) permet d'écrire une condition sur action dans le rectangle d'action.
Par exemple :



Actions sur activation ou sur désactivation d'étape

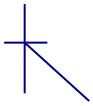
Les symboles  et permettent respectivement de spécifier pour les actions contenues dans un rectangles qu'elles doivent être

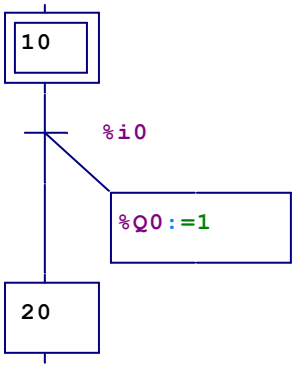
effectuées une seule fois à l'activation ou à la désactivation de l'étape. Par exemple :



Incrémenter le compteur 5 une fois à l'activation de l'étape 20.

Actions sur franchissement de transition

Les symboles  et  permettent de définir des actions sur franchissement de transition. Par exemple :



%Q0 sera activée au franchissement de la transition entre les étapes 10 et 20.

Synchronisation

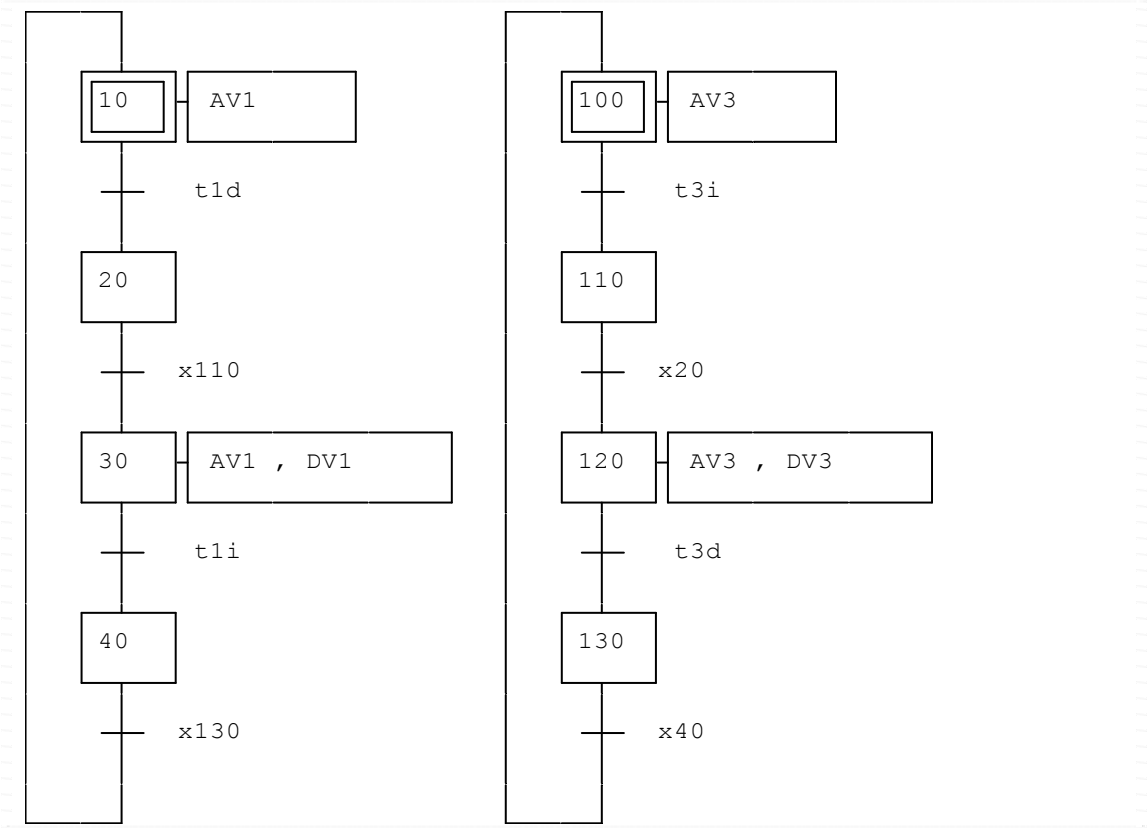
Reprenons un exemple déjà traité pour illustrer la synchronisation de Grafcets.

Cahier des charges:

Aller et retour de deux locomotives sur les voies 1 et 3 avec attente entre les locomotives en bout de voie.

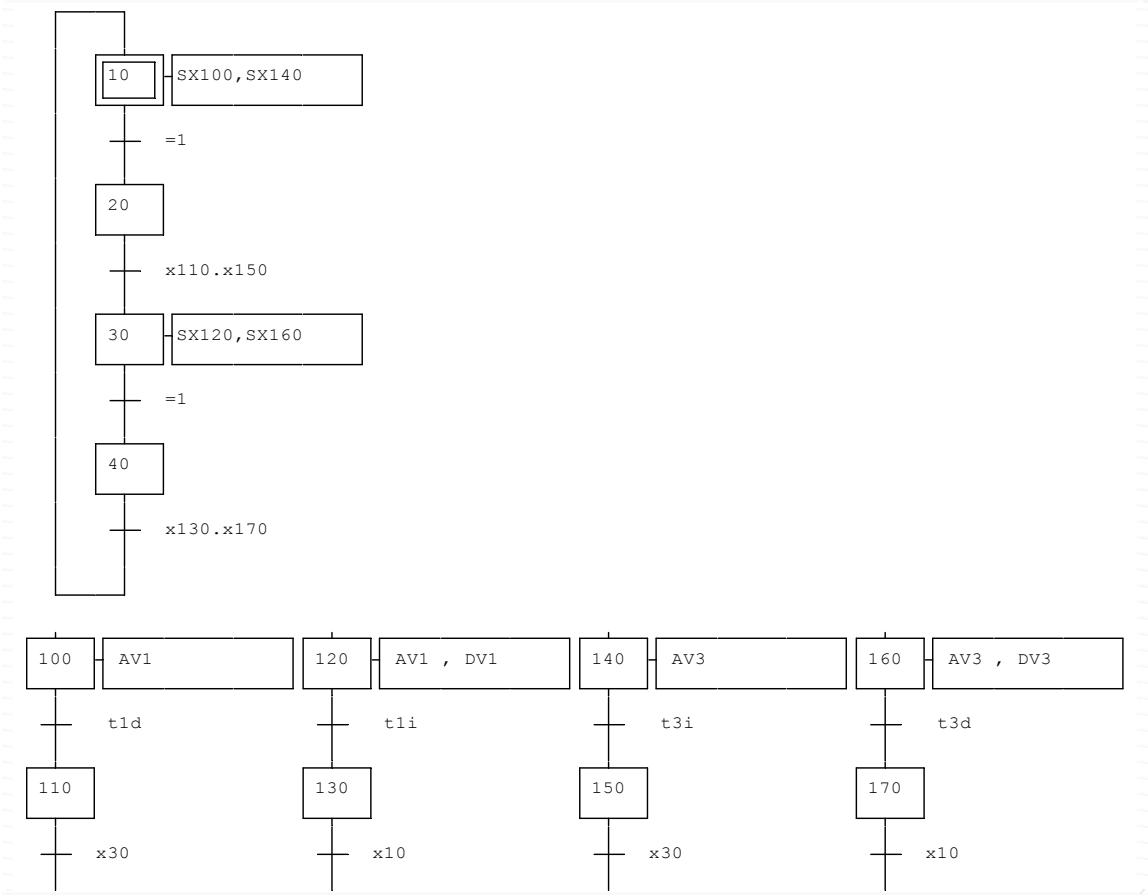
Cet exemple avait été traité avec une divergence en « Et ».

Solution 1 :



exemples\grafcet\synchro1.agn

Solution 2 :



exemples\grafcet\synchro2.agn

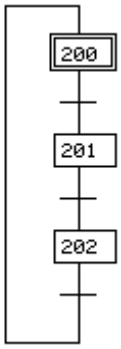
Cette deuxième solution est un excellent exemple illustrant l'art de compliquer les choses les plus simples à des fins pédagogiques.

Forçages de Grafcet

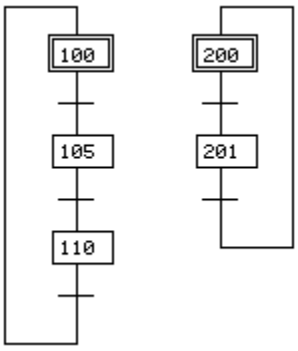
Le compilateur regroupe les étapes en fonction des liens qui sont établis entre elles. Pour désigner un Grafcet, il suffit de faire référence à une des étapes composant ce Grafcet.

On peut également désigner l'ensemble des Grafcets présents sur un folio en mentionnant le nom du folio où ils se trouvent.

Par exemple:



Pour désigner ce Grafcet nous parlerons du Grafcet 200, du Grafcet 201 ou encore du Grafcet 202.
Le Grafcet en tant qu'ensemble d'étapes devient donc une variable de type structurée, composée de n étapes, chacune de ces étapes, étant, soit active, soit inactive.
Comme nous l'avons vu, AUTOMGEN divise les étapes en ensembles indépendants. Ces ensembles peuvent être regroupés, ceci permet donc de les considérer comme un seul Grafcet. Pour regrouper plusieurs Grafcets, il faut utiliser la directive de compilation « `#G:g1,g2` » (commande à inclure dans un commentaire). Cette commande regroupe les Grafcets g1 et g2. Rappelons que la désignation d'un Grafcet s'effectue en invoquant le numéro d'une de ses étapes.
Voyons un exemple :
#G:105,200
cette directive de compilation regroupe ces deux Grafcets :



Remarque : plusieurs directives « #G » peuvent être utilisées afin de regrouper plus de deux Grafcets.
Nous allons maintenant détailler les ordres de forçage utilisables. Ils seront simplement écrits dans des rectangles d'action comme des affectations classiques. Ils supporteront également les opérateurs S(mise à un), R(mise à zéro), N(affectation complémentée) et I(inversion) ainsi que les actions conditionnelles.

Forçage d'un Grafcet selon une liste d'étapes actives

Syntaxe:
« F<Grafcet>:{<liste d'étapes actives>} »
ou
« F/<nom de folio>:{<liste d'étapes actives>} »
Le ou les Grafcets ainsi désignés seront forcés à l'état défini par la liste des étapes actives se trouvant entre accolades. Si plusieurs étapes doivent être actives alors il faut les séparer par le caractère « , » (virgule). Si le ou les Grafcets doivent être forcés à l'état vide (aucune étape active) alors aucune étape ne doit être précisée entre les deux accolades.
Le numéro des étapes peut être précédé de « X ». On peut ainsi associer un symbole au nom d'une étape.
Exemples :
« F10:{0} »
force toutes les étapes du Grafcet 10 à 0 sauf l'étape 0 qui sera forcée à 1.
« F0:{4,8,9,15} »
force toutes les étapes du Grafcet 0 à 0 sauf les étapes 4,8,9 et 15 qui seront forcées à 1.
« F/marche normale :{} »
force tous les Grafcets se trouvant sur le folio « marche normale » à l'état vide.

Mémorisation de l'état d'un Grafcet

Etat actuel d'un Grafcet:
Syntaxe:
« G<Grafcet>:<N° de bit> »
ou
« G/<nom de folio>:<N° de bit> »
Cette commande mémorise l'état d'un ou plusieurs Grafcets dans une série de bits. Il est nécessaire de réserver un espace au stockage de l'état du ou des Grafcets désignés (un bit par étape). Ces bits de stockage doivent être consécutifs. Vous devez utiliser une commande #B pour réserver un espace linéaire de bits.

Le numéro de l'étape désignant le Grafcet peut être précédé de « X » . On peut ainsi associer un symbole au nom d'une étape. Le numéro du bit peut être précédé de « U » ou de « B ». On peut ainsi associer un symbole au premier bit de la zone de stockage d'état.

Etat particulier d'un Grafcet :
Syntaxe:
« G<Grafcet>:<N° de bit> {liste d'étapes actives} »

ou
« G/<nom de folio> :<N° de bit> {liste d'étapes actives} »
Cette commande mémorise l'état défini par la liste d'étapes actives appliquées aux Grafcets spécifiés à partir du bit indiqué. Il est nécessaire ici aussi de réserver un nombre suffisant de bits. Si une situation vide doit être mémorisée alors aucune étape ne doit apparaître entre les deux accolades.
Le numéro des étapes peut être précédé de « X ». On peut ainsi associer un symbole au nom d'une étape. Le numéro du bit peut être précédé de « U » ou de « B ». On peut ainsi associer un symbole au premier bit de la zone de stockage d'état.
Exemples:
« G0:100 »
mémorise l'état actuel du Grafcet 0 à partir de U100.
« G0:U200 »
mémorise l'état vide du Grafcet 0 à partir de U200.
« G10:150{1,2} »
mémorise l'état du Grafcet 10, dans lequel seules les étapes 1 et 2 sont actives, à partir de U150.
« G/PRODUCTION :_SAUVE ETAT PRODUCTION_ »
mémorise l'état des Grafcets se trouvant sur le folio « PRODUCTION » dans la variable _SAUVE ETAT PRODUCTION_.

Forçage d'un Grafcet à partir d'un état mémorisé

Syntaxe:
« F<Grafcet>:<N° de bit> »
ou
« F/<Nom de folio>:<N° de bit> »

Force le ou les Grafcets avec l'état mémorisé à partir du bit précisé.
Le numéro de l'étape désignant le Grafcet peut être précédé de 'X'. On peut ainsi associer un symbole au nom d'une étape. Le numéro du bit peut être précédé de « U » ou de « B ». On peut ainsi associer un symbole au premier bit de la zone de stockage d'état.
Exemple:
« G0:100 »
mémorise l'état actuel du Grafcet 0
« F0:100 »
et restaure cet état

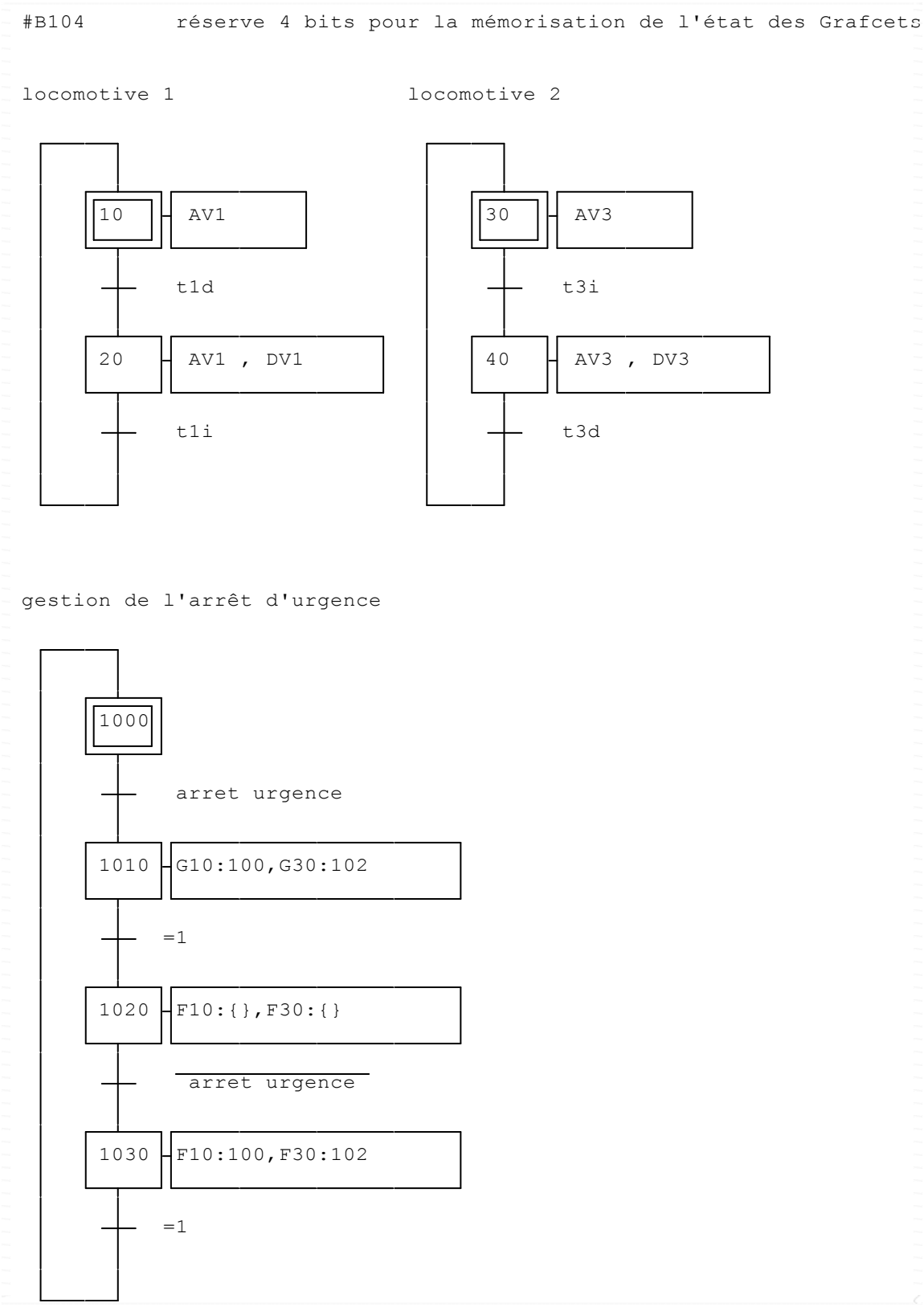
Figeage d'un Grafcet

Syntaxe:
« F<Grafcet> »

ou
« F/<Nom de folio> »
Fige un ou des Grafcets : interdit toute évolution de ceux-ci.
Exemple :
« F100 »
fige le Grafcet 100
« F/production »
fige les Grafcets contenus dans le folio « production »
Illustrons les forçages par un exemple.

Cahier des charges :
Reprenons un exemple déjà traité : aller et retour de deux locomotives sur les voies 1 et 3 (cette fois sans attente entre les locomotives) et ajoutons le traitement d'un arrêt d'urgence. Lorsque l'arrêt d'urgence est détecté toutes les sorties sont remises à zéro. A la disparition de l'arrêt d'urgence le programme doit reprendre là où il s'était arrêté.

Solution 1 :

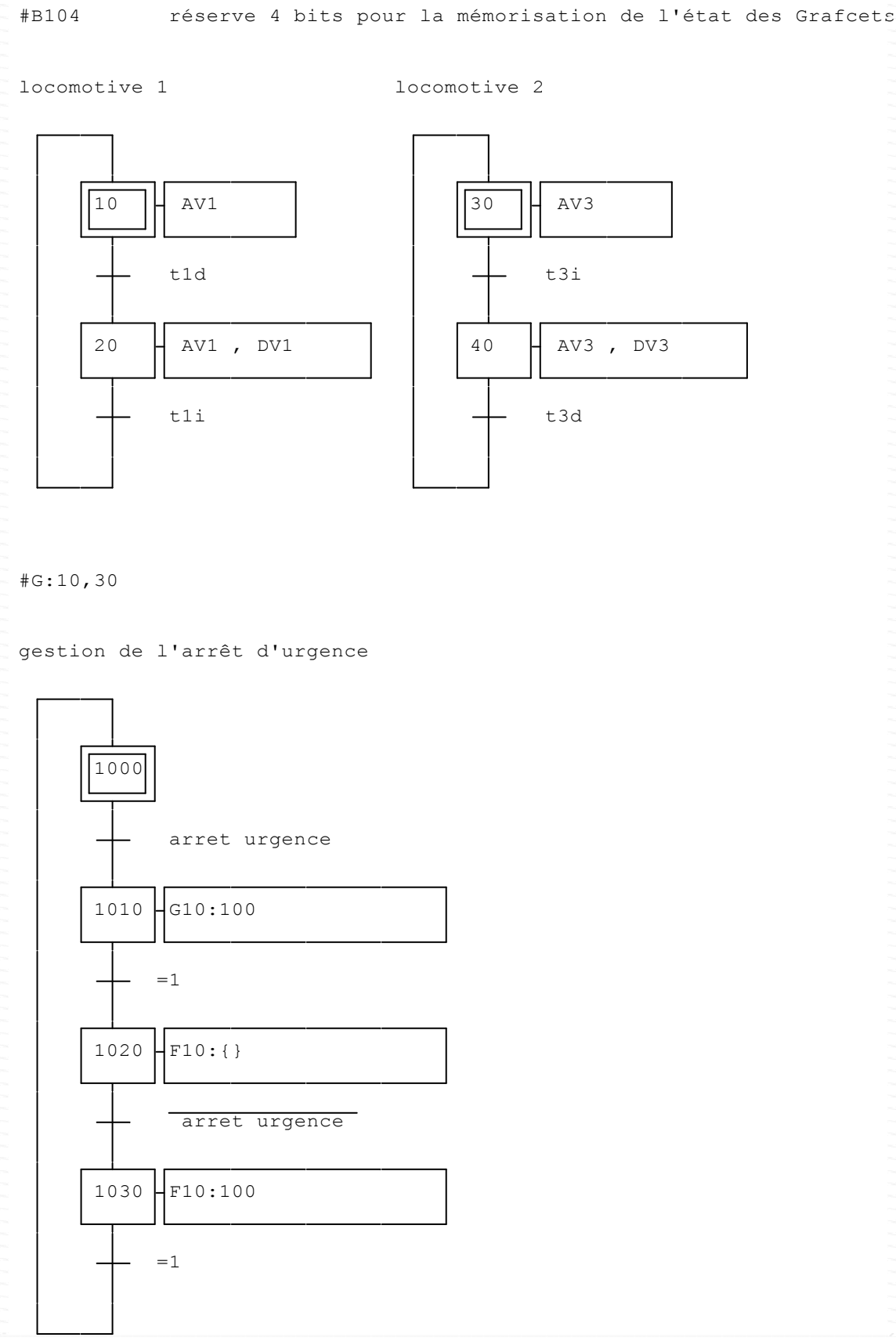


exemples\grafcet\forçage1.agn

Notez l'utilisation de la directive #B104 qui permet de réserver quatre bits consécutifs (U100 à U103) pour la mémorisation de l'état des deux Grafcets.

« _arret urgence_ » a été associé à un bit (U1000). Son état peut donc être modifié à partir de l'environnement en cliquant dessus lorsque la visualisation dynamique est activée.

Solution 2 :



exemples\grafcet\forçage2.agn

Cette deuxième solution montre l'utilisation de la directive de compilation « #G » qui permet de regrouper les Grafcets pour les commandes de forçages.

Forçages de Grafcet (norme 60848)

Cette norme définit les ordres de forçages dans des rectangles d'action doubles. Les actions de forçages sont exécutées tant que la condition associée : étape ou logigramme sont vrais. Des conditions peuvent être ajoutées sur les rectangles d'actions doubles : condition sur action, action évènementielle, action sur activation ou désactivation.

Forçage d'un Grafcet selon une liste d'étapes actives

La syntaxe est : G<grafcet à forcer>{<liste d'étapes à forcer à l'état vrai>}
La ou les étapes mentionnées dans la liste sont forçées à l'état vrai, les autres à l'état faux. Une liste d'étapes vide engendre un forçage de toutes les étapes à l'état faux.
Exemple :

```
G10{20,30}
```

Ici 10 représente le Grafcet à forcer : Grafcet contenant l'étape 10.
Autre exemple :

```
Gfolio à forcer{100,200,300}
```

Force l'ensemble des Grafcets se trouvant sur le folio nommé « folio à forcer » dans l'état tel que les étapes 100, 200 et 300 sont à l'état vrai et les autres étapes à 0.

Forçage d'un Grafcet dans son état initial

La syntaxe est : G<grafcet à forcer>{INIT}
Le ou les Grafcets sont forçés dans leur état initial.
Exemple :

```
G10{INIT}
```

Figeage d'un Grafcet

La syntaxe est : G<grafcet à forcer>{*}
Exemple :

```
G10{*}
```


Macro-étapes

AUTOMGEN implémente les macro-étapes.
Donnons quelques rappels à ce sujet :
Une macro-étape ME est l'unique représentation d'un ensemble unique d'étapes et de transitions nommé « expansion de ME ».
Une macro-étape obéit aux règles suivantes :

- ⇨ l'expansion de ME comporte une étape particulière dite **étape d'entrée** et une étape particulière dite **étape de sortie**.
- ⇨ l'étape d'entrée a la propriété suivante : tout franchissement d'une transition amont de la macro-étape, active l'étape d'entrée de son expansion.
- ⇨ l'étape de sortie a la propriété suivante : elle participe à la validation des transitions aval de la macro-étape.
- ⇨ en dehors des transitions amont et aval de ME, il n'existe aucune liaison structurale entre, d'une part une étape ou une transition de l'expansion ME, et d'autre part, une étape ou une transition n'appartenant pas à ME.

L'utilisation des macro-étapes sous AUTOMGEN a été définie comme suit :

- ⇨ l'expansion d'une macro-étape est un Grafcet se trouvant dans un folio distinct,
- ⇨ l'étape d'entrée de l'expansion d'une macro-étape devra porter le numéro 0 ou le repère Exxx, (avec xxx = un numéro quelconque),
- ⇨ l'étape de sortie de l'expansion d'une macro-étape devra porter le numéro 9999 ou le repère Sxxx, avec xxx = un numéro quelconque,
- ⇨ en dehors de ces deux dernières obligations, l'expansion d'une macro-étape peut être un Grafcet quelconque et pourra à ce titre contenir des macro-étapes (l'imbrication de macro-étapes est possible).

Comment définir une macro-étape ?

Le symbole  doît être utilisé. Pour poser ce symbole, cliquez sur un emplacement vide du folio avec le bouton droit et choisissez « Plus …/Macro-étape » dans le menu contextuel.

Pour définir l'expansion de la macro-étape, créez un folio, dessinez l'expansion et modifiez les propriétés du folio (en cliquant avec le bouton droit de la souris sur le nom du folio dans le navigateur). Réglez le type du folio sur « Expansion de macro-étapes » ainsi que le numéro de la macro-étape.

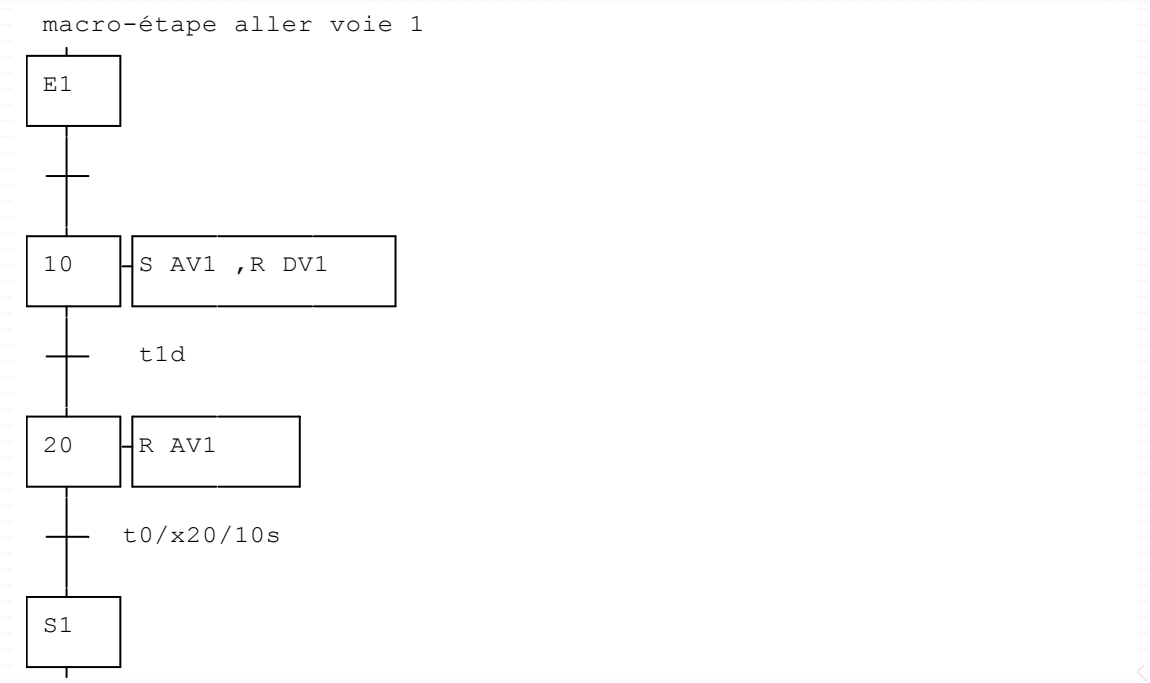
En mode exécution, il est possible de visualiser une expansion de macro-étape. Pour cela, il faut placer le curseur sur la macro-étape et cliquer sur le bouton gauche de la souris.

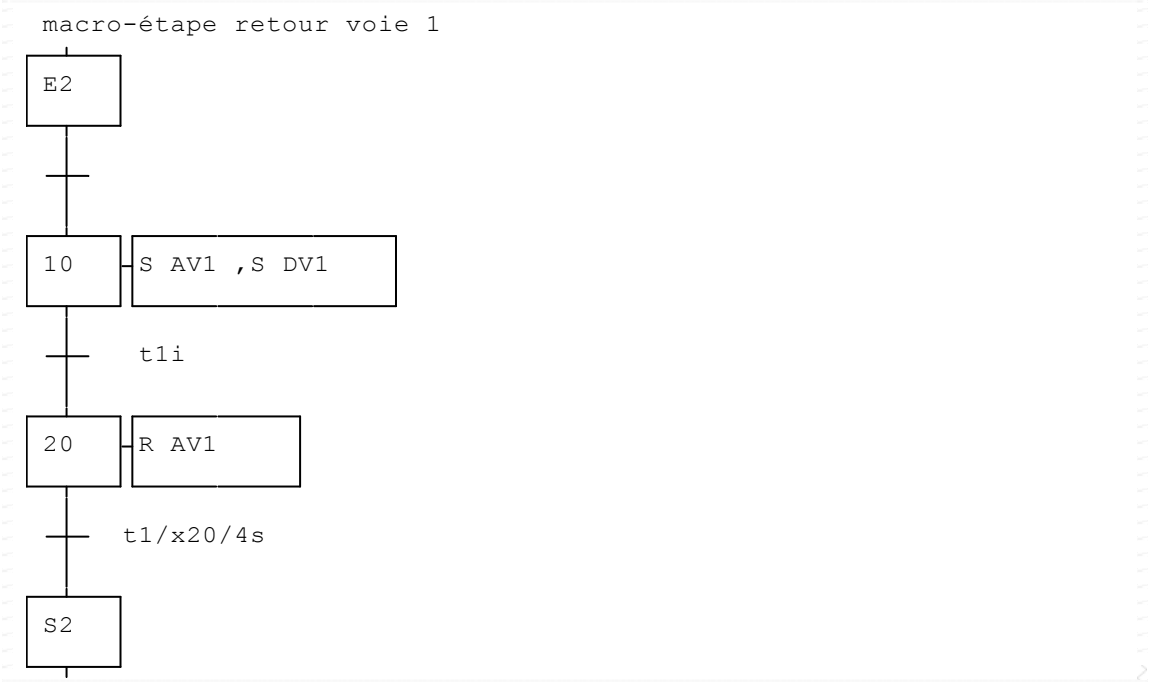
Remarques :

- ⇨ les étapes et les bits Utilisateur utilisés dans une expansion de macro-étape sont locaux, c'est à dire qu'ils n'ont aucun rapport avec les étapes et les bits d'autres Grafcets. Tous les autres types de variables n'ont pas cette caractéristique : ils sont communs à tous les niveaux.
- ⇨ si une zone de bits doit être utilisée de façon globale alors il faut la déclarer avec la directive de compilation « #B ».
- ⇨ les macro-étapes peuvent être imbriquées.

Illustrons l'utilisation des macro-étapes par un exemple déjà traité : aller et retour d'une locomotive sur la voie 1 avec attente en bout de voie. Nous décomposerons l'aller et le retour en deux macro-étapes distinctes.

Solution :





exemples\grafcet\macro-étape.agn

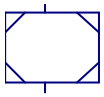
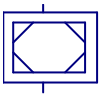
Etapes encapsulantes

Introduites dans la norme 60848, les étapes encapsulantes sont une évolution des notions proposées dans les macro étapes.

L'utilisation des étapes encapsulantes sous AUTOMGEN a été définie comme suit:

⇨ l'encapsulation se trouve dans un folio distinct.

Comment définir une étape encapsulante ?

Le symbole  ou  doît être utilisé. Pour poser ce symbole, cliquez avec le bouton droit sur un emplacement vide du folio et choisissez « Plus …/Etape encapsulante » dans le menu contextuel.

Comment définir une encapsulation ?

Pour définir l'encapsulation, créez un folio, dessinez l'encapsulation et modifiez les propriétés du folio (en cliquant avec le bouton droit de la souris sur le nom du folio dans le navigateur). Réglez le type du folio sur « Encapsulation » ainsi que le numéro de l'étape encapsulante.

Le symbole ✳ permet de définir l'état initial d'une encapsulation.

En mode exécution, il est possible de visualiser une encapsulation. Pour cela, il faut placer le curseur sur l'étape encapsulante et cliquer sur le bouton gauche de la souris.

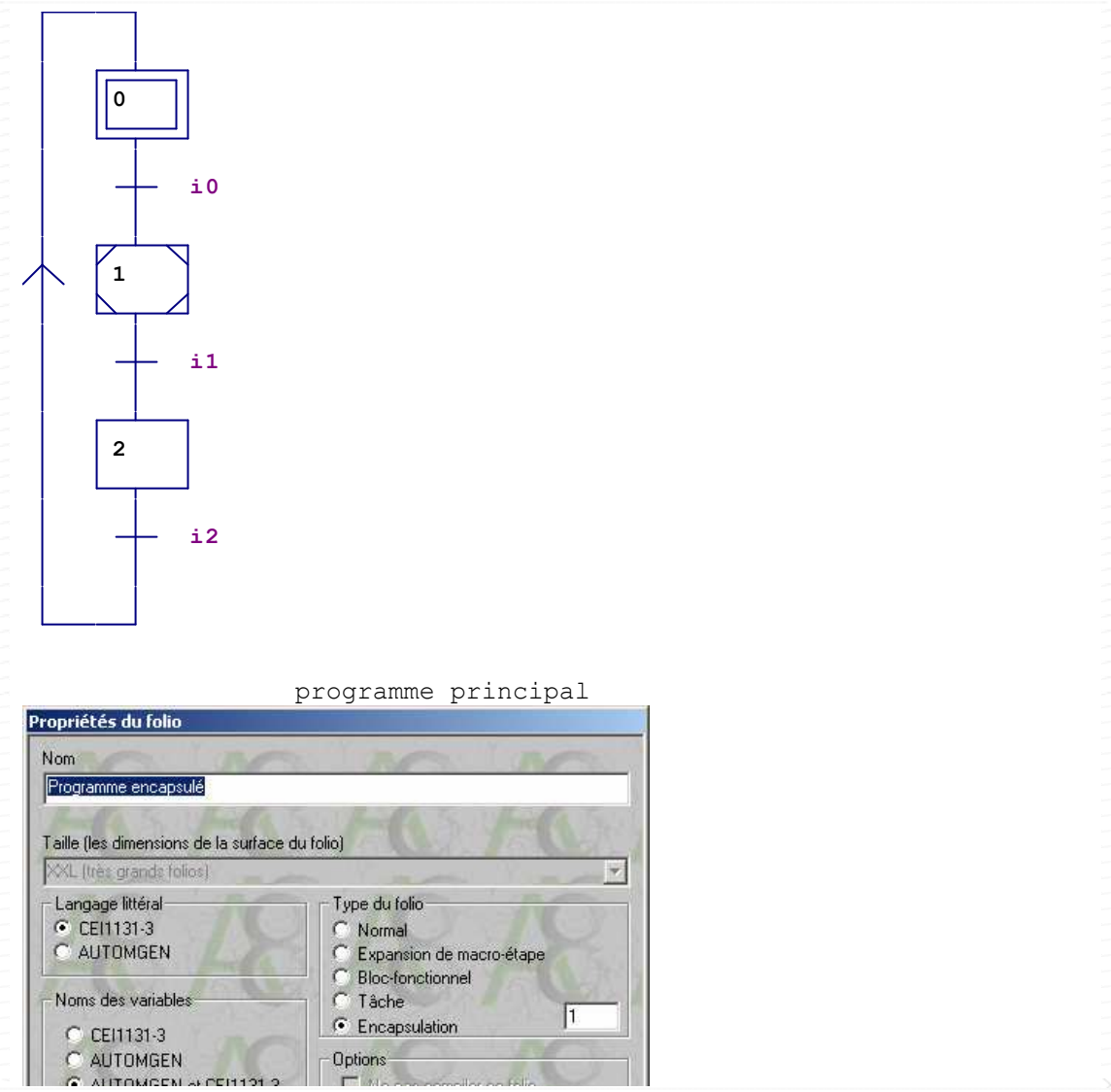
Remarques :

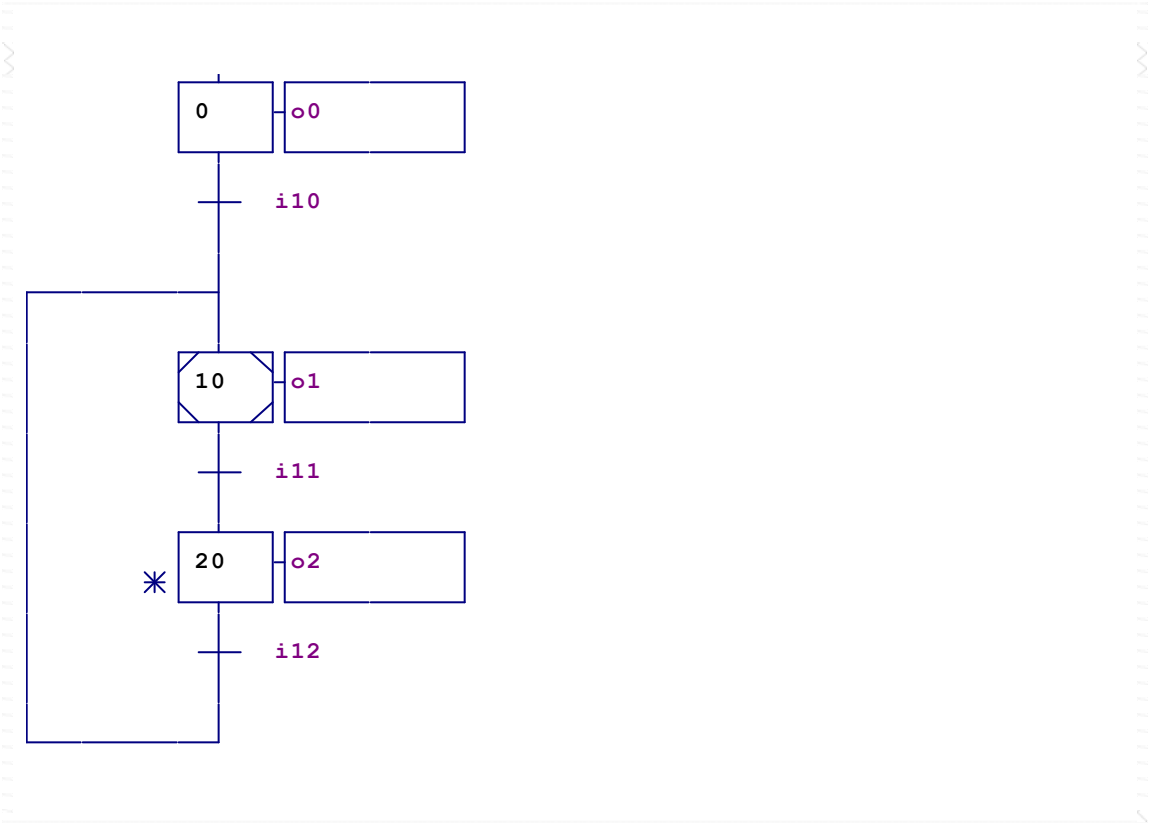
⇨ les étapes et les bits Utilisateur utilisés dans une encapsulation sont locaux, c'est à dire qu'ils n'ont aucun rapport avec les étapes et les bits d'autres niveaux de Grafcets. Tous les autres types de variables n'ont pas cette caractéristique : ils sont communs à tous les niveaux. Vous pouvez par exemple utiliser des bits de mots comme variables globales.

⇨ Les étapes encapsulantes peuvent être imbriquées.

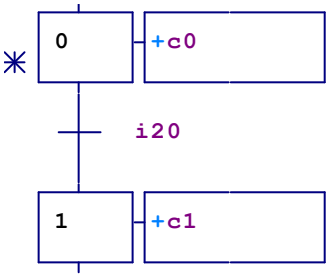
⇨ La syntaxe Xn/Xm ou %Xn/%Xm permet de faire référence à l'étape m contenue dans l'encapsulation associée à l'étape n.

Exemple :



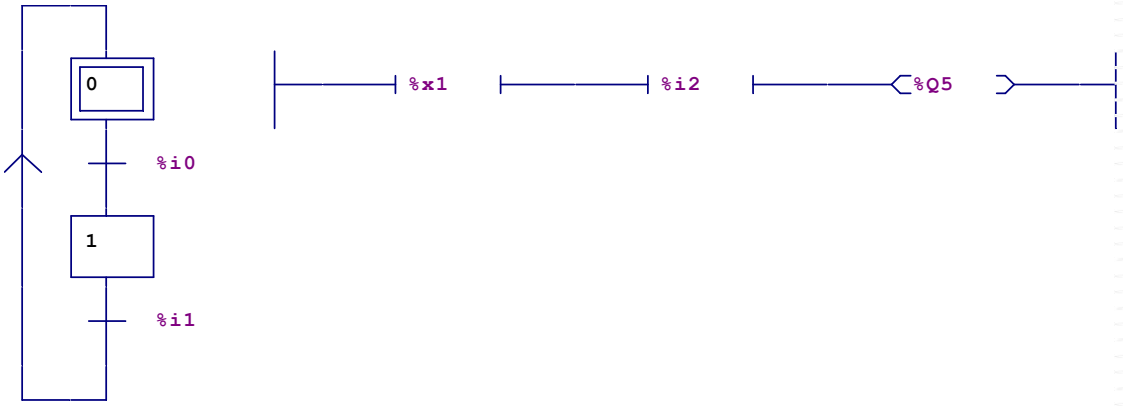


exemples\grafcet\encapsulation 2.agn



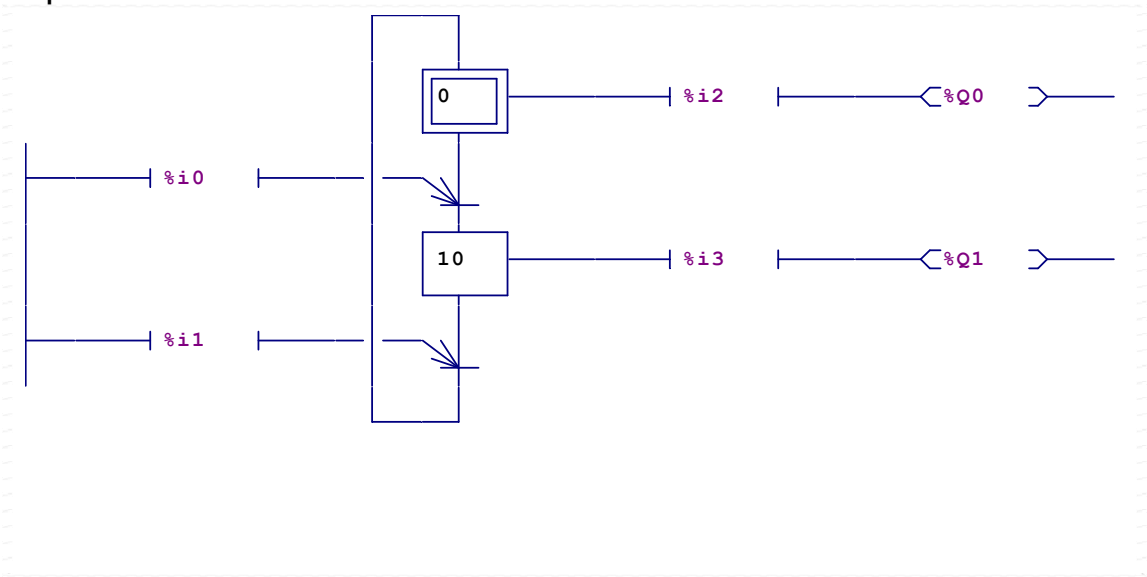
Liens Grafcet Ladder, Grafcet Logigrammes

Les liens peuvent être réalisés par les variables d'étapes du Grafcet.



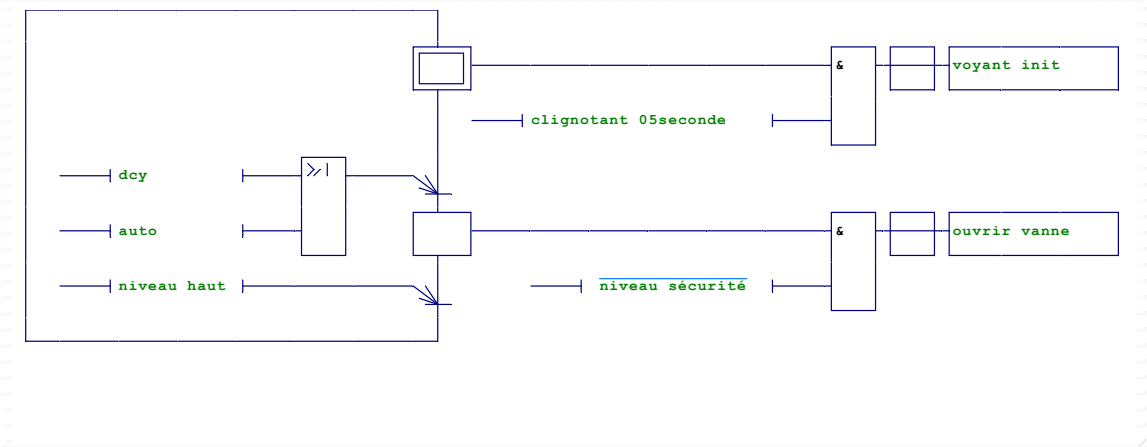
GRAFCET++ : le bloc peut être utilisé pour câbler une transition comme une bobine de ladder. Les étapes Grafcet peuvent être câblées comme le début d'un contact.

Exemple Grafcet / ladder :



exemples\grafcet\grafcet++2.agn

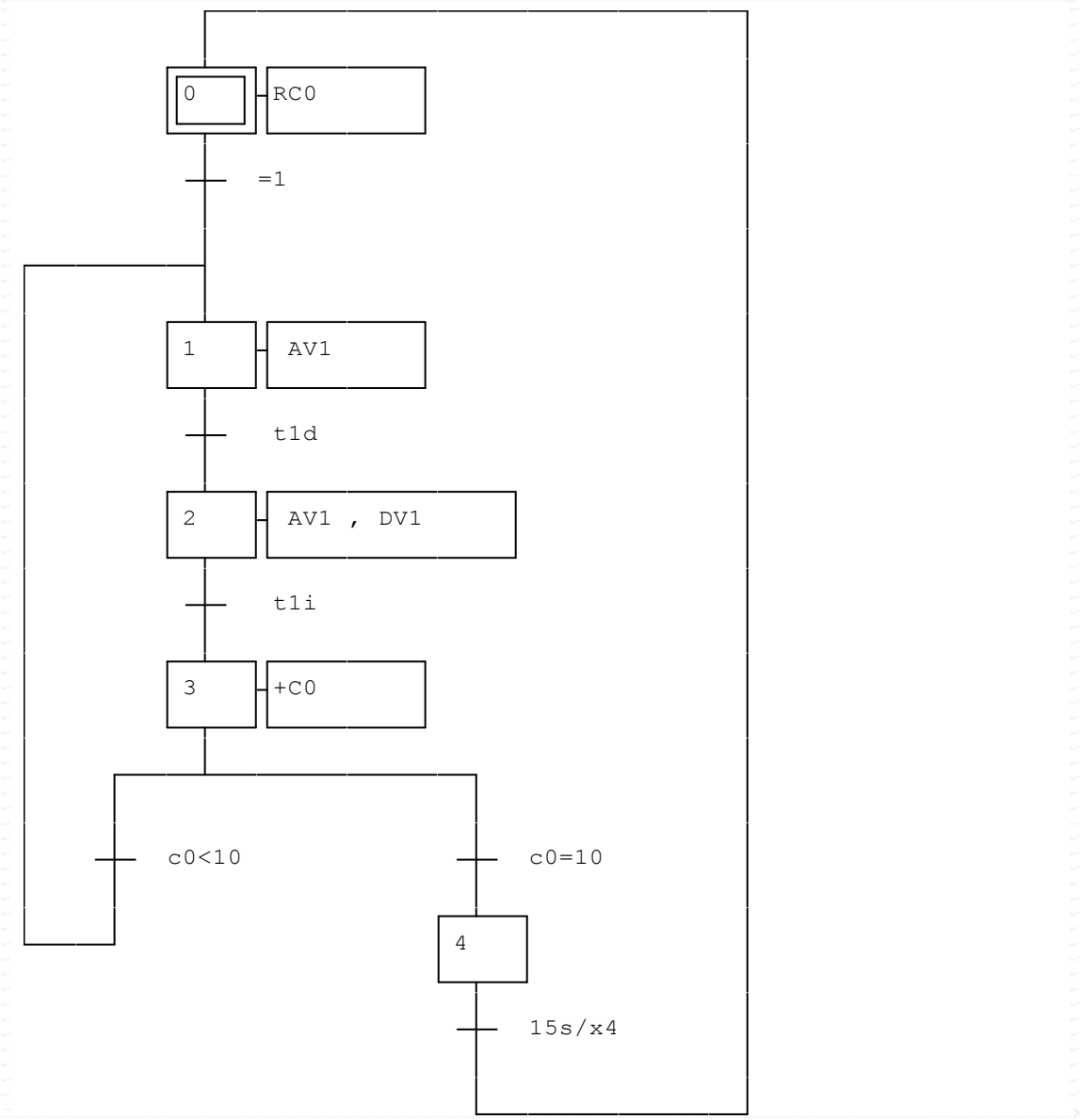
Exemple Grafcet logigrammes :



exemples\grafcet\grafcet++.agn

Compteurs

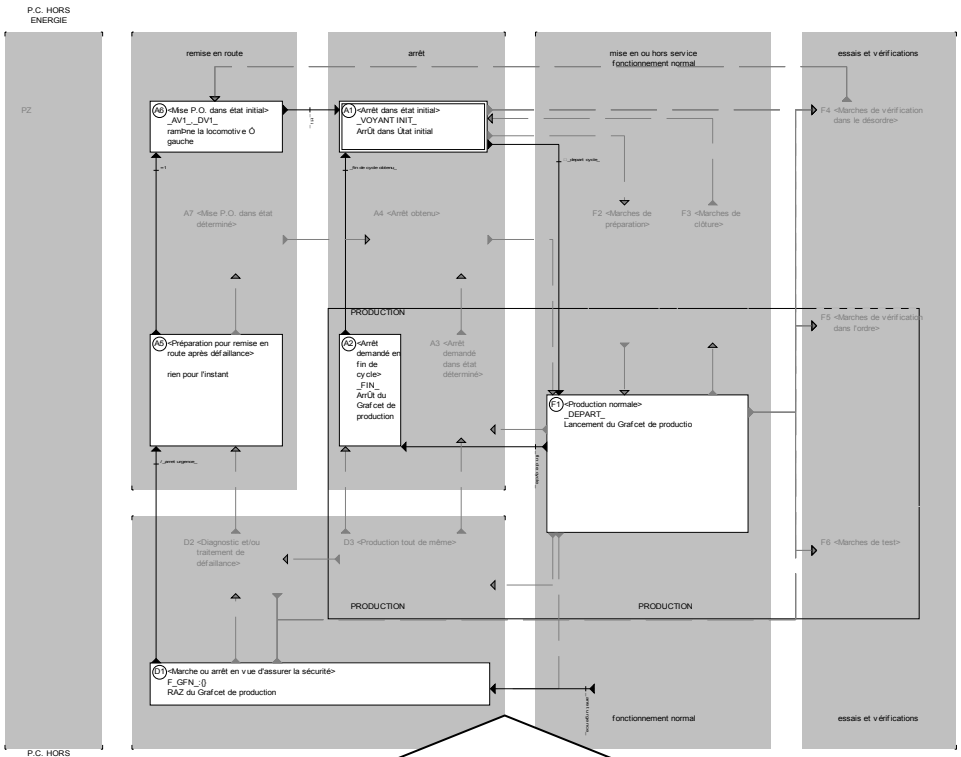
Illustrons l'utilisation des compteurs par un exemple.
Cahier des charges :
Une locomotive doit effectuer 10 allers et retours sur la voie 1, s'immobiliser pendant quinze secondes et recommencer.
Solution :

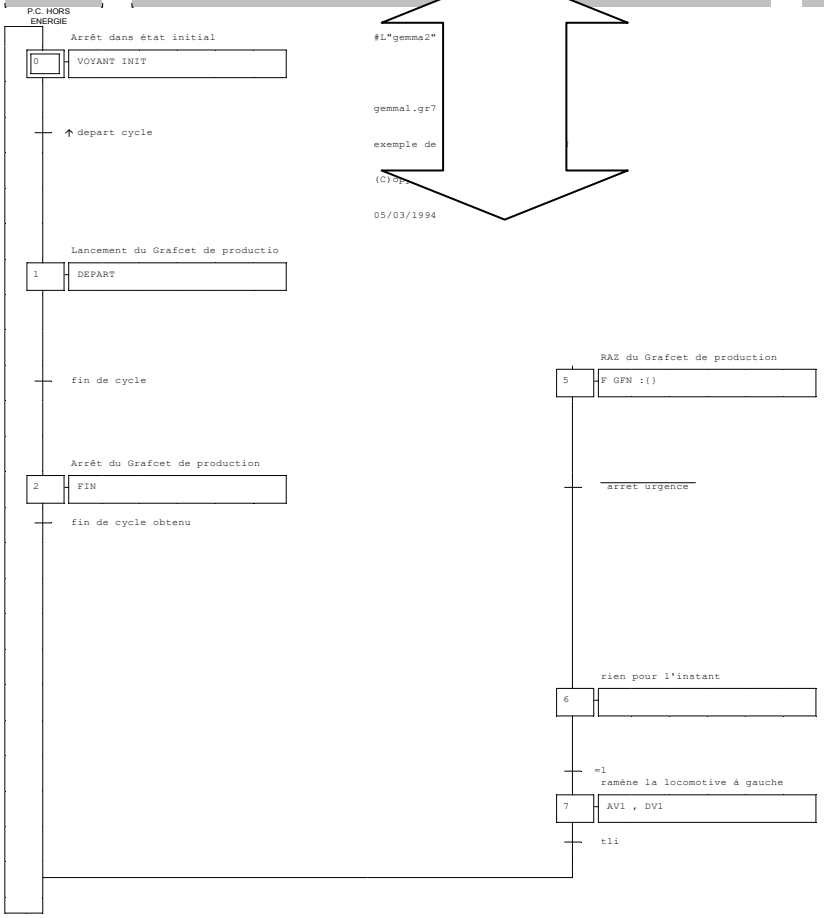


exemples\grafcet\compteur.agn

Gemma

AUTOMGEN implémente la description de Grafcet de gestion des modes de marche sous forme de Gemma. Le principe retenu est un mode d'édition transparent au mode Grafcet. Il est possible de passer du mode d'édition Grafcet au mode d'édition Gemma. La traduction d'un Gemma en Grafcet de gestion des modes de marche est donc automatique et immédiate.


P.C. HORS ENERGIE
PZ
remise en route
arrêt
mise en ou hors service
fonctionnement normal
essais et vérifications
A6 <Mise P.O. dans état initial>
_AV1_,_DV1_
ramène la locomotive à gauche
A1 <Arrêt dans état initial>
VOYANT INIT
Arrêt dans l'état initial
F4 <Marches de vérification dans le désordre>
A7 <Mise P.O. dans état déterminé>
A4 <Arrêt obtenu>
F2 <Marches de préparation>
F3 <Marches de clôture>
PRODUCTION
F5 <Marches de vérification dans l'ordre>
A5 <Préparation pour remise en route après défaillance>
rien pour l'instant
A2 <Arrêt demandé en fin de cycle>
FIN
Arrêt du Grafcet de production
A3 <Arrêt demandé dans état déterminé>
F1 <Production normale>
DEPART
Lancement du Grafcet de productio
D2 <Diagnostic et/ou traitement de défaillance>
D3 <Production tout de même>
F6 <Marches de test>
PRODUCTION
PRODUCTION
D1 <Marche ou arrêt en vue d'assurer la sécurité>
F_GFN_:{}
RAZ du Grafcet de production
fonctionnement normal
essais et vérifications


P.C. HORS ENERGIE
Arrêt dans état initial
0
VOYANT INIT
depart cycle
#L"gemma2"
gemmal.gr7
exemple de
(C)dp
05/03/1994
Lancement du Grafcet de productio
1
DEPART
fin de cycle
RAZ du Grafcet de production
5
F GFN :{}
Arrêt du Grafcet de production
2
FIN
fin de cycle obtenu
arret urgence
rien pour l'instant
6
=1
ramène la locomotive à gauche
7
AV1 , DV1
tli

Création d'un Gemma

Pour créer un Gemma il faut :

⇨ cliquer sur l'élément « Folio » du navigateur avec le bouton droit de la souris puis sélectionner la commande « Ajouter un nouveau folio »,

⇨ dans la liste des tailles choisir « Gemma »,

⇨ cliquer sur le bouton poussoir « OK »,

⇨ cliquer avec le bouton droit de la souris sur le nom du folio ainsi créé dans le navigateur,

⇨ choisir « Propriétés » dans le menu,

⇨ cocher la case « Afficher sous la forme d'un Gemma ».

La fenêtre contient alors un Gemma dont tous les rectangles et tous les liens sont grisés. Pour valider un rectangle ou une liaison il faut cliquer dessus avec le bouton droit de la souris.
Pour modifier le contenu d'un rectangle ou la nature d'une liaison il faut cliquer dessus avec le bouton gauche de la souris.
Le contenu des rectangles du Gemma sera placé dans des rectangles d'action du Grafcet. La nature des liaisons sera placée dans les transitions du Grafcet.
Un commentaire peut être associé à chaque rectangle du Gemma, il apparaîtra à proximité du rectangle d'action correspondant dans le Grafcet.

⇨ La combinaison de touche [CTRL] + [G] permet de basculer du mode GEMMA au mode Grafcet ou du mode Grafcet au mode GEMMA.

Contenu des rectangles du Gemma

Les rectangles du Gemma peuvent recevoir n'importe quelle action utilisable dans le Grafcet. Comme il s'agit de définir une structure de gestion des modes d'arrêt et de marche, il parait judicieux d'utiliser des ordres de forçage vers des Grafcets de plus bas niveau voir le chapitre Forçages de Grafcet.

Obtenir un Grafcet correspondant au Gemma

C'est encore la case à cocher « Afficher sous la forme d'un Gemma » dans les propriétés du folio qui permet de revenir à une représentation Grafcet. Il est possible de revenir à tout moment à une représentation Gemma tant que la structure du Grafcet n'a pas été modifiée. Les transitions, le contenu des rectangles d'action et les commentaires peuvent être modifiés avec une mise à jour automatique du Gemma.

Annuler les espaces vides dans le Grafcet

Il est possible que le Grafcet obtenu occupe plus d'espace que nécessaire sur la page. La commande « Supprimer les lignes et les colonnes vides du folio » du menu « Outils » permet de supprimer tous les espaces non utilisés.

Imprimer le Gemma

Quand l'édition est en mode Gemma c'est la commande « Imprimer » qui permet d'imprimer le Gemma.

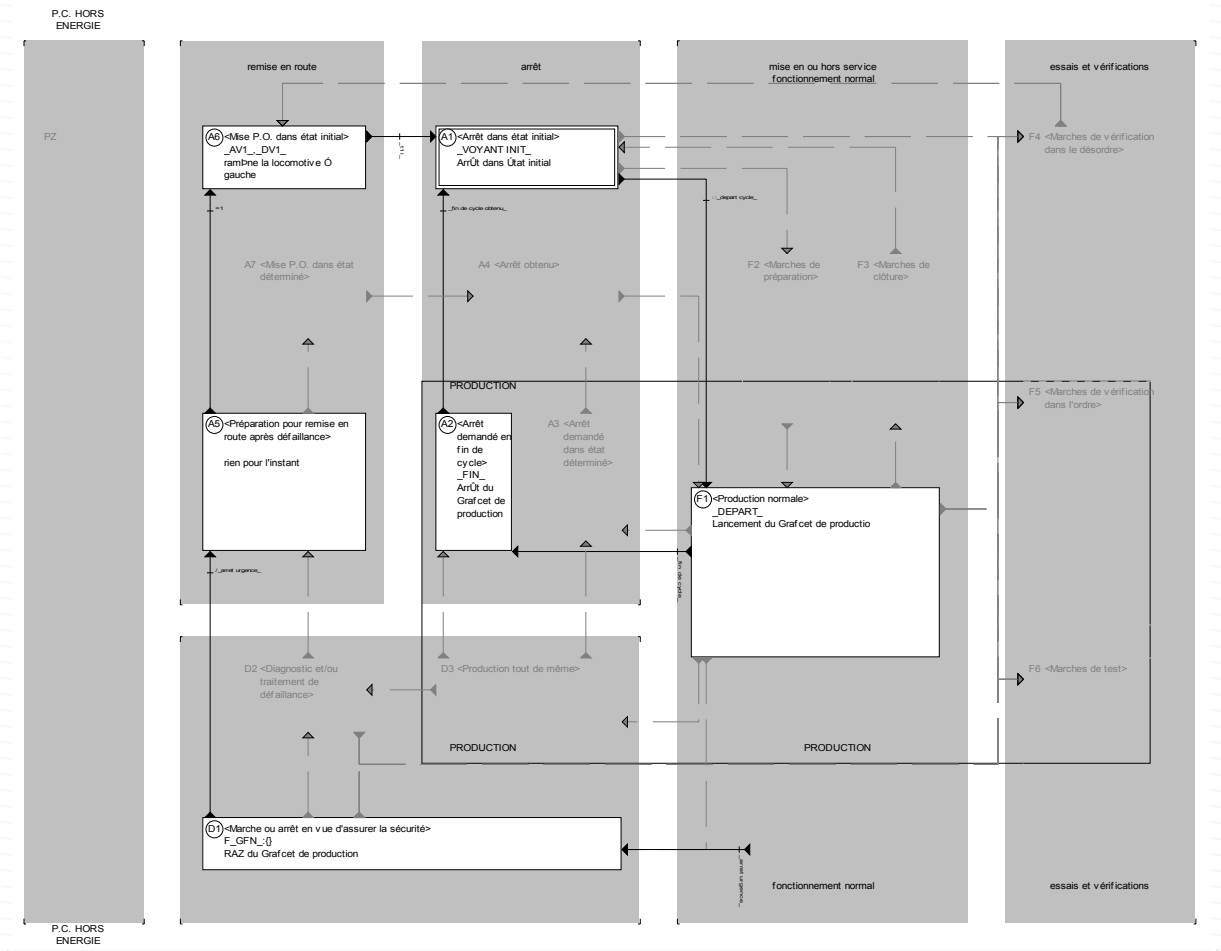
Exporter le Gemma

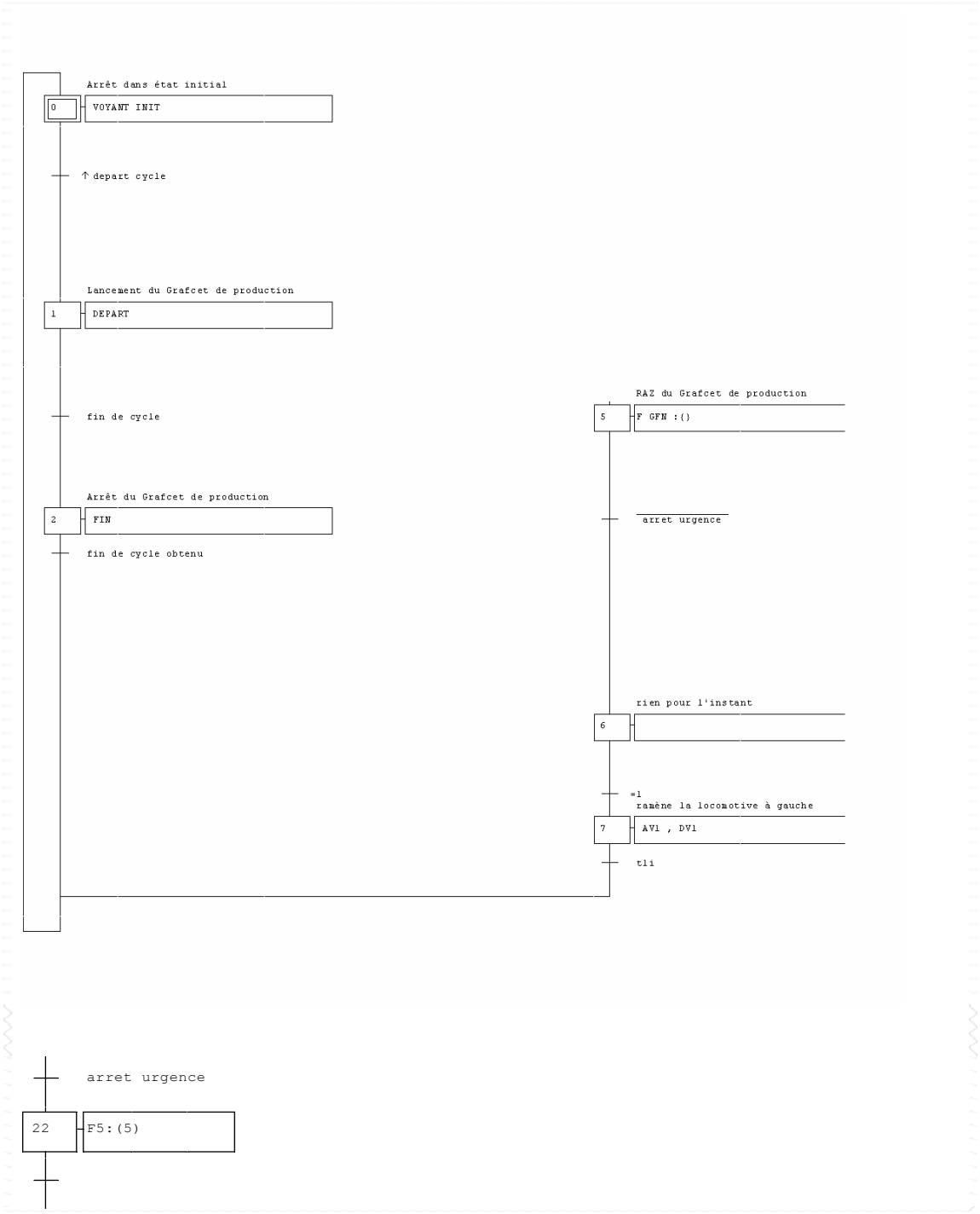
La commande « Copier au format EMF » du menu « Edition » permet d'exporter un Gemma au format vectoriel.

Exemple de Gemma

Illustrons l'utilisation du Gemma.
Cahier des charges :
Imaginons un pupitre composé des boutons poussoirs suivants : « départ cycle », « fin de cycle » et « arrêt d'urgence » et d'un voyant « INIT ».
Le programme principal consistera à faire effectuer des allers et retours à une locomotive sur la voie 1.

Solution :



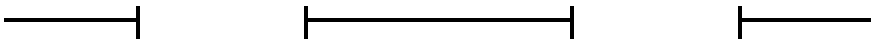


(édité sous la forme d'un Grafcet)

exemples\gemma\gemma.agn

Ladder

Le langage Ladder, également appelé « schéma à contact », permet de décrire graphiquement des équations booléennes. Pour réaliser une fonction logique « Et », il faut écrire des contacts en série. Pour écrire une fonction « Ou », il faut écrire des contacts en parallèle.



Fonction « Et »

Fonction « Ou »

Le contenu des contacts doit respecter la syntaxe définie pour les tests et détaillée dans le chapitre « Eléments communs » de ce manuel.

Le contenu des bobines doit respecter la syntaxe définie pour les actions, elle aussi détaillée dans le chapitre « Eléments communs » de ce manuel.

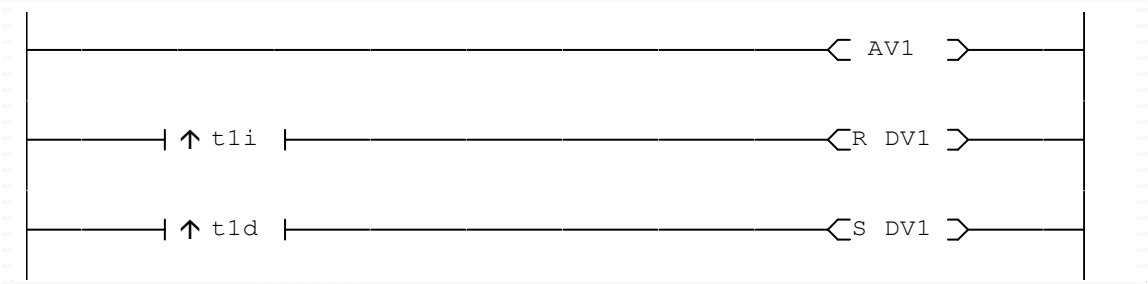
Exemple de Ladder

Commençons par l'exemple le plus simple.

Cahier des charges :

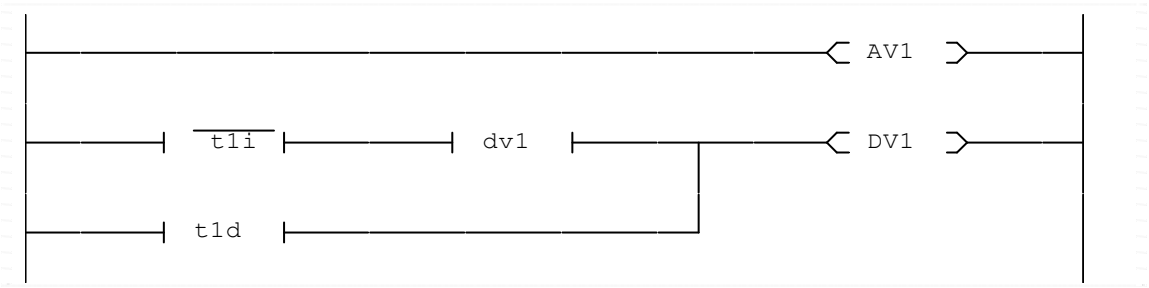
Aller et retour d'une locomotive sur la voie 1.

Solution 1 :



exemples\ladder\ladder1.agn

Solution 2 :

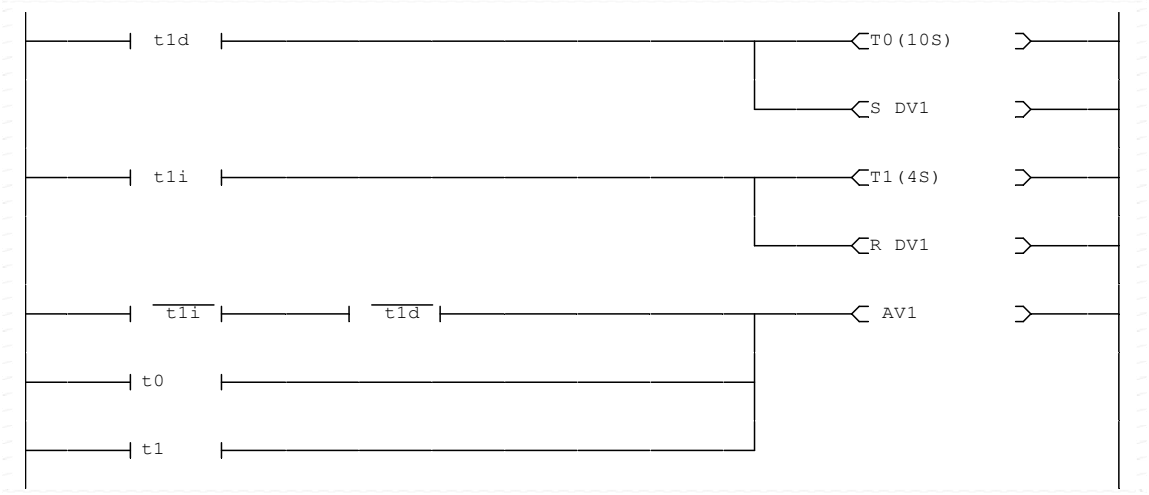


exemples\ladder\ladder2.agn

Cette deuxième solution est identique d'un point de vue fonctionnel. Son intérêt est de montrer l'utilisation d'une variable en auto maintien.

Enrichissons notre exemple.
Cahier des charges :
La locomotive devra s'arrêter 10 secondes à droite de la voie 1 et 4 secondes à gauche.

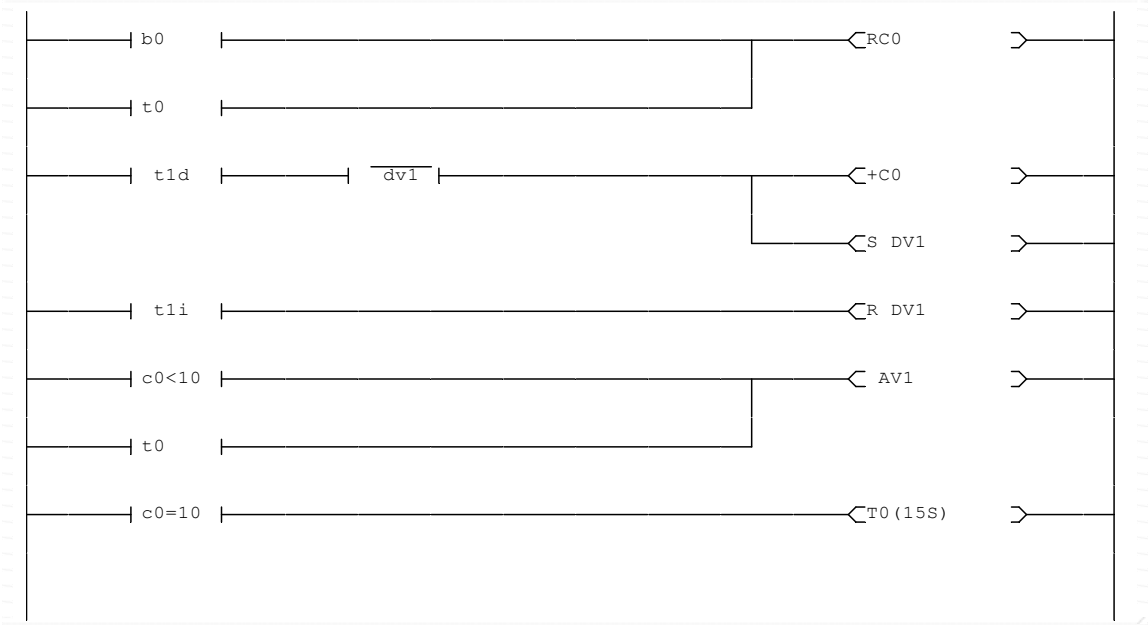
Solution :



exemples\ladder\ladder3.agn

Un dernier exemple un peu plus complexe.
Cahier des charges :
Toujours une locomotive qui fait des allers et retours sur la voie 1. Tous les 10 allers et retours elle devra marquer un temps d'arrêt de 15 secondes.

Solution :



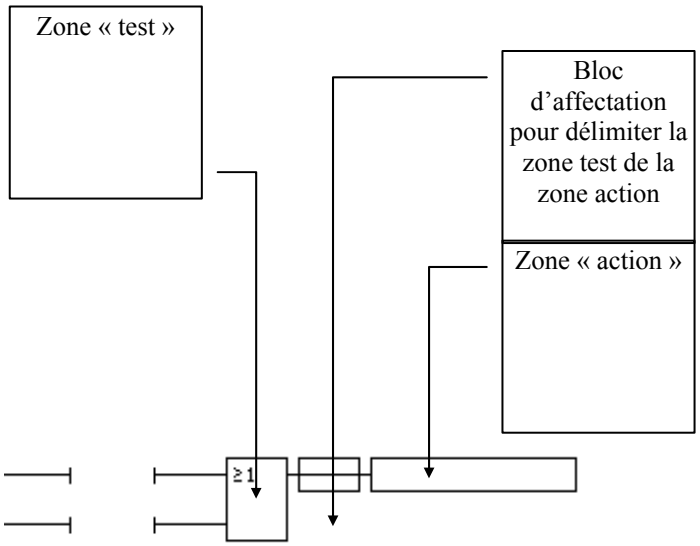
exemples\ladder\ladder4.agn

Logigramme

AUTOMGEN implémente le langage logigramme de la façon suivante :

- utilisation d'un bloc spécial nommé « bloc d'affectation », ce bloc sépare la zone d'action de la zone test, il a la forme suivante  et est associé à la touche [0] (zéro),
- utilisation des fonctions « Pas », « Et » et « Ou »,
- utilisation de rectangles d'action à droite du bloc d'action.

Le langage logigramme permet d'écrire graphiquement des équations booléennes.
Le contenu des tests doit respecter la syntaxe définie dans le chapitre « Eléments communs » de ce manuel.
Le contenu des rectangles d'action doit respecter la syntaxe définie pour les actions et détaillée dans le chapitre « Eléments communs » de ce manuel.

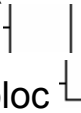


Dessin des logigrammes

Nombre d'entrées des fonctions « Et » et « Ou »

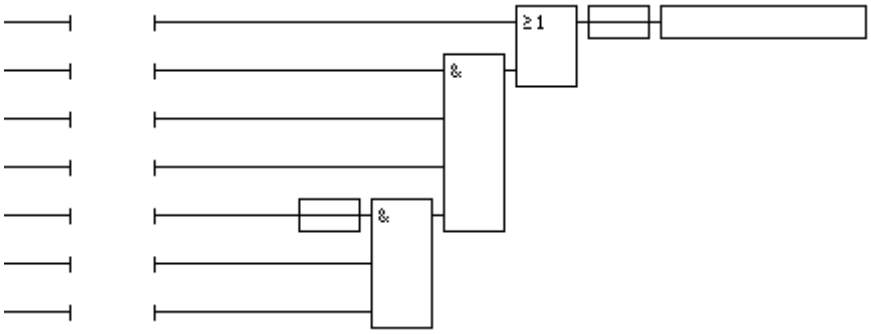
Les fonctions « Et » et « Ou » se composent respectivement d'un bloc 

(touche [2]) ou d'un bloc 

(touche [3]), éventuellement de blocs  (touche [4]) pour ajouter des entrées aux blocs et en fin d'un bloc  (touche [5]).
Les fonctions « Et » et « Ou » comportent donc un minimum de deux entrées.

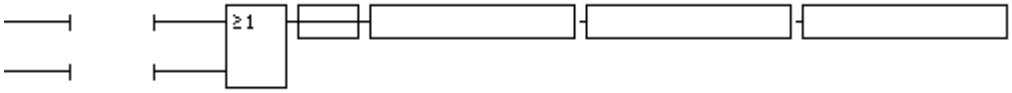
Enchaînement des fonctions

Les fonctions peuvent être chaînées.

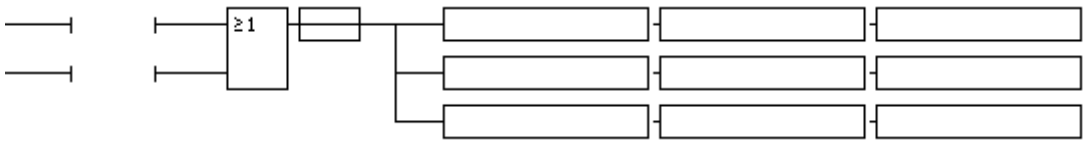


Actions multiples

Plusieurs rectangles d'action peuvent être associés à un logigramme après le bloc d'affectation.

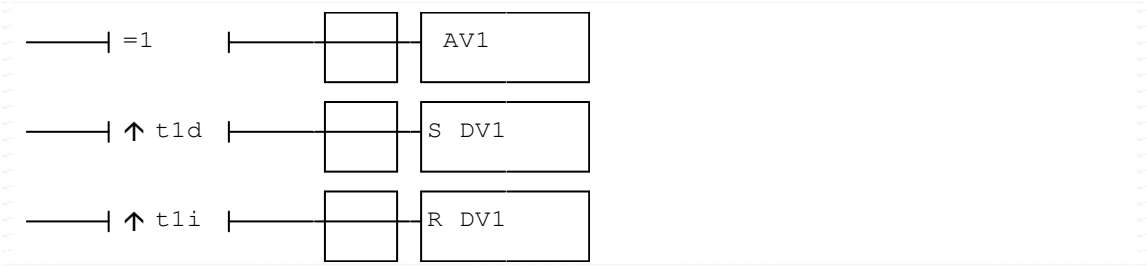


ou



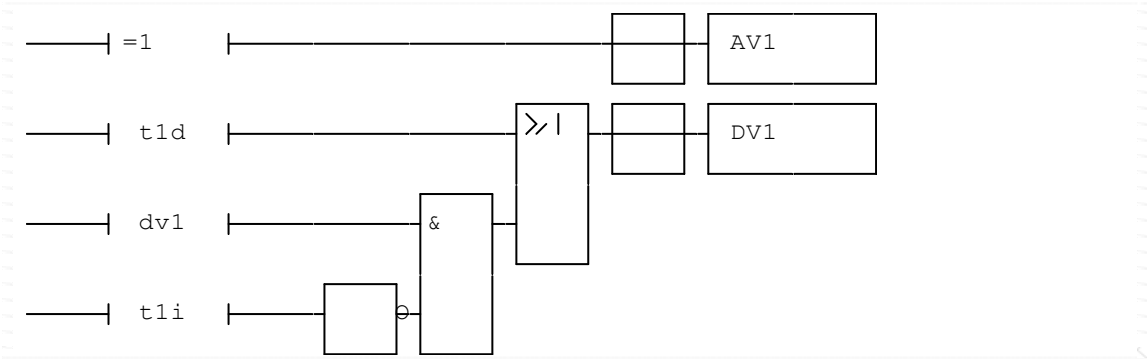
Exemples de logigramme

Commençons par l'exemple le plus simple.
Cahier des charges :
Aller et retour d'une locomotive sur la voie 1.
Solution 1 :



exemples\logigramme\logigramme1.agn

Solution 2 :

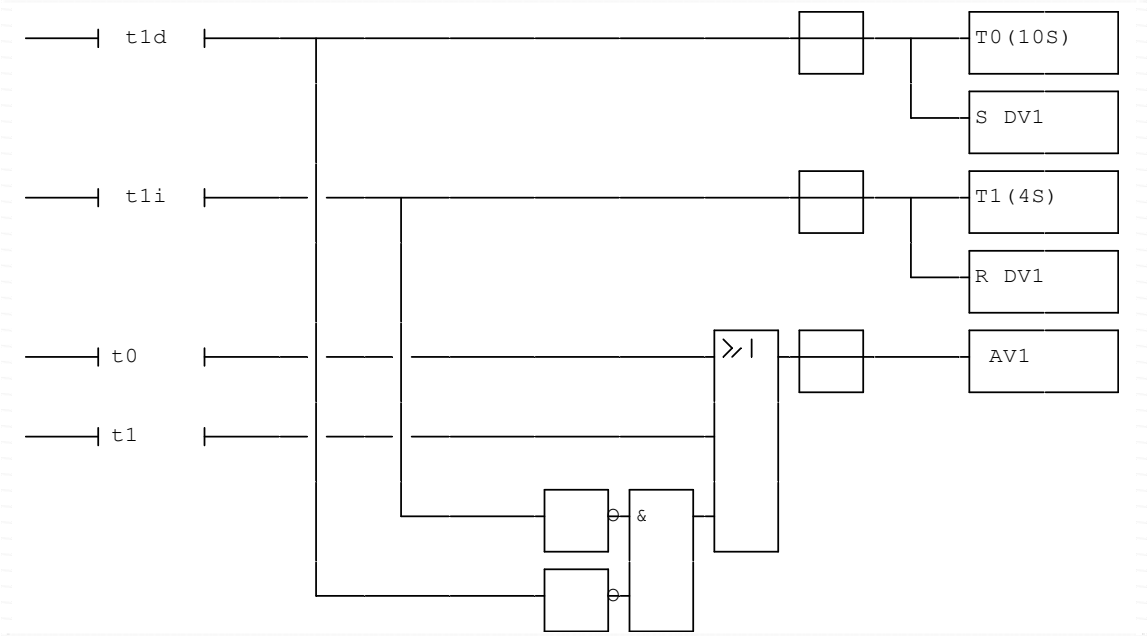


exemples\logigramme\logigramme2.agn

Cette deuxième solution est identique d'un point de vue fonctionnel. Son intérêt consiste à montrer l'utilisation d'une variable en auto maintien.

Enrichissons notre exemple.
Cahier des charges :
La locomotive devra s'arrêter 10 secondes à droite de la voie 1et 4 secondes à gauche.

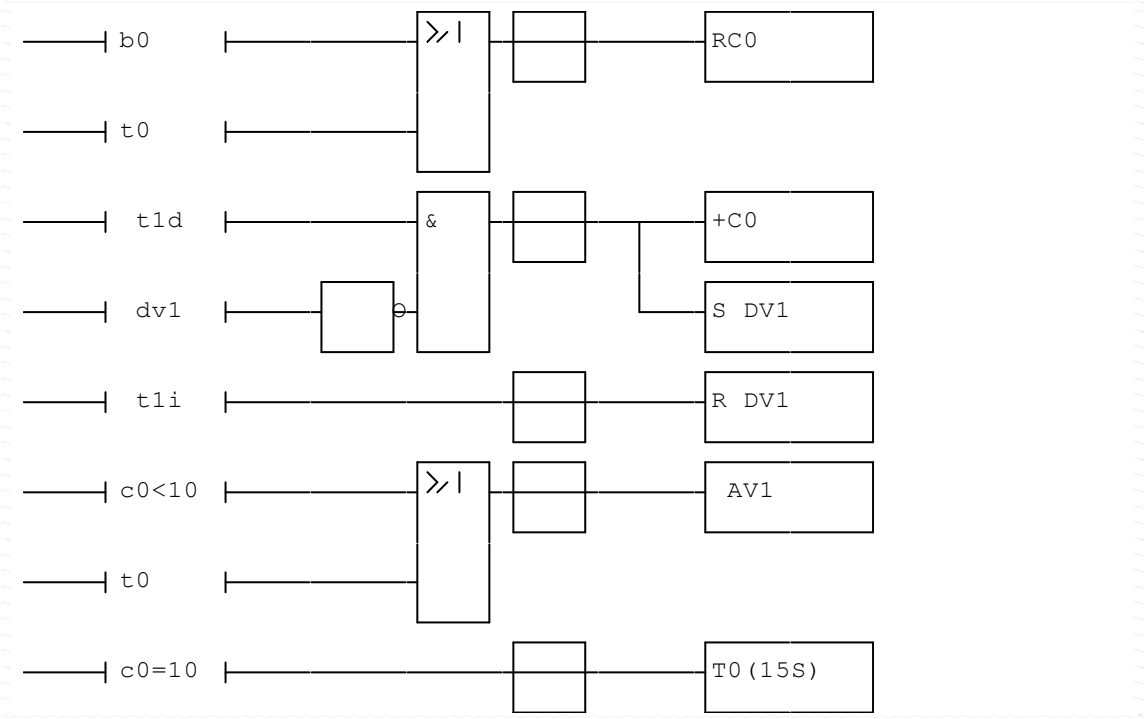
Solution :



exemples\logigramme\logigramme3.agn

Notez le repiquage du bloc « Et » du bas de l'exemple vers les entrées « _t1d_ » et « _t1i_ ». Cela évite d'avoir à écrire une deuxième fois ces deux tests.
Un dernier exemple un peu plus complexe.
Cahier des charges :
Toujours une locomotive qui fait des allers et retours sur la voie 1. Tous les 10 allers et retours, elle devra marquer un temps d'arrêt de 15 secondes.

Solution :



exemples\logigramme\logigramme4.agn

Langages littéraux

Ce chapitre décrit l'utilisation les trois formes de langages littéraux disponibles dans AUTOMGEN :

- le langage littéral bas niveau,
- le langage littéral étendu,
- le langage littéral ST de la norme CEI 1131-3.

Comment utiliser le langage littéral ?

Le langage littéral peut être utilisé sous la forme suivante :

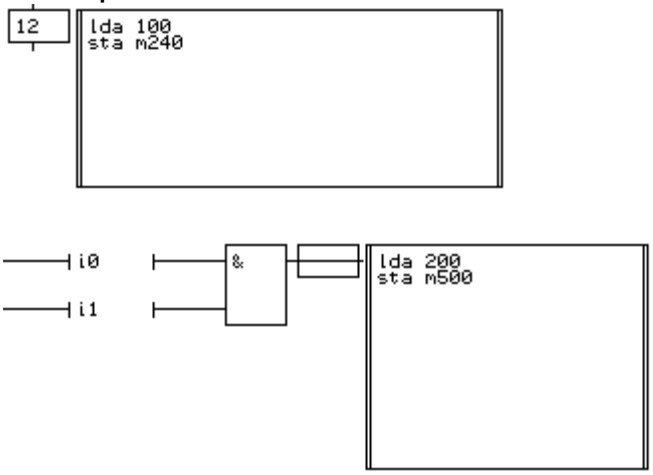
- fichier de code associé à une action (Grafcet, Ladder, logigramme),
- boîte de code associée à une action (Grafcet, logigramme),
- code littéral dans un rectangle d'action ou une bobine (Grafcet, Ladder, Logigramme),
- boîte de code utilisée sous forme d'organigramme (consultez le chapitre « Organigramme »),
- fichier de code régissant le fonctionnement d'un bloc fonctionnel (consultez le chapitre « Blocs fonctionnels »),

⇨ fichier de code régissant le fonctionnement d'une macro-instruction voir chapitre Macro-instruction.

Boîte de code associée à une étape ou un logigramme

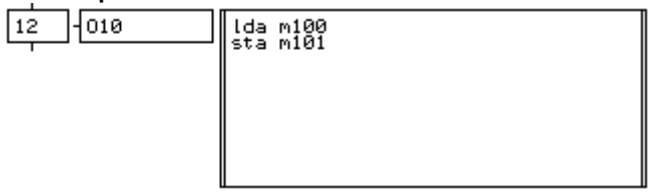
Une boîte de code associée à une action permet d'écrire quelques lignes de langage littéral au sein d'une page de l'application.

Exemples :



Le code ainsi utilisé est scruté tant que l'action est vraie.
Il est possible d'utiliser conjointement des rectangles d'action et des boîtes de code.

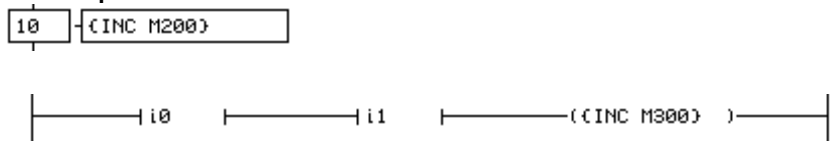
Exemple :

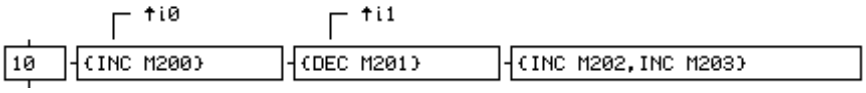


Code littéral dans un rectangle d'action ou une bobine

Les caractères « { » et « } » permettent d'insérer directement des instructions en langage littéral dans un rectangle d'action (langages Grafcet et logigramme) ou une bobine (langage ladder). Le caractère « , » (virgule) est utilisé comme séparateur si plusieurs instructions sont présentes entre « { » et « } ».
Ce type d'insertion peut être utilisé avec des ordres conditionnés.

Exemples :





Définition d'une boîte de code

Pour dessiner une boîte de code, suivez les étapes suivantes :

⇨ cliquez avec le bouton droit de la souris sur un emplacement vide du folio,

⇨ choisissez dans le menu « Plus … / Boîte de code »,

⇨ cliquez sur le bord de la boîte de code pour modifier son contenu.

Pour sortir de la boîte après modification, utilisez la touche [Enter] ou cliquez à l'extérieur.

Le langage littéral bas niveau

Ce chapitre détaille l'utilisation du langage littéral bas niveau. Ce langage est un code intermédiaire entre les langages évolués Grafcet, logigramme, ladder, organigramme, blocs fonctionnels, littéral étendu, littéral ST et les langages exécutables. Il est également connu sous le nom de code pivot. Ce sont les post-processeurs qui traduisent le langage littéral bas niveau en code exécutable pour PC, automate ou carte à microprocesseur.
Le langage littéral peut également être utilisé au sein d'une application pour effectuer divers traitements booléens, numériques ou algorithmiques.
Le langage littéral bas niveau est un langage de type assembleur. Il utilise une notion d'accumulateur pour les traitements numériques.
Le langage littéral étendu et le langage littéral ST décrits dans les chapitres suivants, offrent une alternative simplifiée et de plus haut niveau pour l'écriture de programmes en langage littéral.
Syntaxe générale d'une ligne de langage littéral bas niveau :
«action » [[[« Test »] « Test »]...]
Les actions et les tests du langage littéral bas niveau sont représentés par des mnémoniques formés de trois caractères alphabétiques. Une instruction est toujours suivie d'une expression : variable, constante, etc...
Une ligne est composée d'une seule action et éventuellement d'un test. Si une ligne comporte uniquement une action, alors par convention l'instruction sera toujours exécutée.

Les variables

Les variables utilisables sont les mêmes que celles définies dans le chapitre « Eléments communs ».

Les accumulateurs

Certaines instructions utilisent la notion d'accumulateur. Les accumulateurs sont des registres internes au système qui exécute le programme final et permettent d'accueillir temporairement des valeurs.
Il existe trois accumulateurs : un accumulateur entier 16 bits noté AAA, un accumulateur entier 32 bits noté AAL et un accumulateur flottant noté AAF.

Les drapeaux

Les drapeaux sont des variables booléennes positionnées en fonction du résultat des opérations numériques.
Il y a quatre drapeaux permettant de tester le résultat d'un calcul. Ces quatre indicateurs sont :

- ⇨ indicateur de retenue C : il indique si une opération a engendré une retenue (1) ou n'a pas engendré de retenue (0),
- ⇨ indicateur de zéro Z : il indique si une opération a engendré un résultat nul (1) ou non nul (0),
- ⇨ indicateur de signe S : il indique si une opération a engendré un résultat négatif (1) ou positif (0),
- ⇨ indicateur de débordement O : il indique si une opération a généré un dépassement de capacité (1).

Modes d'adressage

Le langage littéral bas niveau possède 5 modes d'adressage. Un mode d'adressage est une caractéristique associée à chacune des instructions du langage littéral.
Les modes d'adressage utilisables sont :

TYPE	SYNTAXE	EXEMPLE
Immédiat 16 bits	{constante}	100
Immédiat 32 bits	{constante}L	100000L
Immédiat flottant	{constante}R	3.14R
Absolu	{variable} {repère de variable}	O540
Accumulateur 16 bits	AAA	AAA
Accumulateur 32 bits	AAL	AAL
Accumulateur flottant	AAF	AAF

Indirect	{variable}{(repère de mot)}	O(220)
Label	:{nom de label} :	:boucle:

Une instruction possède donc deux caractéristiques : le type de variable et le mode d'adressage. Certaines instructions supportent ou ne supportent pas certains modes d'adressage ou certains types de variables. Par exemple une instruction pourra s'appliquer uniquement à des mots et pas aux autres types de variables.

Remarque : Les variables X et U ne peuvent être associées à un adressage indirect du fait de la non linéarité de leurs affectations. Si toutefois il est nécessaire d'accéder à un tableau de variables U, alors il faut utiliser une directive de compilation #B pour déclarer une table de bits linéaires.

Les tests

Les tests pouvant être associés aux instructions sont composés d'un mnémonique, d'un type de test et d'une variable.
Les mnémoniques de tests permettent de définir des tests combinatoires sur plusieurs variables (et, ou). Si un test est composé d'une seule variable, un opérateur AND doit quand même lui être associé.
Il existe seulement trois mnémoniques de test :
AND et
ORR ou
EOR fin de ou
Voici quelques exemples d'équivalences entre des équations booléennes et le langage littéral bas niveau :

```
o0=i1                          : and i1
o0=i1.i2                       : and i1 and i2
o0=i1+i2                       : orr i1 eor i2
o0=i1+i2+i3+i4                 : orr i1 orr i2 orr i3 eor i4
o0=(i1+i2).(i3+i4)             : orr i1 eor i2 orr i3 eor i4
o0=i1.(i2+i3+i4)               : and i1 orr i2 orr i3 eor i4
o0=(i1.i2)+(i3.i4)             ; impossible à traduire directement,
                               ; il faut utiliser des
                               ; variables intermédiaires :

                    equ u100 and i1 and i2
                           equ u101 and i3 and i4
                           equ o0 orr u100 eor u101
```


Les modificateurs de tests permettent de tester autre chose que l'état vrai d'une variable :

⇨ / pas

⇨ # front montant

⇨ * front descendant

⇨ @ état immédiat

Remarques :

⇨ les variables booléennes sont rafraîchies après chaque cycle d'exécution. En d'autres termes, si une variable binaire est positionnée à un état au cours d'un cycle, alors son nouvel état sera réellement reconnu au cycle suivant. Le modificateur de test @ permet d'obtenir l'état réel d'une variable booléenne sans attendre le cycle suivant.

⇨ les modificateurs de tests ne sont pas utilisables avec les tests numériques.

Exemples:

```
set o100
equ o0 and @o100          ; test vrai dès le premier cycle
equ o1 and o100           ; test vrai au deuxième cycle
```

Pour les tests, seuls deux modes d'adressage sont disponibles : le mode absolu et le mode indirect.
Un test sur compteurs, sur mots, sur longs et sur flottants est disponible :

Syntaxe :

```
« {variable} {=, !, <, >, << , >>} {constante ou variable} »
```

= signifie égal,
! signifie différent,
< signifie inférieur non signé,
> signifie supérieur non signé,
<< signifie inférieur signé,
>> signifie supérieur signé,
Les constantes sont écrites par défaut en décimal. Les suffixes « $ » et « % » permettent de les écrire en hexadécimal ou en binaire. Les guillemets permettent de les écrire en ASCII.
Les constantes 32 bits doivent être suivies du caractère « L ».
Les constantes réelles doivent être suivies du caractère « R ».

Un mot ou un compteur peut être comparé avec un mot, un compteur ou une constante 16 bits.
Un long peut être comparé avec un long ou une constante 32 bits.
Un flottant peut être comparé avec un flottant ou une constante réelle.

Exemples :

```
and c0>100 and m225=10
orr m200=m201 eor m202=m203 and f100=f101 and f200<f203
orr m200<<-100 eor m200>>200
and f200=3.14r
and l200=$12345678L
and m200=%1111111100000000
```

Commentaires

Les commentaires doivent débuter par le caractère « ; » (point virgule), tous les caractères venant à la suite sont ignorés.

Base de numérotation

Les valeurs (repères de variables ou constantes) peuvent être écrites en décimal, hexadécimal, binaire ou en ASCII.
La syntaxe suivante doit être appliquée pour les constantes 16 bits :

⇨ décimal : éventuellement le caractère « - » puis 1 à 5 digits « 0123456789 »,

⇨ hexadécimal : le préfixe « $ » ou « 16# » suivi de 1 à 4 digits « 0123456789ABCDEF »,

⇨ binaire : le préfixe « % » ou « 2# » suivi de 1 à 16 digits « 01 »,

⇨ ASCII : le caractère « " » suivi de 1 ou 2 caractères suivis de « " ».

La syntaxe suivante doit être appliquée pour les constantes 32 bits :

⇨ Décimal : éventuellement le caractère « - » puis 1 à 10 digits « 0123456789 »,

⇨ Hexadécimal : le préfixe « $ » ou « 16# » suivi de 1 à 8 digits « 0123456789ABCDEF »,

⇨ Binaire : le préfixe « % » ou « 2# » suivi de 1 à 32 digits « 01 »,

⇨ ASCII : le caractère « " » suivi de 1 à 4 caractères suivis de « " ».

La syntaxe suivante doit être appliquée pour les constantes réelles :
[-] i [[.d] Esx]
i est la partie entière
d une éventuelle partie décimale
s éventuellement le signe de l'exposant
x éventuellement l'exposant

Prédispositions

Une prédisposition permet de fixer la valeur d'une variable au démarrage de l'application.
Les variables T ou %T, M ou %MW, L ou %MD et F ou %F peuvent être prédisposées.

La syntaxe est la suivante :
« $(variable)=constante{,constante{,constante...}} »

Pour les temporisations la valeur doit être écrite en décimal et comprise entre 0 et 65535.
Pour les mots la syntaxe suivante doit être utilisée :

- ⇨ Décimal : éventuellement le caractère « - » puis 1 à 5 digits « 0123456789 »,
- ⇨ Hexadécimal : le préfixe « $ » ou « 16# » suivi de 1 à 4 digits « 0123456789ABCDEF »,
- ⇨ Binaire : le préfixe « % » ou « 2# » suivi de 1 à 16 digits « 01 »,
- ⇨ ASCII : (deux caractères par mot) le caractère «"» suivi de n caractères suivis de «"»,
- ⇨ ASCII : (un caractère par mot) le caractère «'» suivi de n caractères suivis de « ' ».

Pour les longs la syntaxe suivante doit être utilisée :

- ⇨ Décimal : éventuellement le caractère « - » puis 1 à 10 digits « 0123456789 »,
- ⇨ Hexadécimal : le préfixe « $ » ou « 16# » suivi de 1 à 8 digits « 0123456789ABCDEF »,
- ⇨ Binaire : le caractère « % » ou « 2# » suivi de 1 à 32 digits « 01 »,
- ⇨ ASCII : (quatre caractères par long) le caractère « " » suivi de n caractères suivis de « " »,
- ⇨ ASCII : (un caractère par long) le caractère « ' » suivi de n caractères suivis de « ' »

Pour les flottants, la valeur doit être écrite sous la forme :
[-] i [[.d] Esx]
i est la partie entière
d une éventuelle partie décimale
s éventuellement le signe de l'exposant
x éventuellement l'exposant

Exemples :

```
$t25=100
```

fixe la consigne de la temporisation 25 à 10 s

```
$MW200=100,200,300,400
```

place les valeurs 100,200,300,400 dans les mots 200, 201, 202, 203

```
$m200="ABCDEF"
```

place la chaîne de caractères « ABCDEF » à partir de m200 (« AB » dans m200, « CD » dans m201, « EF » dans m202)

```
$m200='ABCDEF'
```

place la chaîne de caractères « ABCDEF » à partir de m200, chaque mot reçoit un caractère.

```
$f1000=3.14
```

place la valeur 3,14 dans f1000

```
$%mf100=5.1E-15
```

place la valeur 5,1 * 10 exposant -15 dans %mf100

```
$l200=16#12345678
```

place la valeur 12345678 (hexa) dans le long l200

Il est possible d'écrire plus facilement du texte dans les prédispositions.

Exemple :

```
$m200=" Arrêt de la vanne N°10 "
```

Place le message à partir du mot 200 en plaçant deux caractères dans chaque mot.

```
$m400=' Défaut moteur '
```

Place le message à partir du mot 400 en plaçant un caractère dans l'octet de poids faibles de chaque mot, l'octet de poids fort contient 0.

La syntaxe « $...= » permet de poursuivre une table de prédispositions à la suite de la précédente.

Par exemple :

```
#$m200=1,2,3,4,5
#$...=6,7,8,9
```

Place les valeurs de 1 à 9 dans les mots m200 à m208.

Les prédispositions peuvent être écrites de la même façon que le langage littéral bas niveau ou dans une directive de compilation dans un folio. Dans ce cas, la prédisposition commence par le caractère « # ».

Exemple de prédisposition écrite dans une boîte de code :

Exemple de prédisposition écrite dans une directive de compilation :

```
#$m200=12,13
```

Adressage indirect

L'adressage indirect permet d'effectuer une opération sur une variable pointée par un index.
Ce sont les variables M (les mots) qui servent d'index.
Syntaxe :
« variable (index) »

```
lda 10        ; charge 10 dans l'accumulateur
sta m200      ; le place dans le mot 200
set o(200)    ; mise à un de la sortie pointée par le mot 200 (o10)
```

Adresse d'une variable

Le caractère « ? » permet de spécifier l'adresse d'une variable.
Exemple :

```
lda ?o10            ; place la valeur 10 dans l'accumulateur
```

Cette syntaxe est surtout intéressante si des symboles sont utilisés.

Exemple :

```
lda ?_vanne_        ; place dans l'accumulateur le numéro de la variable
                    ; associée au symbole « _vanne_ »
```

Cette syntaxe peut également être utilisée dans les prédispositions pour créer des tables d'adresses de variables.

Exemple :

```
$m200=?_vanne1_,?_vanne2_,?_vanne3_
```

Sauts et labels

Les sauts doivent faire référence à un label. La syntaxe d'un label est :
« :nom du label: »

Exemple :

```
jmp :suite:
...
:suite:
```

Liste des fonctions par type

Fonctions booléennes

SET	mise à un
RES	mise à zéro
INV	inversion
EQU	équivalence
NEQ	non-équivalence

Fonctions de chargement et de stockage sur entiers et flottants

LDA	chargement
STA	stockage

Fonctions arithmétiques sur entiers et flottants

ADA	addition
SBA	soustraction
MLA	multiplication
DVA	division
CPA	comparaison

Fonctions arithmétiques sur flottants

ABS	valeur absolue
SQR	racine carrée

Fonctions d'accès aux ports d'entrées/sorties sur PC

AIN	lecture d'un port
AOU	écriture d'un port

Fonctions d'accès à la mémoire sur PC

ATM	lecture d'une adresse mémoire
MTA	écriture d'une adresse mémoire

Fonctions binaires sur entiers

ANA	et bit à bit
ORA	ou bit à bit
XRA	ou exclusif bit à bit
TSA	test bit à bit
SET	mise à un de tous les bits
RES	mise à zéro de tous les bits
RRA	décalage à droite
RLA	décalage à gauche

Autres fonctions sur entiers

INC	incrémentation
DEC	décrémentation

Fonctions de conversion

ATB	entier vers booléens
BTA	booléens vers entier
FTI	flottant vers entier
ITF	entier vers flottant
LTI	entier 32 bits vers entier 16 bits
ITL	entier 16 bits vers entier 32 bits

Fonctions trigonométriques

SIN	sinus
COS	cosinus
TAN	tangente
ASI	arc sinus
ACO	arc cosinus
ATA	arc tangente

Fonctions de branchement

JMP	saut
JSR	saut à un sous-programme
RET	retour de sous-programme

Fonctions de test

RFZ	flag de résultat nul
RFS	flag de signe
RFO	flag de débordement
RFC	flag de retenue

Fonctions d'accès asynchrones aux entrées sorties

RIN lecture des entrées
WOU écriture des sorties

Informations contenues dans la liste des fonctions

Pour chaque instruction sont donnés :

⇨ Nom : le mnémonique.

⇨ Fonction : une description de la fonction réalisée par l'instruction.

⇨ Variables : les types de variables utilisables avec l'instruction.

⇨ Adressage : les types d'adressages utilisables.

⇨ Voir aussi : les autres instructions ayant un rapport avec ce mnémonique.

⇨ Exemple : un exemple d'utilisation.

Les post-processeurs qui génèrent des langages constructeurs sont soumis à certaines restrictions. Consultez les notices relatives à ces post-processeurs pour avoir le détail de ces restrictions.

ABS

Nom	:	ABS - abs accumulator
Fonction	:	calcule la valeur absolue de l'accumulateur flottant
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	SQR
Exemple	:	

```
lda f200
abs aaf
sta f201
; laisse dans f201 la valeur absolue de f200
```


ACO

Nom	:	ACO – arc cosinus accumulator
Fonction	:	calcule la valeur de l'arc cosinus de l'accumulateur flottant
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	COS, SIN, TAN, ASI, ATA
Exemple	:	

```
lda f200
aco aaf
sta f201
; laisse dans f201 la valeur de l'arc cosinus de f200
```


ADA

Nom	:	ADA - adds accumulator
Fonction	:	ajoute une valeur à l'accumulateur
Variables	:	M ou %MW, L ou %MD, F ou %MF
Adressage	:	absolu, indirect, immédiat
Voir aussi	:	SBA
Exemple	:	

```
ada 200
; ajoute 200 à l'accumulateur 16 bits

ada f124
; ajoute le contenu de f124 à l'accumulateur flottant

ada l200
; ajoute le contenu de l200 à l'accumulateur 32 bits

ada 200L
; ajoute 200 à l'accumulateur 32 bits

ada 3.14R
; ajoute 3.14 à l'accumulateur flottant
```


AIN

Nom	:	AIN - accumulator input
Fonction	:	lecture d'un port d'entrée (8 bits) et stockage dans la partie basse de l'accumulateur 16 bits ; lecture d'un port d'entrée 16 bits et stockage dans l'accumulateur 16 bits (dans ce cas l'adresse du port doit être écrite sous la forme d'une constante 32 bits) utilisable seulement avec l'exécuteur PC
Variables	:	M ou %MW
Adressage	:	indirect, immédiat
Voir aussi	:	AOU
Exemple	:	

```
ain $3f8
; lecture du port $3f8 (8 bits)

ain $3f8l
; lecture du port $3f8 (16 bits)
```


ANA

Nom	:	ANA - and accumulator
Fonction	:	effectue un ET logique entre l'accumulateur 16 bits et un mot ou une constante ou l'accumulateur 32 bits et un long ou une constante
Variables	:	M ou %MW, L ou %MD
Adressage	:	absolu, indirect, immédiat
Voir aussi	:	ORA, XRA
Exemple	:	

```
ana %1111111100000000
; masque les 8 bits de poids faible de
; l'accumulateur 16 bits

ana $ffff0000L
; masque les 16 bits de poids faibles de l'accumulateur 32 bits
```


AOU

Nom	:	AOU - accumulator output
Fonction	:	transfère la partie basse (8 bits) du contenu de l'accumulateur 16 bits sur un port de sortie ; transfère les 16 bits de l'accumulateur 16 bits sur un port de sortie (sans ce cas l'adresse du port doit être écrite sous la forme d'une constante 32 bits) utilisable seulement avec l'exécuteur PC
Variables	:	M ou %MW
Adressage	:	indirect, immédiat
Voir aussi	:	AIN
Exemple	:	

```
lda "A"
aou $3f8
; place le caractère « A » sur le port de sortie $3f8

lda $3f8
sta m200
lda "z"
aou m(200)
; place le caractère « z » sur le port de sortie $3f8

lda $1234
aou $300l
; place la valeur 16 bits 1234 sur le port de sortie $300
```


ASI

Nom	:	ASI – arc sinus accumulator
Fonction	:	calcule la valeur de l'arc sinus de l'accumulateur flottant
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	COS, SIN, TAN, ACO, ATA
Exemple	:	

```
lda f200
asi aaf
sta f201
; laisse dans f201 la valeur de l'arc sinus de f200
```


ATA

Nom	:	ATA – arc tangente accumulator
Fonction	:	calcule la valeur de l'arc tangente de l'accumulateur flottant
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	COS, SIN, TAN, ACO, ASI
Exemple	:	

```
lda f200
ata aaf
sta f201
; laisse dans f201 la valeur de l'arc tangente de f200
```


ATB

Nom	:	ATB - accumulator to bit
Fonction	:	transfère les 16 bits de l'accumulateur 16 bits vers 16 variables booléennes successives ; le bit de poids faible correspond à la première variable booléenne
Variables	:	I ou %I, O ou %Q, B ou %M, T ou %T, U*
Adressage	:	absolu
Voir aussi	:	BTA
Exemple	:	

```
lda m200
atb o0
; recopie les 16 bits de m200 dans les variables
; o0 à o15
```

* Attention : pour pouvoir utiliser les bits U avec cette fonction il faut réaliser une table linéaire de bits avec la directive de compilation #B.

ATM

Nom	:	ATM - accumulator to memory
Fonction	:	transfère l'accumulateur 16 bits à une adresse mémoire ; le mot ou la constante spécifié défini l'offset de l'adresse mémoire à atteindre, le mot m0 doit être chargé avec la valeur du segment de l'adresse mémoire à atteindre ; utilisable seulement avec l'exécuteur PC
Variables	:	M ou %MW
Adressage	:	indirect, immédiat
Voir aussi	:	MTA
Exemple	:	

```
lda $b800
sta m0
lda 64258
atm $10
; place la valeur 64258 à l'adresse $b800:$0010
```


BTA

Nom	:	BTA - bit to accumulator
Fonction	:	transfère 16 variables booléennes successives vers les 16 bits de l'accumulateur 16 bits ; le bit de poids faible correspond à la première variable booléenne
Variables	:	I ou %I, O ou %Q, B ou %M, T ou %T, U*
Adressage	:	absolu
Voir aussi	:	ATB
Exemple	:	

```
bta i0
sta m200
; recopie les 16 entrées i0 à i15 dans le mot m200
```

* Attention : pour pouvoir utiliser les bits U avec cette fonction il faut réaliser une table linéaire de bits avec la directive de compilation #B.

COS

Nom	:	COS – cosinus accumulator
Fonction	:	calcule la valeur du cosinus de l'accumulateur flottant
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	SIN, TAN, ACO, ASI, ATA
Exemple	:	

```
lda f200
cos aaf
sta f201
; laisse dans f201 la valeur du cosinus de f200
```


CPA

Nom	:	CPA - compares accumulator
Fonction	:	compare une valeur à l'accumulateur 16 bits ou 32 bits ou flottant ; effectue la même opération que SBA mais sans modifier le contenu de l'accumulateur
Variables	:	M ou %MW, L ou %MD, F ou %MF
Adressage	:	absolu, indirect, immédiat
Voir aussi	:	SBA
Exemple	:	

```
lda m200
cpa 4
rfz o0
; met o0 à 1 si m200 est égal à 4, autrement o0
; est mis à 0

lda f200
cpa f201
rfz o1
; met o1 à 1 si f200 est égal à f201, autrement o1
; est mis à 0
```


DEC

Nom	:	DEC – decrement
Fonction	:	décrémente un mot, un compteur, un long, l'accumulateur 16 bits ou 32 bits
Variables	:	M ou %MW, C ou %C, L ou %MD
Adressage	:	absolu, indirect, accumulateur
Voir aussi	:	INC
Exemple	:	

```
dec m200
; décrémente m200

dec aal
; décrémente l'accumulateur 32 bits

dec m200
dec m201 and m200=-1
; décrémente une valeur 32 bits composée de
; m200 (poids faibles)
; et m201 (poids forts)
```


DVA

Nom	:	DVA - divides accumulator
Fonction	:	division de l'accumulateur 16 bits par un mot ou une constante ; division de l'accumulateur flottant par un flottant ou une constante ; division de l'accumulateur 32 bits par un long ou une constante, pour l'accumulateur 16 bits le reste est placé dans le mot m0 ; en cas de division par 0 le bit système 56 passe à 1
Variables	:	M ou %MW, L ou %MD, F ou %MF
Adressage	:	absolu, indirect, immédiat
Voir aussi	:	MLA
Exemple	:	

```
lda m200
dva 10
sta m201
; m201 est égal à m200 divisé par 10, m0 contient le
; reste de la division

lda l200
dva $10000L
sta l201
```


EQU

Nom	:	EQU - equal
Fonction	:	force une variable à 1 si le test est vrai, dans le cas contraire la variable est forcée à 0
Variables	:	I ou %I, O ou %Q, B ou %M, T ou %T, X ou %X, U
Adressage	:	absolu, indirect (sauf sur les variables X)
Voir aussi	:	NEQ, SET, RES, INV
Exemple	:	

```
equ o0 and i10
; force la sortie o0 au même état que l'entrée i10

lda 10
sta m200
equ o(200) and i0
; force o10 au même état que l'entrée i0

$t0=100
equ t0 and i0
equ o0 and t0
; force o0 à l'état de i0 avec un retard à l'activation
; de 10 secondes
```


FTI

Nom	:	FTI - float to integer
Fonction	:	transfère l'accumulateur flottant vers l'accumulateur 16 bits
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	ITF
Exemple	:	

```
lda f200
fti aaa
sta m1000
; laisse la partie entière de f200 dans m1000
```


INC

Nom	:	INC - increment
Fonction	:	incrémente un mot, un compteur, un long, l'accumulateur 16 bits ou 32 bits
Variables	:	M ou %MW, C ou %C, L ou %MD
Adressage	:	absolu, indirect, accumulateur
Voir aussi	:	DEC
Exemple	:	

```
inc m200
; ajoute 1 à m200

inc m200
inc m201 and m201=0
; incrémente une valeur sur 32 bits, m200
; représente les
; poids faibles, et m201 les poids forts

inc l200
; incrémente le long l200
```


INV

Nom	:	INV - inverse
Fonction	:	inverse l'état d'une variable booléenne, ou inverse tous les bits d'un mot, d'un long ou de l'accumulateur 16 bits ou 32 bits
Variables	:	I ou %I, O ou %Q, B ou %M, T ou %T, X ou %X, U, M ou %MW, L ou %MD
Adressage	:	absolu, indirect, accumulateur
Voir aussi	:	EQU, NEQ, SET, RES
Exemple	:	

```
inv o0
; inverse l'état de la sortie 0

inv aaa
; inverse tous les bits de l'accumulateur 16 bits

inv m200 and i0
; inverse tous les bits de m200 si i0 est à l'état 1
```


ITF

Nom	:	ITF - integer to float
Fonction	:	transfère l'accumulateur 16 bits vers l'accumulateur flottant
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	FTI
Exemple	:	

```
lda 1000
itf aaa
sta f200
; laisse la constante 1000 dans f200
```


ITL

Nom	:	ITL - integer to long
Fonction	:	transfère l'accumulateur 16 bits vers l'accumulateur 32 bits
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	LTI
Exemple	:	

```
lda 1000
itl aaa
sta f200
; laisse la constante 1000 dans l200
```


JMP

Nom	:	JMP - jump
Fonction	:	saut à un label
Variables	:	label
Adressage	:	label
Voir aussi	:	JSR
Exemple	:	

```
jmp :fin de programme:
; branchement inconditionnel au label :fin
; de programme:


jmp :suite: and i0
set o0
set o1
:suite:
; branchement conditionnel au label :suite:
; suivant l'état de i0
```


JSR

Nom	:	JSR - jump sub routine
Fonction	:	effectue un branchement à un sous-programme
Variables	:	label
Adressage	:	label
Voir aussi	:	RET
Exemple	:	

```
        lda m200
        jsr :carre:
        sta m201
        jmp :fin:

:carre:
        sta m53
        mla m53
        sta m53
        ret m53

:fin:

; le sous-programme « carre » élève le contenu de
; l'accumulateur au carré
```


LDA

Nom	:	LDA - load accumulator
Fonction	:	charge dans l'accumulateur 16 bits une constante, un mot ou un compteur ; charge dans l'accumulateur 32 bits un long ou une constante, charge dans l'accumulateur flottant un flottant ou une constante, charge un compteur de temporisation dans l'accumulateur 16 bits
Variables	:	M ou %MW, C ou %C, L ou %MD, F ou %MF, T ou %T
Adressage	:	absolu, indirect, immédiat
Voir aussi	:	STA
Exemple	:	

```
lda 200
; charge la constante 200 dans l'accumulateur 16 bits

lda 0.01R
; charge la constante réelle 0.01 dans l'accumulateur flottant

lda t10
; charge le compteur de la temporisation 10 dans
; l'accumulateur
```


LTI

Nom	:	LTI - long to integer
Fonction	:	transfère l'accumulateur 32 bits vers l'accumulateur 16 bits
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	ITL
Exemple	:	

```
lda l200
lti aaa
sta m1000
; laisse les 16 bits de poids faibles de l200 dans m1000
```


MLA

Nom	:	MLA - multiples accumulator
Fonction	:	multiplication de l'accumulateur 16 bits par un mot ou une constante ; multiplication de l'accumulateur 32 bits par un long ou une constante, multiplication de l'accumulateur flottant par un flottant ou une constante ; pour l'accumulateur 16 bits, les 16 bits de poids forts du résultat de la multiplication sont transférés dans m0
Variables	:	M ou %MW, L ou %MD, F ou %MF
Adressage	:	absolu, indirect, immédiat
Voir aussi	:	DVA
Exemple	:	

```
lda m200
mla 10
sta m201
; multiplie m200 par 10, m201 est chargé avec les
; 16 bits de poids faibles, et m0 avec les 16 bits de
; poids forts
```


MTA

Nom	:	MTA - memory to accumulator
Fonction	:	transfère le contenu d'une adresse mémoire dans l'accumulateur 16 bits ; le mot ou la constante spécifié défini l'offset de l'adresse mémoire à atteindre, le mot m0 doit être chargé avec la valeur du segment de l'adresse mémoire à atteindre ; utilisable seulement avec l'exécuteur PC
Variables	:	M ou %MW
Adressage	:	indirect, immédiat
Voir aussi	:	ATM
Exemple	:	

```
lda $b800
sta m0
mta $10
; place la valeur contenue à l'adresse $b800:$0010
; dans l'accumulateur 16 bits
```


NEQ

Nom	:	NEQ - not equal
Fonction	:	force une variable à 0 si le test est vrai, dans le cas contraire la variable est forcée à 1
Variables	:	I ou %I, O ou %Q, B ou %M, T ou %T, X ou %X, U
Adressage	:	absolu, indirect (sauf sur les variables X)
Voir aussi	:	EQU, SET, RES, INV
Exemple	:	

```
neq o0 and i00
; force la sortie o0 à l'état complémenté de l'entrée
; i10

lda 10
sta m200
neq o(200) and i0
; force o10 à l'état complémenté de l'entrée i0

$t0=100
neq t0 and i0
neq o0 and t0
; force o0 à l'état de i0 avec un retard à la
; désactivation de 10 secondes
```


ORA

Nom	:	ORA - or accumulator
Fonction	:	effectue un OU logique entre l'accumulateur 16 bits et un mot ou une constante, ou entre l'accumulateur 32 bits et un long ou une constante
Variables	:	M ou %M, L ou %MD
Adressage	:	absolu, indirect, immédiat
Voir aussi	:	ANA, XRA
Exemple	:	

```
ora %1111111100000000
; force les 8 bits de poids forts de
; l'accumulateur 16 bits à 1

ora $ffff0000L
; force les 16 bits de poids forts de l'accumulateur 32 bits
; à 1
```


RES

Nom	:	RES - reset
Fonction	:	force une variable booléenne, un mot, un compteur, un long, l'accumulateur 16 bits ou l'accumulateur 32 bits à 0
Variables	:	I ou %I, O ou %Q, B ou %M, T ou %T, X ou %X, U, M ou %MW, C ou %C, L ou %MD
Adressage	:	absolu, indirect (sauf sur les variables X), accumulateur
Voir aussi	:	NEQ, SET, EQU, INV
Exemple	:	

```
res o0
; force la sortie o0 à 0

lda 10
sta m200
res o(200) and i0
; force o10 à 0 si l'entrée i0 est à 1

res c0
; force le compteur 0 à 0
```


RET

Nom	:	RET - return
Fonction	:	marque le retour d'un sous-programme et place dans l'accumulateur 16 bits un mot ou une constante ; ou place dans l'accumulateur 32 bits un long ou une constante, ou place dans l'accumulateur flottant un flottant ou une constante
Variables	:	M ou %MW, L ou %MD, F ou %MF
Adressage	:	absolu, indirect, immédiat
Voir aussi	:	JSR
Exemple	:	

```
ret 0
; retour de sous-programme en plaçant 0 dans
; l'accumulateur 16 bits

ret f200
; retour de sous-programme en plaçant le contenu
; de f200 dans l'accumulateur flottant
```


RFC

Nom	:	RFC - read flag : carry
Fonction	:	transfère le contenu de l'indicateur de retenue dans une variable booléenne
Variables	:	I ou %I, O ou %Q, B ou %M, T ou %T, X ou %X, U
Adressage	:	absolu
Voir aussi	:	RFZ, RFS, RFO
Exemple	:	

```
rfc o0
; transfère l'indicateur de retenue dans o0

lda m200
ada m300
sta m400
rfc b99
lda m201
ada m301
sta m401
inc m401 and b99
; effectue une addition sur 32 bits
; (m400,401)=(m200,201)+(m300,301)
; m200, m300 et m400 sont les poids faibles
; m201, m301 et m401 sont les poids forts
```


RFO

Nom	:	RFO - read flag : overflow
Fonction	:	transfère le contenu de l'indicateur de débordement dans une variable booléenne
Variables	:	I ou %I, O ou %Q, B ou %M, T ou %T, X ou %X, U
Adressage	:	absolu
Voir aussi	:	RFZ, RFS, RFC
Exemple	:	

```
rfo o0
; transfère l'indicateur de débordement dans o0
```


RFS

Nom	:	RFS - read flag : signe
Fonction	:	transfère le contenu de l'indicateur de signe dans une variable booléenne
Variables	:	I ou %I, O ou %Q, B ou %M, T ou %T, X ou %X, U
Adressage	:	absolu
Voir aussi	:	RFZ, RFC, RFO
Exemple	:	

```
rfs o0
; transfère l'indicateur de signe dans o0
```


RFZ

Nom	:	RFZ - read flag : zero
Fonction	:	transfère le contenu de l'indicateur de résultat nul dans une variable booléenne
Variables	:	I ou %I, O ou %Q, B ou %M, T ou %T, X ou %X, U
Adressage	:	absolu
Voir aussi	:	RFC, RFS, RFO
Exemple	:	

```
rfz o0
; transfère l'indicateur de résultat nul dans o0

lda m200
cpa m201
rfz o0
; positionne o0 à 1 si m200 est égal à m201
; ou à 0 dans le cas contraire
```


RIN

Nom	:	RIN - read input
Fonction	:	effectue une lecture des entrées physiques. Cette fonction est uniquement implémentée sur cibles Z et varie suivant la cible. Consultez la documentation relative à chaque éxécuteur pour plus de détails.
Variables	:	aucune
Adressage	:	immédiat
Voir aussi	:	WOU

RLA

Nom	:	RLA - rotate left accumulator
Fonction	:	effectue une rotation à gauche des bits de l'accumulateur 16 bits ou 32 bits ; le bit évacué à gauche entre à droite ; l'argument de cette fonction est une constante qui précise le nombre de décalages à effectuer, la taille de l'argument (16 ou 32 bits) détermine quel est l'accumulateur qui doit subir la rotation
Variables	:	aucune
Adressage	:	immédiat
Voir aussi	:	RRA
Exemple	:	

```
ana $f000
; isole le digit de poids fort de l'accumulateur 16 bits
rla 4
; et le ramène à droite

rla 8L
; effectue 8 rotations à gauche des bits de l'accumulateur
; 32 bits
```


RRA

Nom	:	RRA - rotate right accumulator
Fonction	:	effectue une rotation à droite des bits de l'accumulateur 16 bits ou 32 bits ; le bit évacué à droite entre à gauche ; l'argument de cette fonction est une constante qui précise le nombre de décalages à effectuer, la taille de l'argument (16 ou 32 bits) détermine si c'est l'accumulateur 16 ou 32 bits qui doit subir la rotation
Variables	:	aucune
Adressage	:	immédiat
Voir aussi	:	RLA
Exemple	:	

```
ana $f000
; isole le digit de poids fort de l'accumulateur 16 bits
rra 12
; et le ramène à droite

rra 1L
; effectue une rotation des bits de l'accumulateur 32 bits
; d'une position vers la droite
```


SBA

Nom	:	SBA - substacts accumulator
Fonction	:	enlève le contenu d'un mot ou une constante à l'accumulateur 16 bits ; enlève le contenu d'un long ou d'une constante à l'accumulateur 32 bits, enlève le contenu d'un flottant ou d'une constante à l'accumulateur flottant
Variables	:	M ou %MW, L ou %MD, F ou %MF
Adressage	:	absolu, indirect, immédiat
Voir aussi	:	ADA
Exemple	:	

```
sba 200
; enlève 200 à l'accumulateur 16 bits

sba f(421)
; enlève le contenu du flottant dont le numéro est
; contenu dans le mot 421 à l'accumulateur flottant
```


SET

Nom	:	SET - set
Fonction	:	force une variable booléenne à 1; force tous les bits d'un mot, d'un compteur, d'un long, de l'accumulateur 16 bits ou de l'accumulateur 32 bits à 1
Variables	:	I ou %I, O ou %Q, B ou %M, T ou %T, X ou %X, U, M ou %MW, C ou %C, L ou %MD
Adressage	:	absolu, indirect (sauf sur les variables X), accumulateur
Voir aussi	:	NEQ, RES, EQU, INV
Exemple	:	

```
set o0
; force la sortie o0 à 1

lda 10
sta m200
set o(200) and i0
; force o10 à 1 si l'entrée i0 est à 1

set m200
; force m200 à la valeur -1

set aal
; force tous les bits de l'accumulateur 32 bits à 1
```


SIN

Nom	:	SIN – sinus accumulator
Fonction	:	calcule la valeur du sinus de l'accumulateur flottant
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	COS, TAN, ACO, ASI, ATA
Exemple	:	

```
lda f200
sin aaf
sta f201
; laisse dans f201 la valeur du sinus de f200
```


SQR

Nom	:	SQR - square root
Fonction	:	calcule la racine carrée de l'accumulateur flottant
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	ABS
Exemple	:	

```
lda 9
itf aaa
sqr aaf
fti aaa
sta m200
; laisse la valeur 3 dans m200
```


STA

Nom	:	STA - store accumulator
Fonction	:	stocke l'accumulateur 16 bits dans un compteur ou un mot; stocke l'accumulateur 32 bits dans un long, stocke l'accumulateur flottant dans un flottant, stocke l'accumulateur 16 bits dans une consigne de temporisation
Variables	:	M ou %MW, C ou %C, L ou %MD, F ou %MF, T ou %T
Adressage	:	absolu, indirect
Voir aussi	:	LDA
Exemple	:	

```
sta m200
; transfère le contenu de l'accumulateur 16 bits
; dans le mot 200

sta f200
; transfère le contenu de l'accumulateur flottant
; dans le flottant 200

sta l200
; transfère l'accumulateur 32 bits dans le long l200
```


TAN

Nom	:	TAN – tangent accumulator
Fonction	:	calcule la valeur de la tengente de l'accumulateur flottant
Variables	:	aucune
Adressage	:	accumulateur
Voir aussi	:	COS, SIN, ACO, ASI, ATA
Exemple	:	

```
lda f200
tan aaf
sta f201
; laisse dans f201 la valeur de la tengente de f200
```


TSA

Nom	:	TSA - test accumulator
Fonction	:	effectue un ET logique entre l'accumulateur 16 bits et un mot ou une constante ; effectue un ET logique entre l'accumulateur 32 bits et un long ou une constante, opère de façon similaire à l'instruction ANA, mais sans modifier le contenu de l'accumulateur
Variables	:	M ou %MW, L ou %MD
Adressage	:	absolu, indirect, immédiat
Voir aussi	:	ANA
Exemple	:	

```
tsa %10
rfz b99
jmp :suite: and b99
; branchement au label :suite: si le bit 1
; de l'accumulateur 16 bits est à 0
```


WOU

Nom	:	WOU - write output
Fonction	:	effectue une écriture des sorties physiques. Cette fonction est uniquement implémentée sur cibles Z (et varie suivant la cible). Consultez la documentation relative à chaque exécuteur pour plus de détails.
Variables	:	aucune
Adressage	:	immédiat
Voir aussi	:	RIN

XRA

Nom	:	XRA - xor accumulator
Fonction	:	effectue un OU EXCLUSIF entre l'accumulateur 16 bits et un mot ou une constante, effectue un OU EXCLUSIF entre l'accumulateur 32 bits et un long ou une constante
Variables	:	M ou %MW, L ou %MD
Adressage	:	absolu, indirect, immédiat
Voir aussi	:	ORA, ANA,
Exemple	:	

```
xra %1111111100000000
; inverse les 8 bits de poids fort de l'accumulateur 16 bits

xra 1L
; inverse le bit de poids faible de l'accumulateur 32 bits
```


Macro-instruction

Les macro-instructions sont des nouvelles instructions du langage littéral derrière lesquelles se cache un ensemble d'instructions de base.
Syntaxe d'appel d'une macro-instruction :
« %<nom de la macro-instruction*> {paramètres ...} »
Syntaxe de déclaration d'une macro-instruction :

```
#MACRO
<programme>
#ENDM
```

Cette déclaration se trouve dans un fichier portant le nom de la macro-instruction et possédant l'extension « .M ».

Le fichier .M peut être placé dans sous-répertoire « lib » du répertoire d'installation d'AUTOMGEN ou dans les ressources du projet.

Dix paramètres peuvent être passés à la macro-instruction. A l'appel, ces paramètres seront placés sur la même ligne que la macro-instruction et seront séparés par un espace.
Dans le programme de la macro-instruction la syntaxe « {?n} » fait référence au paramètre n.

Exemple :
Réalisons la macro-instruction « carre » qui élève le premier paramètre de la macro-instruction au carré et range le résultat dans le deuxième paramètre.
Appel de la macro-instruction :

```
lda 3
sta m200
%carre m200 m201
; m201 contiendra 9 ici
```

Fichier « CARRE.M » :

```
#MACRO
lda {?0}
mla {?0}
sta {?1}
#ENDM
```

* Le nom de la macro-instruction peut être un chemin d'accès complet vers le fichier « .M », il peut contenir une désignation de lecteur et de répertoire.

Librairie

La notion de librairie permet la définition de ressources qui seront compilées une seule fois dans une application, quel que soit le nombre d'appels à ces ressources.

Syntaxe de définition d'une librairie :

```
#LIBRARY <nom de la librairie>
<programme>
#ENDL
```

<nom de la librairie> est le nom de la fonction qui sera appelée par une instruction
jsr :<nom de la librairie>:
Au premier appel rencontré par le compilateur, le code de la librairie est compilé. Pour les suivants, l'appel est simplement dirigé vers la routine existante.
Ce mécanisme est particulièrement adapté à l'utilisation des blocs fonctionnels et des macro-instructions pour limiter la génération de code dans le cas de l'utilisation multiple de mêmes ressources programmes.
Les mots m120 à m129 sont réservés aux librairies et peuvent être utilisés pour le passage des paramètres.

Macro-instructions prédéfinies

Des macro-instructions de conversion se trouvent dans le sous répertoire « LIB » du répertoire où a été installé AUTOMGEN.
Les équivalents en blocs fonctionnels sont également présents.

Description des macro-instructions prédéfinies

Conversions

```
%ASCTOBIN <deux premiers digits> <deux derniers digits> <résultat en binaire>
```

Effectue une conversion ASCII hexadécimal (deux premiers paramètres) vers binaire (troisième paramètre), en sortie l'accumulateur contient $FFFF si les deux premiers paramètres ne sont pas des nombres ASCII valides, 0 autrement. Tous les paramètres sont des mots de 16 bits.

```
%BCDTOBIN <valeur en BCD> <valeur en binaire>
```

Effectue une conversion BCD vers binaire. En sortie l'accumulateur contient $FFFF si le premier paramètre n'est pas un nombre bcd valide, 0 autrement. Les deux paramètres sont des mots de 16 bits.

```
%BINTOASC <valeur en binaire> <résultat partie haute> <résultat partie basse>
```


Effectue une conversion binaire (premier paramètre) vers ASCII hexadécimal (deuxième et troisième paramètres). Tous les paramètres sont des mots de 16 bits.

```
%BINTOBCD <valeur en binaire> <valeur en BCD>
```

Effectue une conversion BCD (premier paramètre) vers binaire (deuxième paramètre). En sortie l'accumulateur contient $FFFF si le nombre binaire ne peut être converti en BCD, 0 autrement.

```
%GRAYTOB <valeur en code GRAY> <valeur en binaire>
```

Effectue une conversion code Gray (premier paramètre) vers binaire (deuxième paramètre).

Traitement sur table de mots

```
%COPY <premier mot table source> <premier mot table destination> <nombre de mots>
```

Copie une table de mots source vers une table de mots destination. La longueur est donnée en nombre de mots.

```
%COMP <premier mot table 1> <premier mot table 2> <nombre de mots> <résultat>
```

Compare deux tables de mots. Le résultat est une variable binaire qui prend la valeur 1 si tous les éléments de la table 1 sont identiques à la table 2.

```
%FILL <premier mot table> <valeur> <nombre de mots>
```

Remplit une table de mots avec une valeur.

Traitement sur chaîne de caractères

Le codage des chaînes de caractères est le suivant : un caractère par mot, un mot contenant la valeur 0 marque la fin de la chaîne. Dans les macro-instructions, les chaînes sont passées en paramètres en désignant le premier mot qui les compose.

```
%STRCPY <chaîne source> <chaîne destination>
```

Copie une chaîne vers une autre.

```
%STRCAT <chaîne source> <chaîne destination>
```

Ajoute la chaîne source à la fin de la chaîne destination.

```
%STRCMP <chaîne 1> <chaîne 2> <résultat>
```

Compare deux chaînes. Le résultat est une variable booléenne qui passe à 1 si les deux chaînes sont identiques.

```
%STRLEN <chaîne> <résultat>
```

Place la longueur de la chaîne dans le mot résultat.

```
%STRUPR <chaîne>
```


Transforme tous les caractères de la chaîne en majuscules.

```
%STRLWR <chaîne>
```

Transforme tous les caractères de la chaîne en minuscules.

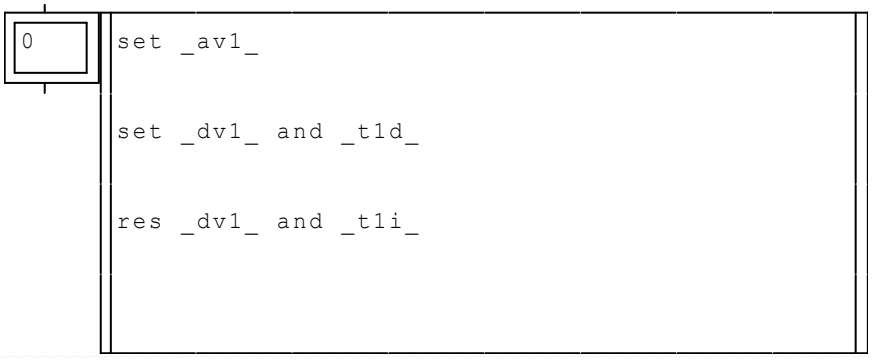
Exemple:
Conversion de m200 (binaire) vers m202, m203 en 4 digits (ASCII bcd)

```
%bintobcd m200 m201
%bintoasc m201 m202 m203
```

Exemple en langage littéral bas niveau

Cahier des charges : commençons par l'exemple le plus simple : aller et retour d'une locomotive sur la voie 1.

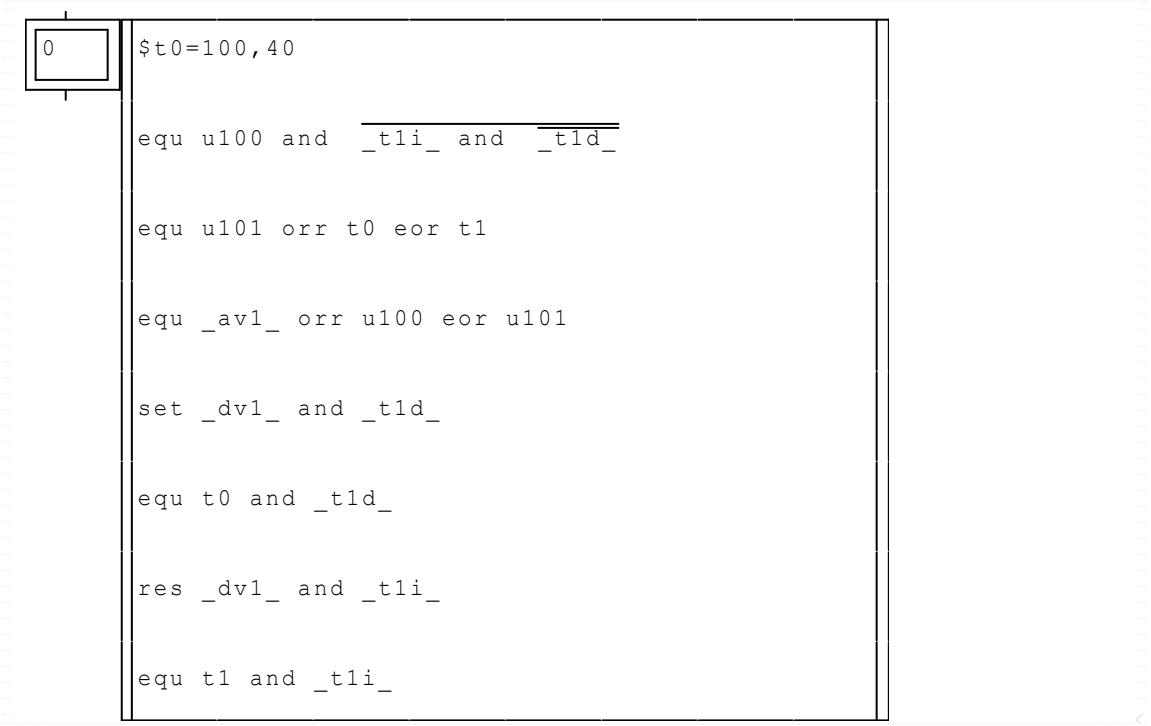
Solution :



exemples\lit\littéral bas niveau1.agn

Un exemple un peu plus évolué.
Cahier des charges :
La locomotive devra maintenant marquer une attente de 10 secondes à l'extrémité droite de la voie et une attente de 4 secondes à l'extrémité gauche.

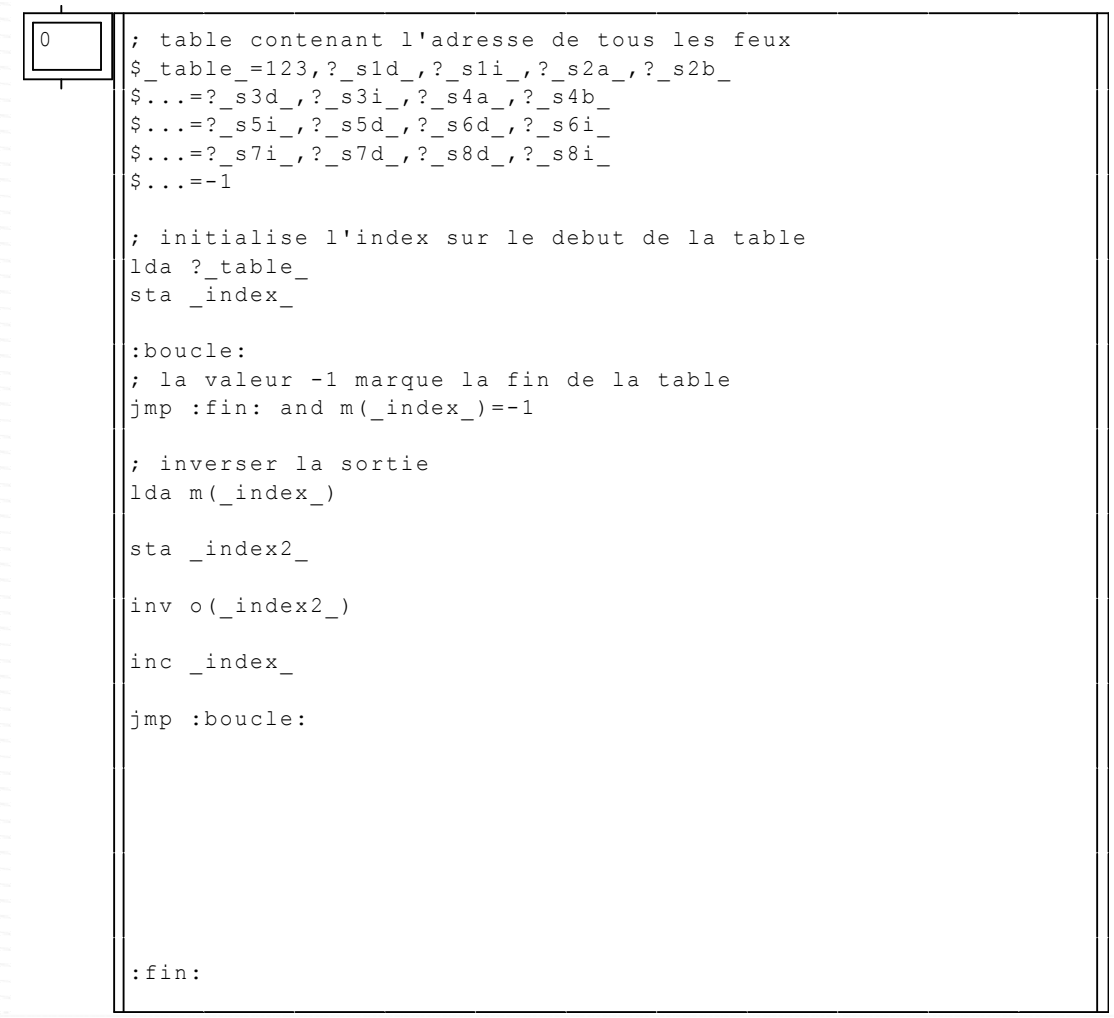
Solution :



exemples\lit\littéral bas niveau 2.agn

Autre exemple :
Cahier des charges :
Faire clignoter tous les feux de la maquette.

Solution :



exemples\lit\littéral bas niveau 3.agn

Cet exemple montre l'utilisation des prédispositions. Elles sont utilisées ici pour créer une table d'adresses de variables. La table contient l'adresse de toutes les sorties qui pilotent les feux de la maquette.
A chaque cycle d'exécution, l'état de tous les feux est inversé.

Un problème se pose, les feux clignotent très vite et l'on ne voit pas grand chose. Modifions notre exemple.
Cahier des charges :
Il faut maintenant inverser un par un l'état des feux tous les dixièmes de seconde.

Solution :

exemples\lit\littéral bas niveau 4.agn

Langage littéral étendu

Le langage littéral étendu est un « sur ensemble » du langage littéral bas niveau.
Il permet d'écrire plus simplement et sous une forme plus concise des équations booléennes et numériques.
Il permet également d'écrire des structures de type IF ... THEN ... ELSE et WHILE ... ENDWHILE (boucle).
L'utilisation du langage littéral étendu est soumise aux mêmes règles que le langage littéral bas niveau, il utilise la même syntaxe pour les variables, les mnémoniques, les types de test (fronts, état complémenté, état immédiat) et les modes d'adressage.

Il est possible de « mixer » le langage littéral bas niveau et le langage littéral étendu.
Lorsque le compilateur de langage littéral détecte une ligne écrite en langage littéral étendu, il la décompose en instructions du langage littéral bas niveau, puis la compile.

Ecriture d'équations booléennes

Syntaxe générale :

```
« variable bool.=(type d'affectation) (variable bool. 2 opérateur 1 variable bool. 3... opérateur n -1 variable bool. n ) »
```

Le type d'affectation doit être précisé s'il est autre que « Affectation ».
Il peut être :

⇨ « (/) » : affectation complémentée,

⇨ « (0) » : mise à zéro,

⇨ « (1) » : mise à un.

Les opérateurs peuvent être :

⇨ « . » : et,

⇨ « + » : ou.

Les équations peuvent contenir plusieurs niveaux de parenthèses pour préciser l'ordre d'évaluation. Par défaut, les équations sont évaluées de la gauche vers la droite.

Exemples et équivalences avec le langage littéral bas niveau :

`o0=(i0)`	`equ o0 and i0`
`o0=(i0.i1)`	`equ o0 and i0 and i1`
`o0=(i0+i1)`	`equ o0 orr i0 eor i1`
`o0=(1)`	`set o0`
`o0=(0)`	`res o0`
`o0=(1)(i0)`	`set o0 and i0`
`o0=(0)(i0)`	`res o0 and i0`
`o0=(1)(i0.i1)`	`set o0 and i0 and i1`
`o0=(0)(i0+i1)`	`res o0 orr o0 eor i1`
`o0=(/)(i0)`	`neq o0 and i0`
`o0=(/)(i0.i1)`	`neq o0 and i0 and i1`
`o0=(/i0)`	`equ o0 and /i0`
`o0=(/i0./i1)`	`equ o0 and /i0 and /i1`
`o0=(c0=10)`	`equ o0 and c0=10`
`o0=(m200<100+m200>200)`	`equ o0 orr m200<100 eor m200>200`

Ecriture d'équations numériques

Syntaxe générale pour les entiers :

```
« variable num.1=[variable num.2 opérateur 1 ... opérateur n-1 variable num.n] »
```

Les équations peuvent contenir plusieurs niveaux de crochets pour préciser l'ordre d'évaluation. Par défaut, les équations sont évaluées de la gauche vers la droite.

Les opérateurs pour les entiers 16 et 32 bits peuvent être :

```
« + » : addition (équivalent à l'instruction ADA),
« - » : soustraction (équivalent à l'instruction SBA),
« * » : multiplication (équivalent à l'instruction MLA),
« / » : division (équivalent à l'instruction DVA),
« < » : décalage à gauche (équivalent à l'instruction RLA),
« > » : décalage à droite (équivalent à l'instruction RRA),
« & » : « Et » binaire (équivalent à l'instruction ANA),
« | »* : « Ou » binaire (équivalent à l'instruction ORA),
« ^ » : « Ou exclusif » binaire (équivalent à l'instruction XRA).
```

Les opérateurs pour les flottants peuvent être :

⇨ « + » : addition (équivalent à l'instruction ADA),

⇨ « - » : soustraction (équivalent à l'instruction SBA),

⇨ « * » : multiplication (équivalent à l'instruction MLA),

⇨ « / » : division (équivalent à l'instruction DVA).

On ne peut pas préciser de constante dans les équations sur les flottants. Si cela est nécessaire, il faut utiliser des prédispositions sur des flottants.

Les équations sur les flottants peuvent faire appeler les fonctions « SQR » et « ABS ».

Remarque : suivant la complexité des équations, le compilateur peut utiliser des variables intermédiaires. Ces variables sont les mots m53 à m59 pour les entiers 16 bits, les longs l53 à l59 pour les entiers 32 bits et les flottants f53 à f59.

Exemples et équivalences avec le langage littéral bas niveau :

```
m200=[10]                                   lda 10
                                            sta m200
m200=[m201]                                 lda m201
                                            sta m200
m200=[m201+100]                             lda m201
                                            ada 100
                                            sta m200
```

* Ce caractère est généralement associé à la combinaison de touches [ALT] + [6] sur les claviers.


```
m200=[m200+m201-m202]
```

```
lda m200
ada m201
sba m202
sta m200
```

```
m200=[m200&$ff00]
```

```
lda m200
ana $ff00
sta m200
```

```
f200=[f201]
```

```
lda f201
sta f200
```

```
f200=[f201+f202]
```

```
lda f201
ada f202
sta f200
```

```
f200=[sqr[f201]]
```

```
lda f201
sqr aaa
sta f200
```

```
f200=[sqr[abs[f201*100R]]]
```

```
lda f201
mla 100R
abs aaa
sqr aaa
sta f200
```

```
l200=[l201+$12345678L]
```

```
lda l201
ada $12345678L
sta l200
```

```
t0=[m200]
```

```
lda m200 ; transfert le contenu
sta t0 ; de m200 dans la consigne de t0
```

```
m1000=[t9]
```

```
lda t9 ; transfert la valeur
sta m1000; courante de t9 dans m1000
```


Structure de type IF ... THEN ... ELSE ...

Syntaxe générale :

```
IF(test)
      THEN
      action si test vrai
      ENDIF
      ELSE
      action si test faux
      ENDIF
```

Le test doit respecter la syntaxe décrite au chapitre consacré aux équations booléennes dans ce chapitre.
Seule une action si test vrai ou une action si test faux peut figurer.
Il est possible d'imbriquer plusieurs structures de ce type.
Les bits Système u90 à u99 sont utilisés comme variables temporaires pour la gestion de ce type de structure.

Exemples :

```
IF(i0)
      THEN
      inc m200            ; incrémenter le mot 200 si i0
      ENDIF

IF(i1+i2)
      THEN
      m200=[m200+10]      ; ajouter 10 au mot 200 si i1 ou i2
      ENDIF
      ELSE
      res m200            ; sinon effacer m200
      ENDIF
```

Structure de type WHILE ... ENDWHILE

Syntaxe générale :

```
WHILE(test)
      action à répéter tant que le test est vrai
ENDWHILE
```

Le test doit respecter la syntaxe décrite au chapitre consacré aux équations booléennes dans ce chapitre.
Il est possible d'imbriquer plusieurs structures de ce type.
Les bits Système u90 à u99 sont utilisés comme variables temporaires pour la gestion de ce type de structure.

Exemples :

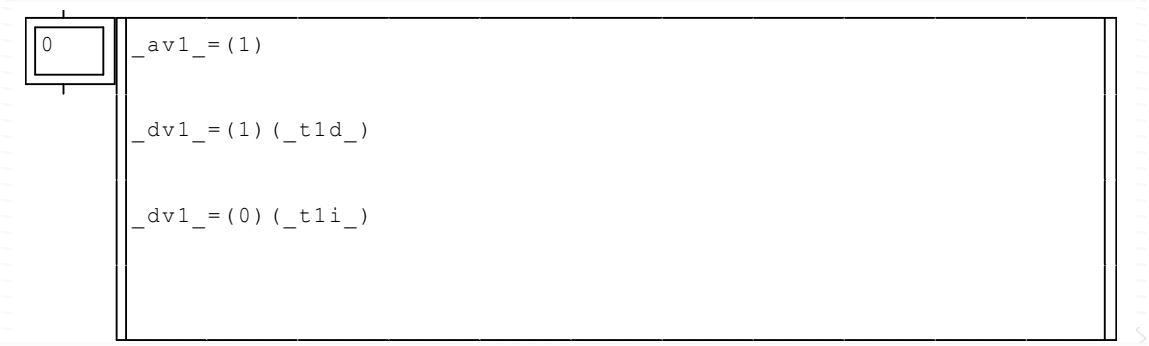
```
m200=[0]
WHILE(m200<10)
       set o(200)
       inc m200            ; incrémenter le mot 200
ENDWHILE
```

Cet exemple effectue une mise à un des sorties o0 à o9.

Exemple de programme en langage littéral étendu

Reprenons le premier exemple du chapitre précédent.

Solution :



exemples\lit\littéral étendu 1.agn

Enrichissons notre exemple avec des calculs.
Cahier des charges :
Calculer la vitesse en millimètres par seconde et en mètres par heure de la locomotive sur le trajet gauche vers droite.

Solution :

exemples\lit\littéral étendu 2.agn

Le mot 32 est utilisé pour lire le temps Système. La valeur est ensuite transférée dans des flottants pour pouvoir effectuer les calculs sans perte de précision.

Langage littéral ST

Le langage littéral ST est le langage littéral structuré défini par la norme CEI1131-3. Ce langage permet d'écrire des équations booléennes et numériques ainsi que des structures de programmation.

Généralités

Le langage littéral ST s'utilise aux mêmes endroits que le langage littéral bas niveau et le langage littéral étendu.

Des directives permettent de définir des sections en langage littéral ST :
« #BEGIN_ST » marque le début d'une section en langage ST.
« #END_ST » marque la fin d'une section en langage ST.
Exemple :

```
m200=[50]           ; langage littéral étendu
#BEGIN_ST
m201:=4;            (* langage ST *)
#END_ST
```

Il est également possible de choisir l'utilisation du langage ST pour tout un folio. Le choix s'effectue dans la boîte de dialogue de propriétés de chaque folio.

Dans un folio ou le langage ST est le langage par défaut il est possible d'insérer du langage littéral bas niveau et étendu en encadrant les lignes par deux directives « #END_ST » et « #BEGIN_ST ».

Pour le langage ST les commentaires doivent débuter par « (* » et se terminer par « *) ».

Les instructions du langage ST sont terminées par le caractère « ; ». Plusieurs instructions peuvent être écrites sur une même ligne.

Exemple :

```
o0:=1; m200:=m200+1;
```

Equations booléennes

La syntaxe générale est :

```
variable := équation booléenne;
```

L'équation booléenne peut être composée d'une constante, d'une variable ou de plusieurs variables séparées par des opérateurs.

Les constantes peuvent être : 0, 1, FALSE ou TRUE.

Exemples :

```
o0:=1;
o1:=FALSE;
```

Les opérateurs permettant de séparer plusieurs variables sont : + (ou), . (et), OR ou AND.

Le « Et » est prioritaire sur le « Ou ».

Exemple :
```
o0:=i0+i1.i2+i3;
```

Sera traité comme :
```
o0:=i0+(i1.i2)+i3;
```

Les parenthèses peuvent être utilisées dans les équations pour spécifier les priorités.

Exemple :
```
o0:=(i0+i1).(i2+i3);
```

Des tests numériques peuvent êtres utilisés.

Exemple :
```
o0:=m200>5.m200<100;
```

Equations numériques

La syntaxe générale est :
```
variable := équation numérique;
```

L'équation numérique peut être composée d'une constante, d'une variable ou de plusieurs variables et constantes séparées par des opérateurs.

Les constantes peuvent être des valeurs exprimées en décimal, hexadécimal (préfixe 16#) ou binaire (préfixe 2#).

Exemples :
```
m200:=1234;
m201:=16#aa55;
m202:=2#100000011101;
```

Les opérateurs permettant de séparer plusieurs variables ou constantes sont dans l'ordre de leurs priorités:
* (multiplication), / (division), + (addition), - (soustraction), & ou AND (et binaire), XOR (ou exclusif binaire), OR (ou binaire).

Exemples :
```
m200:=1000*m201;
m200:=m202-m204*m203;          (* équivalent à m200:=m202-(m204*m203) *)
```


Les parenthèses peuvent être utilisées dans les équations pour spécifier les priorités.

Exemple :
```
m200:=(m202-m204)*m203;
```

Ecriture de la consigne d'une temporisation

Syntaxe :
```
temporisation:=valeur ;
```

Exemples :
```
t0:=1000 ;
%t5:=%mw200 ;
```

Lecture du compteur d'une temporisation

Syntaxe :
```
variable:=temporisation;
```

Exemples :
```
m200:=t4;
%mw200:=%t8;
```

Structures de programmation

Test SI ALORS SINON

Syntaxe :
```
IF condition THEN action ENDIF;
```

et

```
IF condition THEN action ELSE action ENDIF;
```

Exemple :
```
if i0
       then o0:=TRUE;
       else
       o0:=FALSE;
       if i1 then m200:=4; endif;
endif ;
```

Boucle TANT QUE

Syntaxe :
```
WHILE condition DO action ENDWHILE;
```

Exemple :
```
while m200<1000
```



```
       do
       m200:=m200+1;
endwhile;
```

Boucle JUSQU'A CE QUE

Syntaxe :

```
REPEAT action UNTIL condition; ENDREPEAT;
```

Exemple :

```
repeat
       m200:=m200+1;
until m200=500
endrepeat;
```

Boucle DEPUIS JUSQU'A

Syntaxe :

```
FOR variable:=valeur de départ TO valeur de fin DO action ENDFOR;
```

ou

```
FOR variable:=valeur de départ TO valeur de fin BY pas DO action ENDFOR;
```

Exemple :

```
for m200:=0 to 100 by 2
       do
       m201:=m202*m201;
       endfor;
```

Sortie de boucle

Le mot clé « EXIT » permet de sortir d'une boucle.

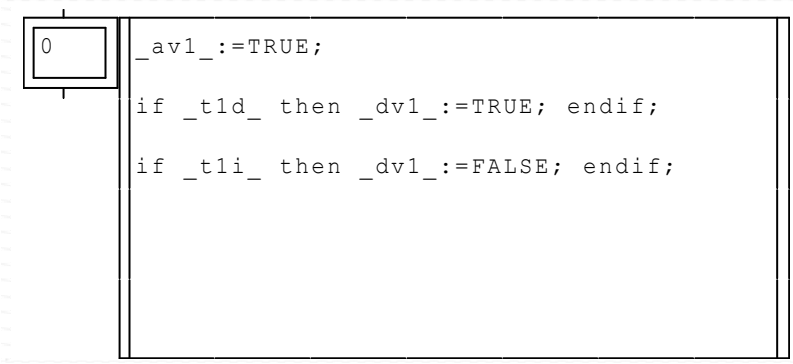
Exemple :

```
while i0
       m200:=m200+1;
       if m200>1000 then exit; endif;
endwhile;
```

Exemple de programme en langage littéral étendu

Reprenons le premier exemple du chapitre précédent.

Solution :



exemples\lit\littéral ST 1.agn

Organigramme

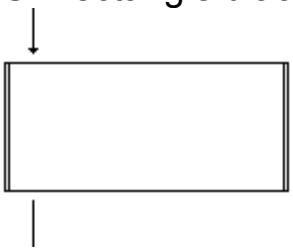
AUTOMGEN implémente une programmation de type « organigramme ».
Pour utiliser ce type de programmation il faut utiliser les langages littéraux Veuillez consulter les chapitres précédents pour apprendre à utiliser ces langages.
L'intérêt de la programmation sous forme d'organigramme est la représentation graphique d'un traitement algorithmique.
Contrairement au langage Grafcet, la programmation sous forme d'organigramme génère un code qui sera exécuté une fois par cycle de scrutation. Cela signifie que l'on ne peut rester en attente dans un rectangle d'organigramme, il faut obligatoirement que l'exécution sorte de l'organigramme pour pouvoir continuer à exécuter la suite du programme.
C'est un point très important à ne pas oublier lorsqu'on choisit ce langage.
Seuls des rectangles peuvent être dessinés. C'est le contenu des rectangles et les liaisons qui en partent qui déterminent si le rectangle est une action ou un test.

Dessin d'un organigramme

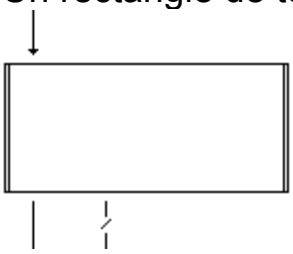
Les rectangles sont dessinés en choisissant la commande « Plus … / Boîte de code » du menu contextuel (cliquez sur le bouton droit de la souris sur le fond du folio pour ouvrir le menu contextuel).

Il faut placer un bloc ↓ (touche [<]) à l'entrée de chacun des rectangles, cette entrée doit être placée sur la partie supérieure du rectangle.
Si le rectangle représente une action, il y aura une seule sortie matérialisée par un bloc | (touche [E]) en bas et à gauche du rectangle.

Un rectangle d'action :



Si le rectangle représente un test, il y aura obligatoirement deux sorties. La première, matérialisée par un bloc | (touche [E]) en bas et à gauche représente la sortie si le test est vrai, la deuxième, matérialisée par un bloc / (touche [=]) se trouvant immédiatement à droite de l'autre sortie représente la sortie si le test est faux.
Un rectangle de test :



Les branches d'organigrammes doivent toujours se terminer par un rectangle sans sortie qui peut éventuellement rester vide.

Contenu des rectangles

Contenu des rectangles d'action

Les rectangles d'action peuvent contenir n'importe quelles instructions du langage littéral.

Contenu des rectangles de test

Les rectangles de test doivent contenir un test respectant la syntaxe de la partie test de la structure de type IF...THEN...ELSE... du langage littéral étendu.
Par exemple :

```
IF (i0)
```

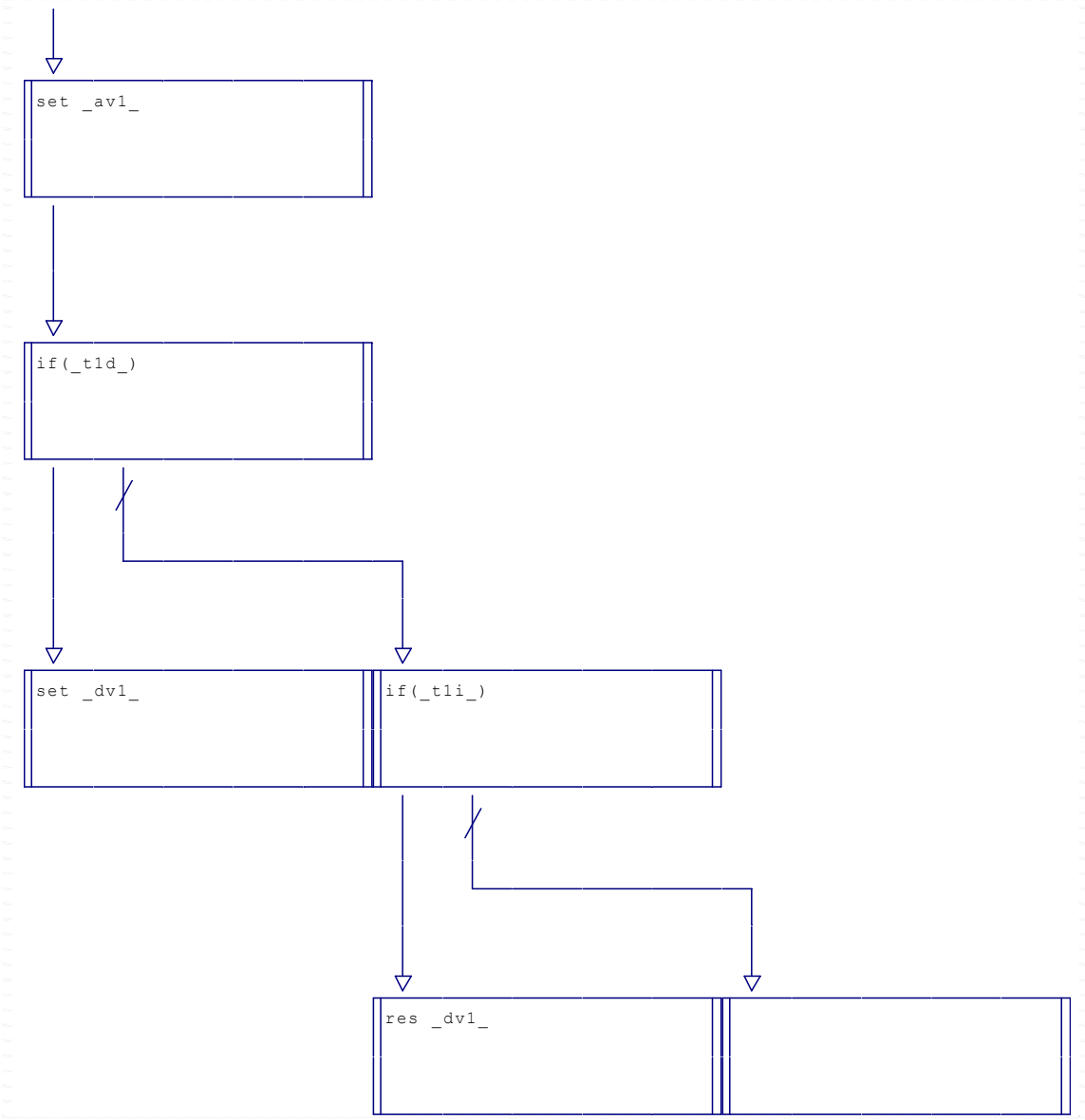
Il est possible d'écrire avant ce test des actions dans le rectangle de test.
Cela permet par exemple d'effectuer certains calculs avant le test.
Si par exemple nous voulons tester si le mot 200 est égal au mot 201 plus 4 :

```
m202=[m201+4]

IF(m200=m202)
```


Illustration

Notre premier exemple désormais classique, consistera à faire effectuer des allers et retours à une locomotive sur la voie 1 de la maquette.

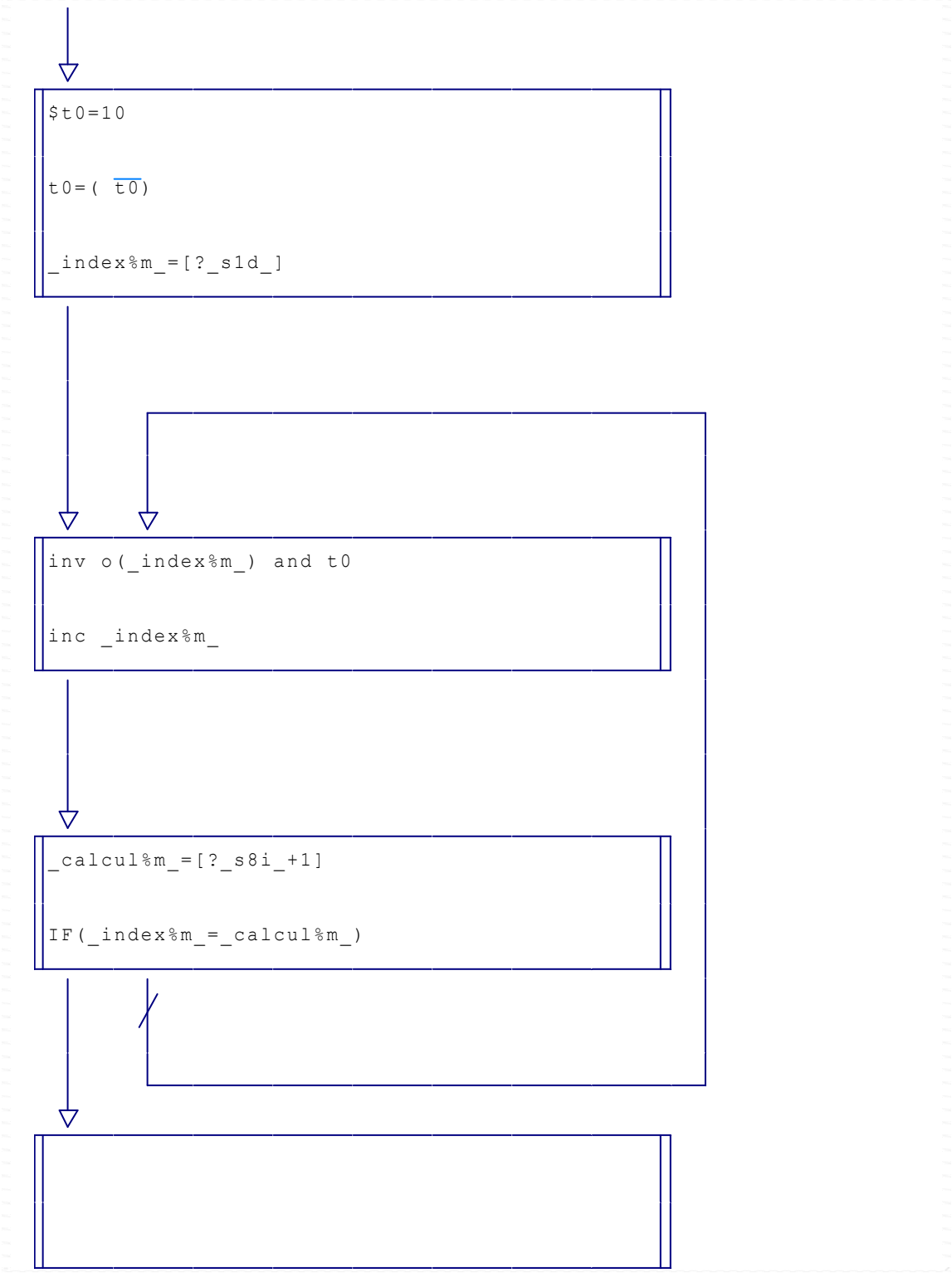
Solution :



exemples\organigramme\organigramme 1.agn

Deuxième exemple
Cahier des charges :
Faire clignoter tous les feux de la maquette. Les feux changeront d'état toutes les secondes.

Solution :

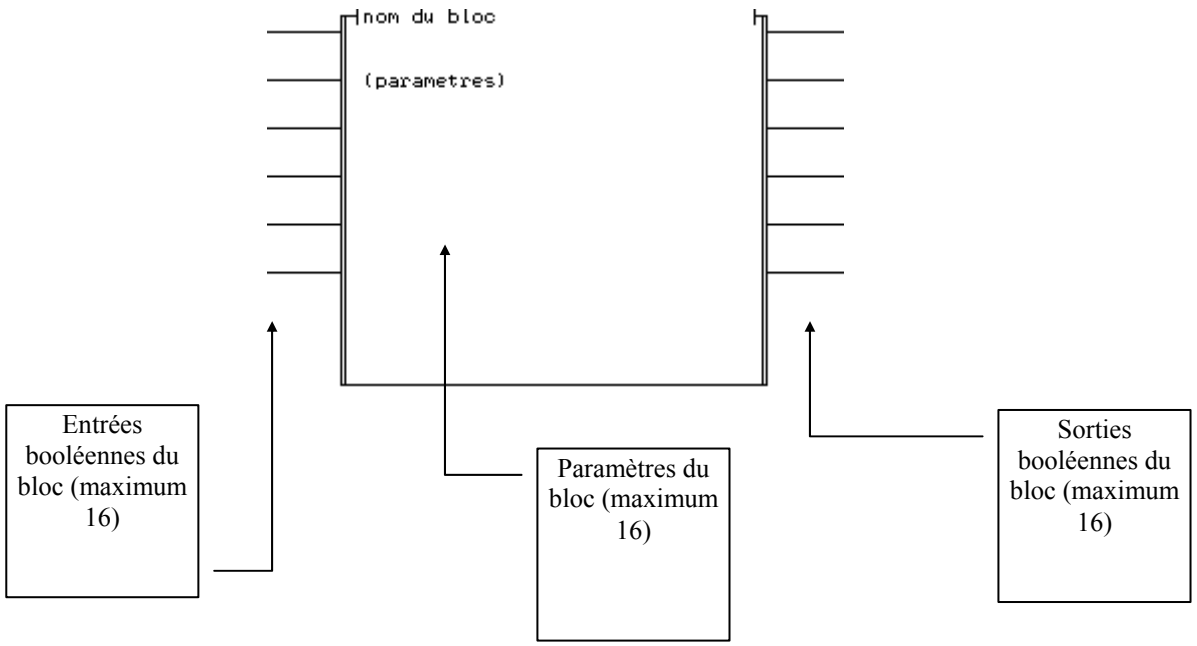


exemples\organigramme\organigramme 2.agn

Notez l'utilisation de symboles automatiques dans cet exemple.

Blocs fonctionnels

AUTOMGEN implémente la notion de blocs fonctionnels.
Cette méthode de programmation modulaire permet d'associer à un élément graphique un ensemble d'instructions écrites en langage littéral.
Les blocs fonctionnels sont définissables par le programmeur. Leur nombre n'est pas limité. Il est ainsi possible de constituer des ensembles de blocs fonctionnels permettant une conception modulaire et standardisée des applications.
Les blocs fonctionnels s'utilisent à l'intérieur de schémas de types logigramme ou ladder, ils possèdent de une à n entrées booléennes et de une à n sorties booléennes. Si le bloc doit traiter des variables autres que booléennes, alors celles-ci seront mentionnées dans le dessin du bloc fonctionnel. L'intérieur du bloc peut recevoir des paramètres : constantes ou variables.



Création d'un bloc fonctionnel

Un bloc fonctionnel est composé de deux fichiers distincts. Un fichier portant l'extension « .ZON » qui contient le dessin du bloc fonctionnel et un fichier portant l'extension « .LIB » qui contient une suite d'instructions écrites en langage littéral définissant le fonctionnement du bloc fonctionnel.
Les fichiers « .ZON » et « .LIB » doivent porter le nom du bloc fonctionnel. Par exemple si nous décidons de créer un bloc fonctionnel « MEMOIRE », nous devrons créer les fichiers « MEMOIRE.ZON »

(pour le dessin du bloc) et « MEMOIRE.LIB » (pour le fonctionnement du bloc).

Dessin du bloc et création du fichier « .ZON »

L'enveloppe d'un bloc fonctionnel est constituée d'une boîte de code à laquelle il faut ajouter des blocs dédiés aux blocs fonctionnels.

Dessin avec l'assistant

Ouvrez l'assistant (clic droit de la souris sur le folio puis « Assistant ») et choisissez « Blocs-fonctionnels ». Paramétrez ensuite le bloc avant de cliquer sur « OK ».

Dessin manuel

Pour dessiner un bloc fonctionnel il faut effectuer les opérations suivantes :

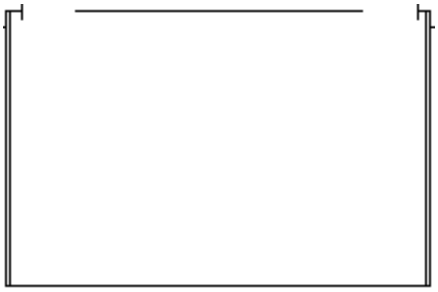
⇨ utiliser l'assistant (conseillé)

Ou :

⇨ dessiner une boîte de code (utilisez la commande « Plus …/Boîte de code » du menu contextuel) :

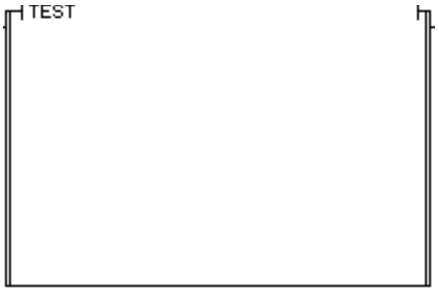
⇨ poser un bloc  (touche [8]) sur le coin supérieur gauche de la boîte de code :

⇨ poser un bloc  (touche [9]) sur le coin supérieur droit de la boîte de code :



⇨ effacer la ligne qui reste en haut du bloc (la touche [A] permet de poser des blocs blancs) :

⇨ cliquer avec le bouton gauche de la souris sur le coin supérieur gauche du bloc fonctionnel, entrer alors le nom du bloc fonctionnel qui ne doit pas dépasser 8 caractères (les fichiers « .ZON » et « .LIB » devront porter ce nom), presser ensuite [ENTER].



⇨ si des entrées booléennes supplémentaires sont nécessaires, il faut utiliser un bloc (touche [;]) ou (touche [:]), les entrées ainsi ajoutées doivent se trouver immédiatement en dessous de la première entrée, aucun espace libre ne doit être laissé,

⇨ si des sorties booléennes supplémentaires sont nécessaires il faut ajouter un bloc (touche [>]) ou (touche [?]), les sorties ainsi ajoutées doivent se trouver immédiatement en dessous de la première sortie, aucun espace libre ne doit être laissé,

⇨ l'intérieur du bloc peut contenir des commentaires ou des paramètres, les paramètres sont écrits entre accolades « {...} ». Tout ce qui n'est pas écrit entre accolades est ignoré par le compilateur. Il est intéressant de repérer l'usage des entrées et des sorties booléennes à l'intérieur du bloc.

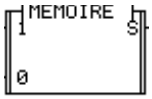
⇨ lorsque le bloc est terminé, il faut utiliser la commande « Sélectionner » du menu « Edition » pour sélectionner le dessin du bloc fonctionnel, puis le sauvegarder dans un fichier « .ZON » avec la commande « Copier vers » du menu « Edition ».

Création du fichier « .LIB »

Le fichier « .LIB » est un fichier texte contenant des instructions en langage littéral (bas niveau ou étendu). Ces instructions définissent le fonctionnement du bloc fonctionnel.
Une syntaxe spéciale permet de faire référence aux entrées booléennes du bloc, aux sorties booléennes du bloc et aux paramètres du bloc.
Pour faire référence à une entrée booléenne du bloc, il faut utiliser la syntaxe « {Ix} » où x est le numéro de l'entrée booléenne exprimé en hexadécimal (0 à f).
Pour faire référence à une sortie booléenne du bloc, il faut utiliser la syntaxe « {Ox} » où x est le numéro de la sortie booléenne exprimé en hexadécimal (0 à f).
Pour faire référence à un paramètre du bloc, il faut utiliser la syntaxe « {?x} » où x est le numéro du paramètre en hexadécimal (0 à f).
Le fichier .LIB peut être placé dans sous-répertoire « lib » du répertoire d'installation d'AUTOMGEN ou dans les ressources du projet.

Exemple simple de bloc fonctionnel

Créons le bloc fonctionnel « MEMOIRE » qui possède deux entrées booléennes (mise à un et mise à zéro) et une sortie booléenne (l'état de la mémoire).
Le dessin du bloc contenu dans le fichier « MEMOIRE.ZON » est :

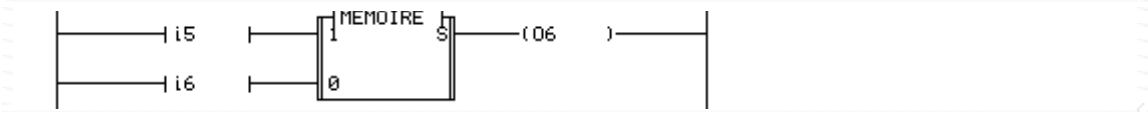


Le fonctionnement du bloc contenu dans le fichier « MEMOIRE.LIB » est :

```
{O0}=(1)({I0})
{O0}=(0)({I1})
```

Le bloc peut ensuite être utilisé de la façon suivante :

ou

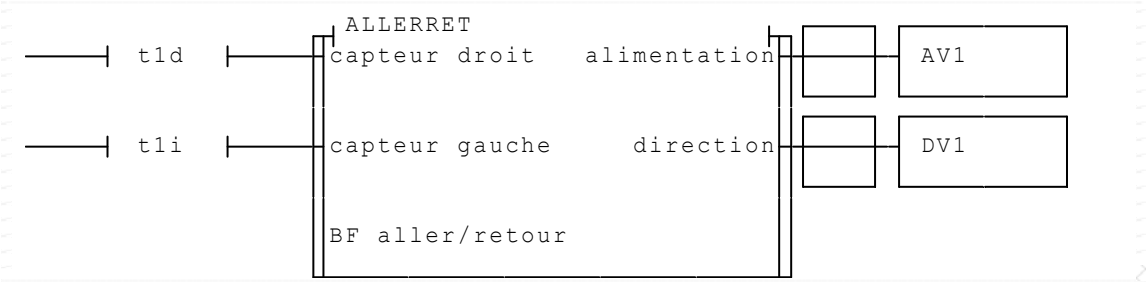


Pour utiliser un bloc fonctionnel dans une application il faut choisir la commande « Coller à partir de » du menu « Edition » et choisir le fichier « .ZON » correspondant au bloc fonctionnel à utiliser.

Illustration

Reprenons un exemple désormais classique.
Cahier des charges :
Aller et retour d'une locomotive sur la voie 1 de la maquette.

Solution :

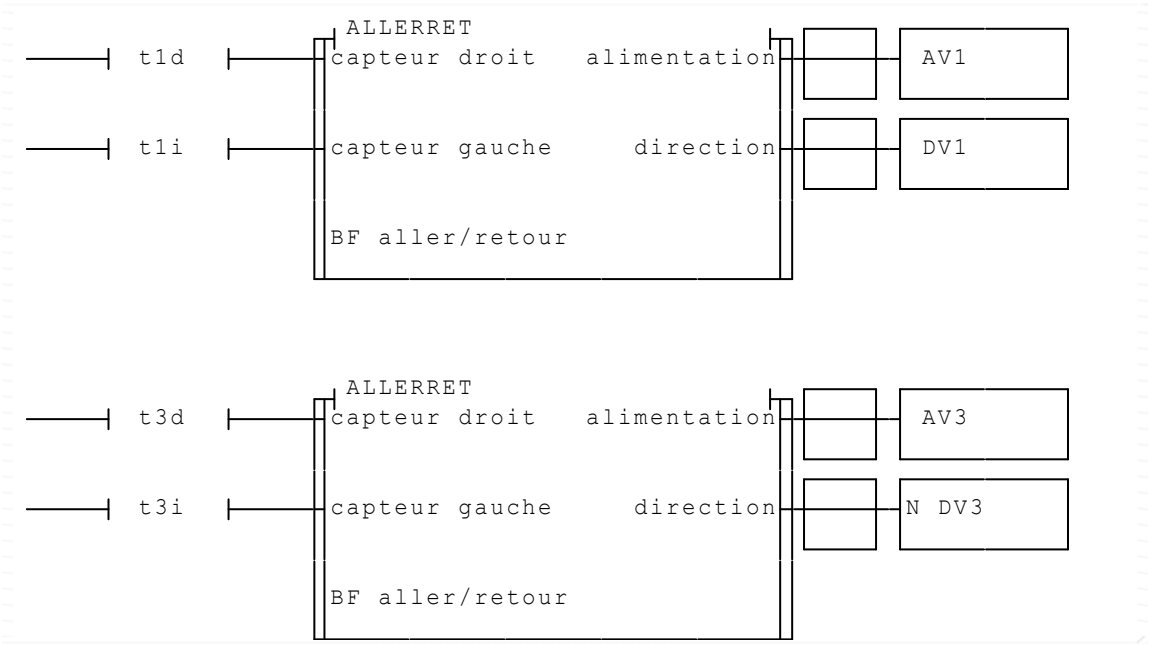


exemples\bf\bloc-fonctionnel 1.agn

```
; bloc fonctionnel ALLERRET
; aller retour d'une locomotive sur une voie
; les entrées booléennes sont les fins de course
; les sorties booléennes sont l'alimentation de la voie (0) et la
direction (1)

; toujours alimenter la voie
set {O0}

; piloter la direction en fonction des fins de course

{O1}=(1)({I0})
{O1}=(0)({I1})
```

Pour illustrer l'intérêt de l'utilisation des blocs fonctionnels, complétons notre exemple.
Cahier des charges :
Aller et retour de deux locomotives sur les voies 1 et 3.

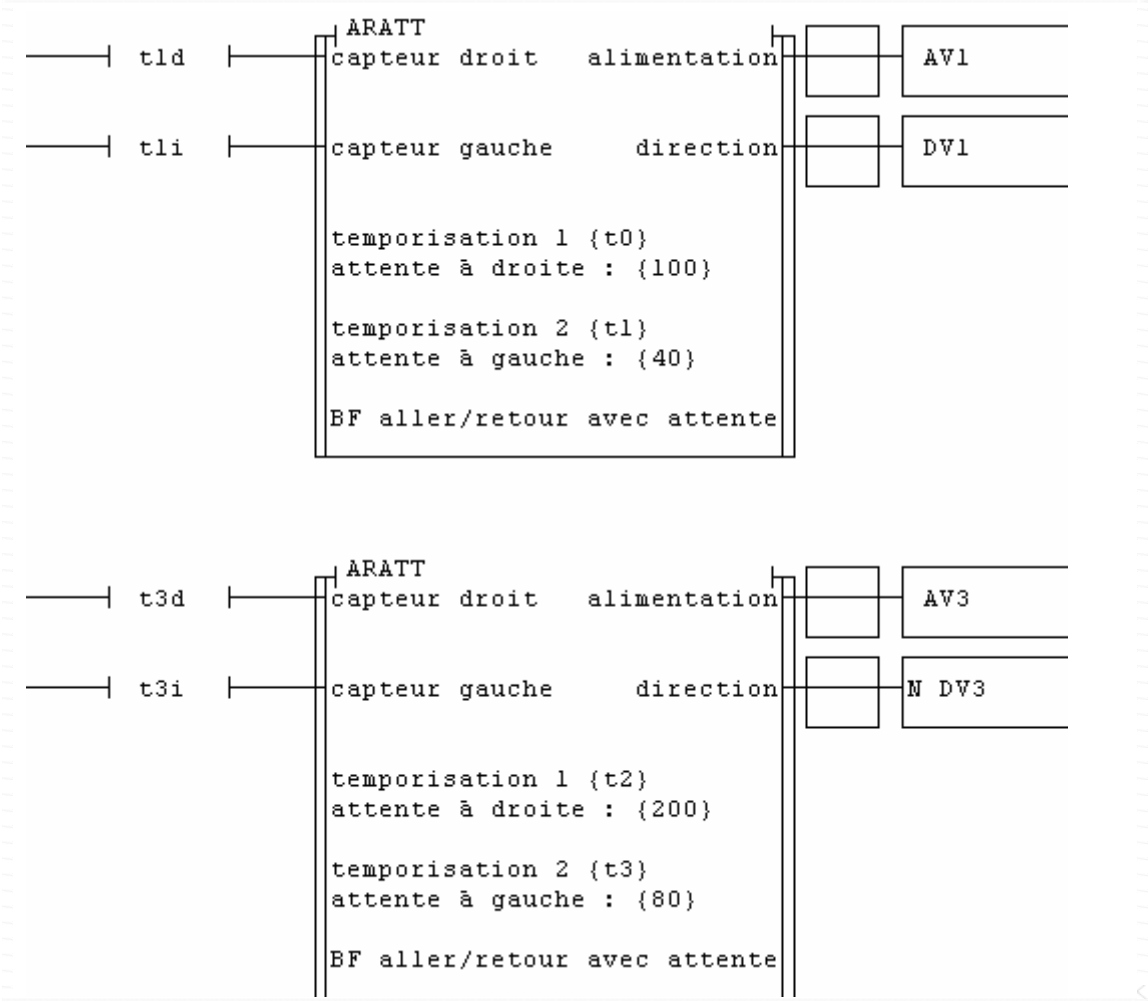
Solution :



exemples\bf\bloc-fonctionnel 2.agn

Cet exemple montre qu'avec le même bloc fonctionnel, il est aisé de faire fonctionner de façon identique différents modules d'une partie opérative.
Complétons notre exemple pour illustrer l'utilisation de paramètres.
Cahier des charges :
Les deux locomotives devront maintenant marquer une attente en bout de voie. Pour la locomotive 1 : 10 secondes à droite et 4 secondes à gauche, pour la locomotive 2 : 20 secondes à droite et 8 secondes à gauche.

Solution :




```
; bloc fonctionnel ARATT
; aller retour d'une locomotive sur une voie avec attente
; les entrées booléennes sont les fins de course
; les sorties booléennes sont l'alimentation de la voie (0) et la
direction (1)
; les paramètres sont :
;               0 : première temporisation
;               1 : durée de la première temporisation
;               2 : deuxième temporisation
;               3 : durée de la deuxième temporisation


; prédisposition des deux temporisations
${?0}={?1}
${?2}={?3}

; alimenter la voie si pas les fins de course ou si tempo. terminées
set {O0}
res {O0} orr {I0} eor {I1}
set {O0} orr {?0} eor {?2}

; gestion des temporisations
{?0}=({I0})
{?2}=({I1})

; piloter la direction en fonction des fins de course
{O1}=(1)({I0})
{O1}=(0)({I1})
```

exemples\bf\bloc-fonctionnel 3.agn

Complément de syntaxe

Une syntaxe complémentaire permet d'effectuer un calcul sur les numéros de variables référencées dans le fichier « .LIB ».
La syntaxe « ~+n » ajoutée à la suite d'une référence à une variable ou un paramètre, ajoute n.
La syntaxe « ~-n » ajoutée à la suite d'une référence à une variable ou un paramètre, soustrait n.
La syntaxe « ~*n » ajoutée à la suite d'une référence à une variable ou un paramètre, multiplie par n.
On peut écrire plusieurs de ces commandes à la suite, elles sont évaluées de la gauche vers la droite.
Ce mécanisme est utile lorsqu'un paramètre du bloc fonctionnel doit permettre de faire référence à une table de variables.

Exemples :

```
{?0}~+1
```

fait référence à l'élément suivant le premier paramètre, par exemple si le premier paramètre est m200 cette syntaxe fait référence à m201.

```
M{?2}~*100~+200
```

fait référence au troisième paramètre multiplié par 100 plus 200, par exemple si le troisième paramètre est 1 cette syntaxe fait référence à M 1*100 + 200 donc M300.

Blocs fonctionnels évolués

Cette fonctionnalité permet de créer des blocs fonctionnels très puissants avec plus de simplicité que les blocs fonctionnels gérés par des fichiers écrits en langage littéral. Cette méthode de programmation permet une approche de type analyse fonctionnelle.
N'importe quel folio ou ensemble de folios peut devenir un bloc fonctionnel (on parle parfois d'encapsuler un programme).
Le ou les folios décrivant le fonctionnement d'un bloc fonctionnel peuvent accéder aux variables externes du bloc fonctionnel : les entrées booléennes du bloc, les sorties booléennes et les paramètres.
Le principe d'utilisation et notamment l'utilisation des variables externes reste identique aux anciens blocs fonctionnels.

Syntaxe

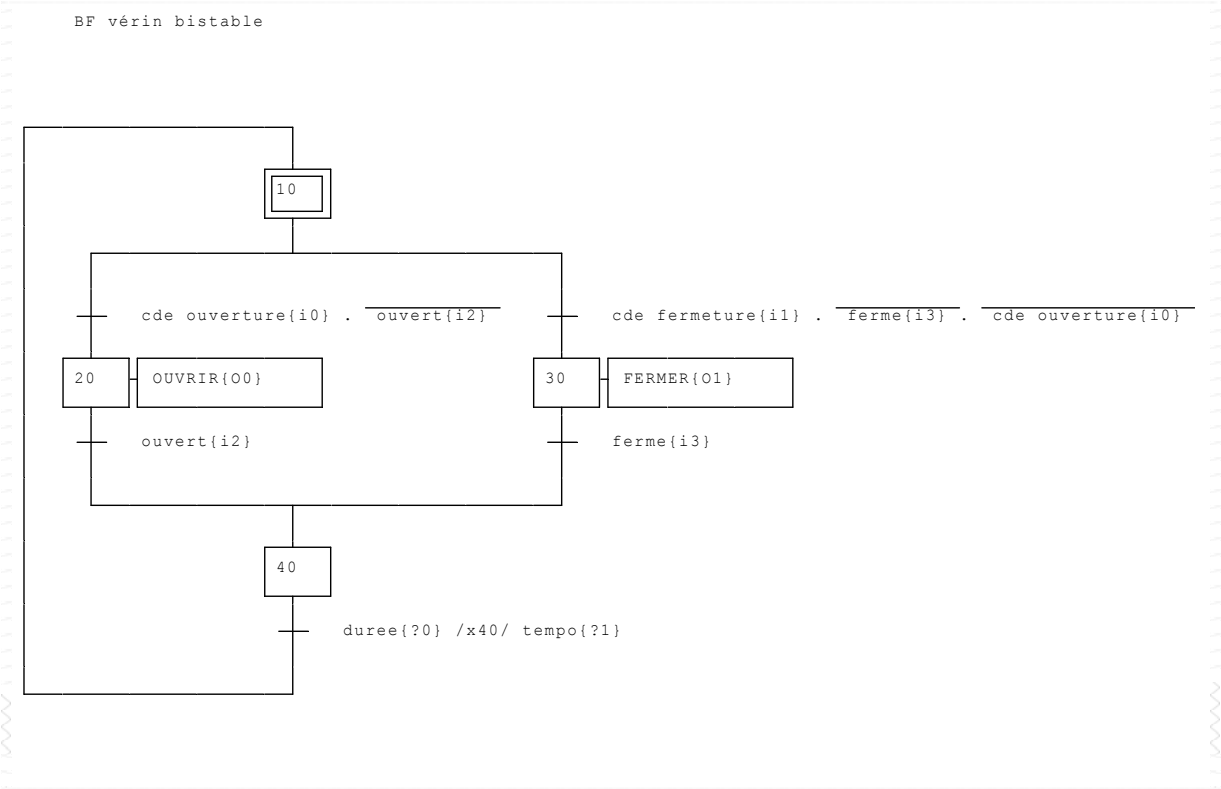
Pour référencer une variable externe d'un bloc fonctionnel il faut utiliser un mnémonique incluant le texte suivant : {In} pour référencer l'entrée booléenne n, {On} pour référencer la sortie booléenne n, {?n} pour référencer le paramètre n. Le mnémonique doit commencer par une lettre.

Différencier anciens et nouveaux blocs fonctionnels

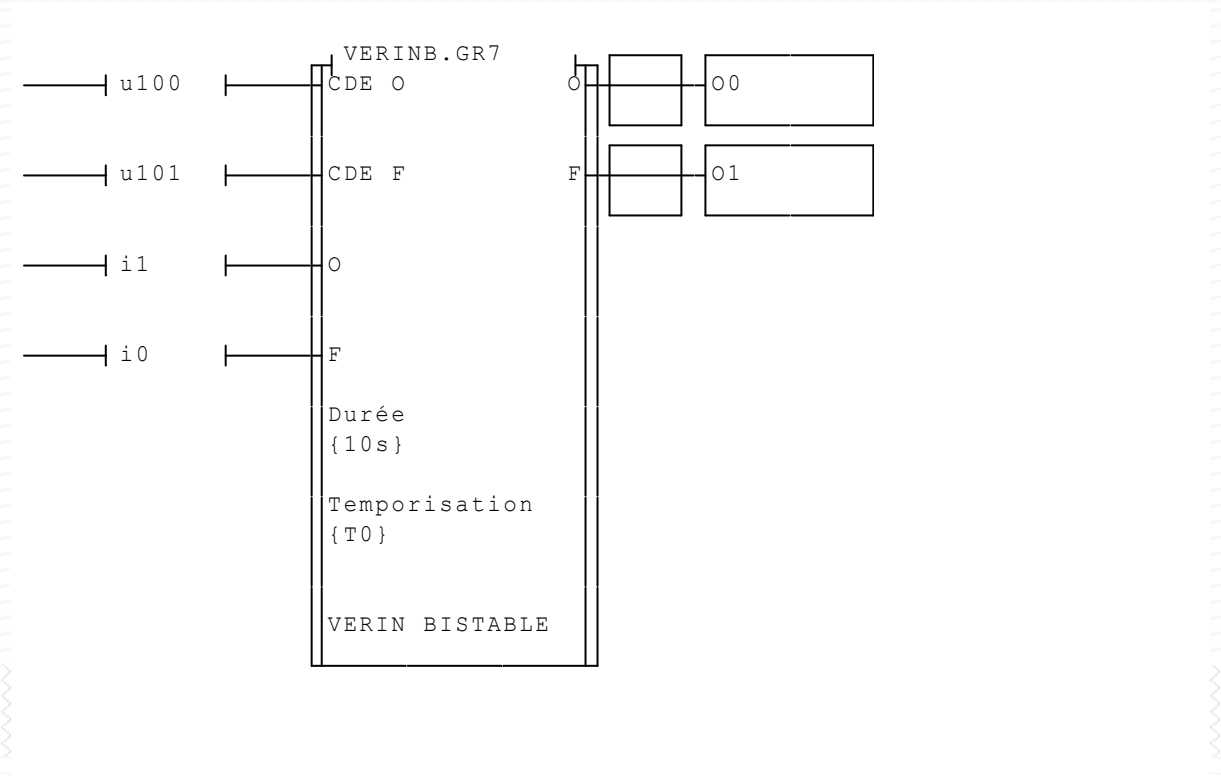
Le nom de fichier inscrit sur le dessin bloc fonctionnel indique s'il s'agit d'un ancien (géré par un fichier .LIB) ou d'un nouveau bloc fonctionnel (géré par un folio .GR7). Pour un ancien bloc fonctionnel le nom ne porte pas d'extension, pour un nouveau l'extension .GR7 doit être ajoutée. Le folio contenant le code qui gère le fonctionnement du bloc fonctionnel doit être intégré dans la liste des folios du projet. Dans les propriétés du folio, le type « Bloc-fonctionnel » doit être choisi.

Exemple

Contenu du folio VERINB :



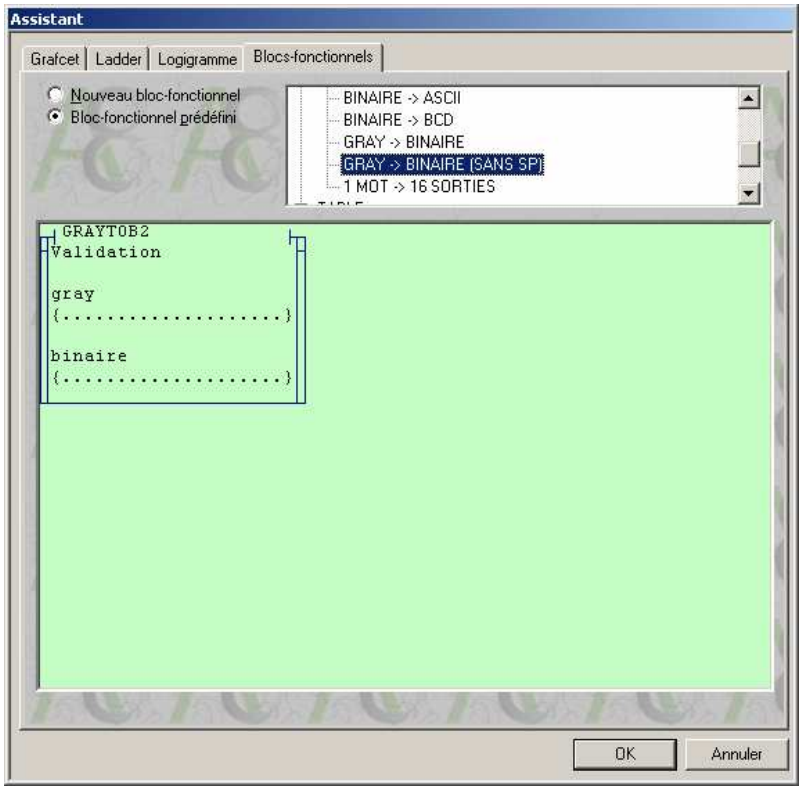
Appel du bloc fonctionnel :



exemples\bf\bloc-fonctionnel 3.agn

Blocs fonctionnels prédéfinis

Des blocs fonctionnels de conversion sont accessibles dans l'assistant d'AUTOMGEN : clic droit de la souris sur un folio, « Assistant », « Blocs-fonctionnels ».



Blocs de conversion

ASCTOBIN : conversion ASCII vers binaire
BCDTOBIN : conversion BCD vers binaire
BCD2BIN2 : (idem mais compatible avec les APIS n'intégrant pas la notion de sous-programmes)
BINTOASC : conversion binaire vers ASCII
BINTOBCD : conversion binaire vers BCD
BIN2BCD2 : (idem mais compatible avec les APIS n'intégrant pas la notion de sous-programmes)
GRAYTOB : conversion code gray vers binaire
GRAYTOB2 : (idem mais compatible avec les APIS n'intégrant pas la notion de sous-programmes)
16BINTOM : transfert de 16 variables booléennes dans un mot
MTO16 BIN : transfert d'un mot vers 16 variables booléennes

Blocs de temporisation

TEMPO : temporisation à la montée
PULSOR : sortie à créneau
PULSE : impulsion temporisée

Blocs de manipulations de chaîne de caractères

STRCMP : comparaison
STRCAT : concaténation
STRCPY : copie
STRLEN : calcul de la longueur
STRUPR : mise en minuscules
STRLWR : mise en majuscules

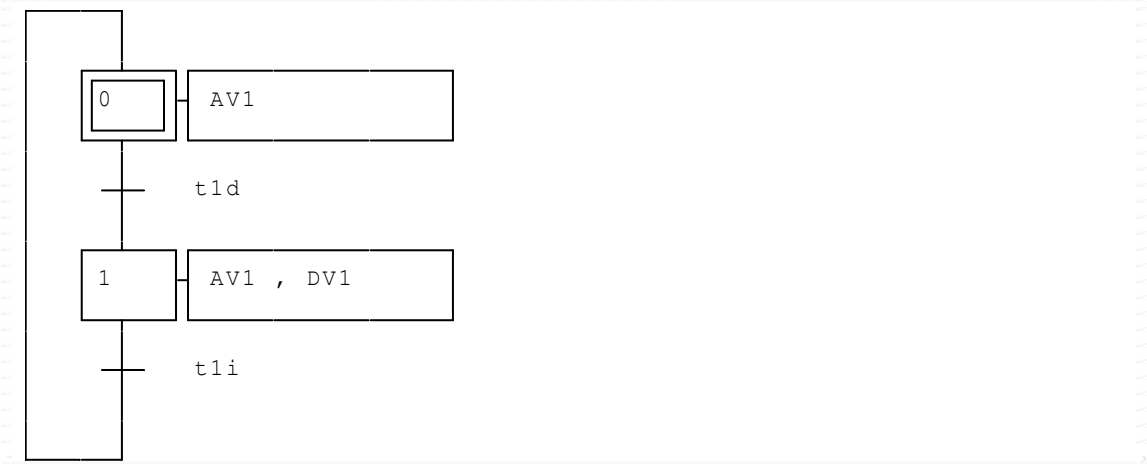
Blocs de manipulation de table de mots

COMP : comparaison
COPY : copie
FILL : remplissage

Techniques avancées

Code généré par le compilateur

Nous allons aborder dans ce chapitre, la forme du code généré par la compilation de tel ou tel type de programme.
L'utilitaire « CODELIST.EXE »* permet de traduire « en clair » un fichier de code intermédiaire « .EQU » (aussi appelé langage pivot).
Faisons l'expérience suivante : chargeons et compilons le premier exemple de programme du chapitre « Grafcet » : « simple1.agn » du répertoire « exemples\grafcet » :



Double cliquez sur l'élément « Fichiers générés/Code pivot » dans le navigateur.

* Cet utilitaire doit être exécuté à partir de la ligne de commande DOS.

Vous obtenez la liste d'instructions suivantes :

```
; Le code qui suit a été généré par la compilation de : 'Folio : GRAF1'
:00000000: RES x0  AND i0
:00000002: SET x0  AND b0
:00000004: SET x0  AND x1    AND i1
:00000007: RES x1  AND i1
:00000009: SET x1  AND x0    AND i0
; Le code qui suit a été généré par la compilation de : 'affectations
(actions Grafcet, logigrammes et ladder)'
:0000000C: EQU o0  ORR @x0   EOR @x1
:0000000F: EQU o23 AND @x1
```

Elle représente la traduction de l'application « simple1.agn » en instructions du langage littéral bas niveau.
Les commentaires indiquent la provenance des portions de code, cela est utile si une application est composée de plusieurs folios.
Obtenir cette liste d'instructions peut être utile pour répondre aux questions concernant le code généré par telle ou telle forme de programme ou l'utilisation de tel ou tel langage.
Dans certains cas « critiques », pour lesquels il est important de connaître des informations comme « au bout de combien de cycles cette action devient-elle vraie ? » le mode pas à pas et l'examen approfondi du code généré s'avèrent indispensables.

Optimisation du code généré

Plusieurs niveaux d'optimisation sont possibles.

Optimisation du code généré par le compilateur

L'option d'optimisation du compilateur permet de réduire sensiblement la taille du code généré. Cette directive demande au compilateur de générer moins de lignes de langage littéral bas niveau, ce qui a pour conséquence d'augmenter le temps de compilation.
Suivant les post-processeurs utilisés, cette option entraîne un gain sur la taille du code et ou le temps d'exécution. Il convient d'effectuer des essais pour déterminer si cette directive est intéressante ou pas suivant la nature du programme et le type de cible utilisée.
Il est en général intéressant de l'utiliser avec les post-processeurs pour cibles Z.

Optimisation du code généré par les post-processeurs

Chaque post-processeur peut posséder des options pour optimiser le code généré. Pour les post-processeurs qui génèrent du code constructeur veuillez consulter la notice correspondante.

Optimisation du temps de cycle : réduire le nombre de temporisations sur cibles Z

Pour les cibles Z, le nombre de temporisations déclarées influe directement sur le temps de cycle. Veillez à déclarer le minimum de temporisations en fonction des besoins de l'application.

Optimisation du temps de cycle : annuler la scrutation de certaines parties du programme

Seules les cibles acceptant les instructions JSR et RET supportent cette technique.
Des directives de compilation spéciales permettent de valider ou de « dévalider » la scrutation de certaines parties du programme.
Ce sont les folios qui définissent ces portions d'application.
Si une application est décomposée en quatre folios alors chacun d'eux pourra être indépendamment « validé » ou « dévalidé ».
Une directive « #C(condition) » placée sur un folio conditionne la scrutation du folio jusqu'au folio contenant une directive « #R ».
Cette condition doit utiliser la syntaxe définie pour les tests voir le chapitre Les tests

Exemple :
Si un folio contient les deux directives :

```
#C(m200=4)
#R
```

Alors tout ce qu'il contient ne sera exécuté que si le mot 200 contient 4.

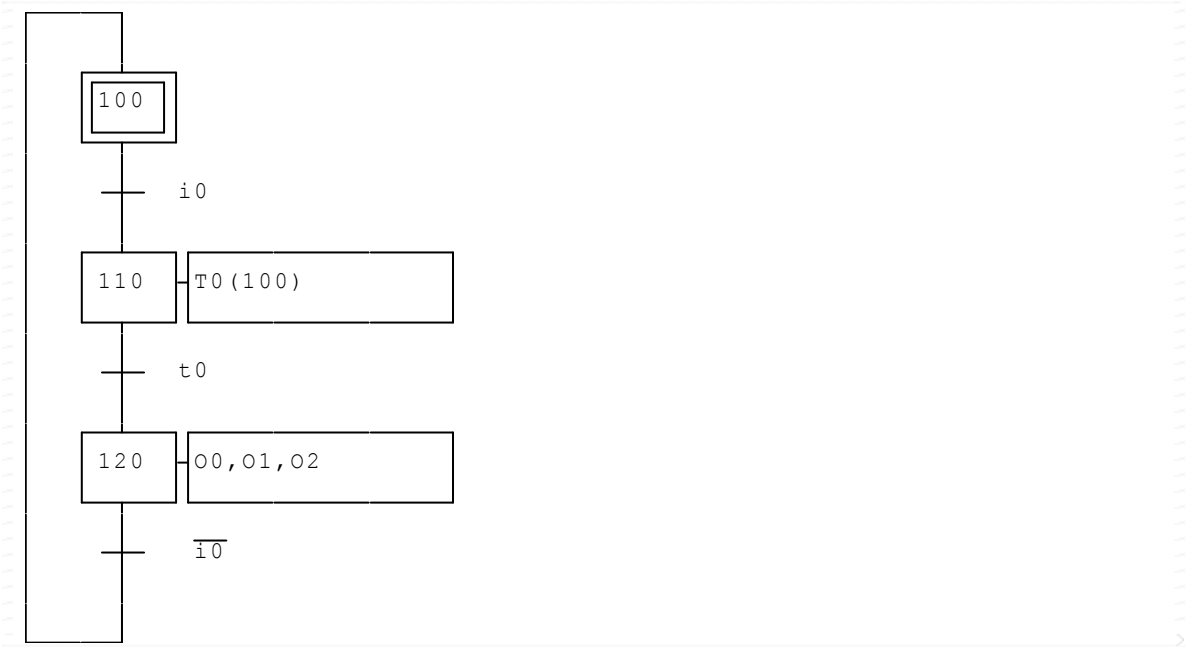
Exemples

A propos des exemples

Cette partie regroupe une série d'exemples donnant une illustration des différentes possibilités de programmation offertes par AUTOMGEN.
Tous ces exemples se trouvent dans le sous répertoire « exemple » du répertoire où a été installé AUTOMGEN.
Cette partie contient également des exemples plus complets et plus complexes développés pour une maquette de train. La description de cette maquette se trouve au début du manuel de référence langage.

Grafcet simple

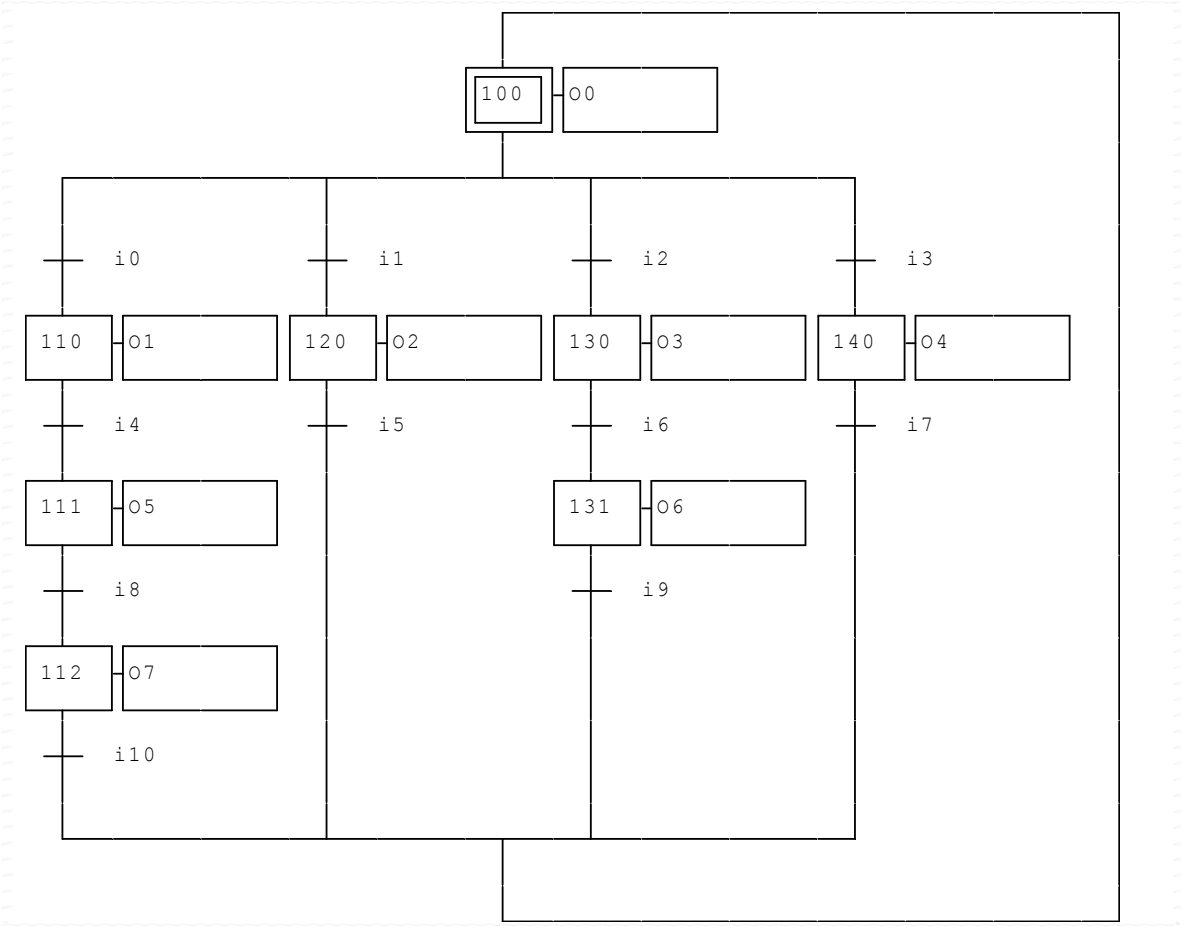
Ce premier exemple est un Grafcet simple en ligne :



exemples\grafcet\sample1.agn

- la transition entre l'étape 100 et l'étape 110 est constituée du test sur l'entrée 0,
- l'étape 110 active la temporisation 0 d'une durée de 10 secondes, cette temporisation est utilisée comme transition entre l'étape 110 et l'étape 120,
- l'étape 120 active les sorties 0, 1 et 2,
- le complément de l'entrée 0 sert de transition entre l'étape 120 et 100.

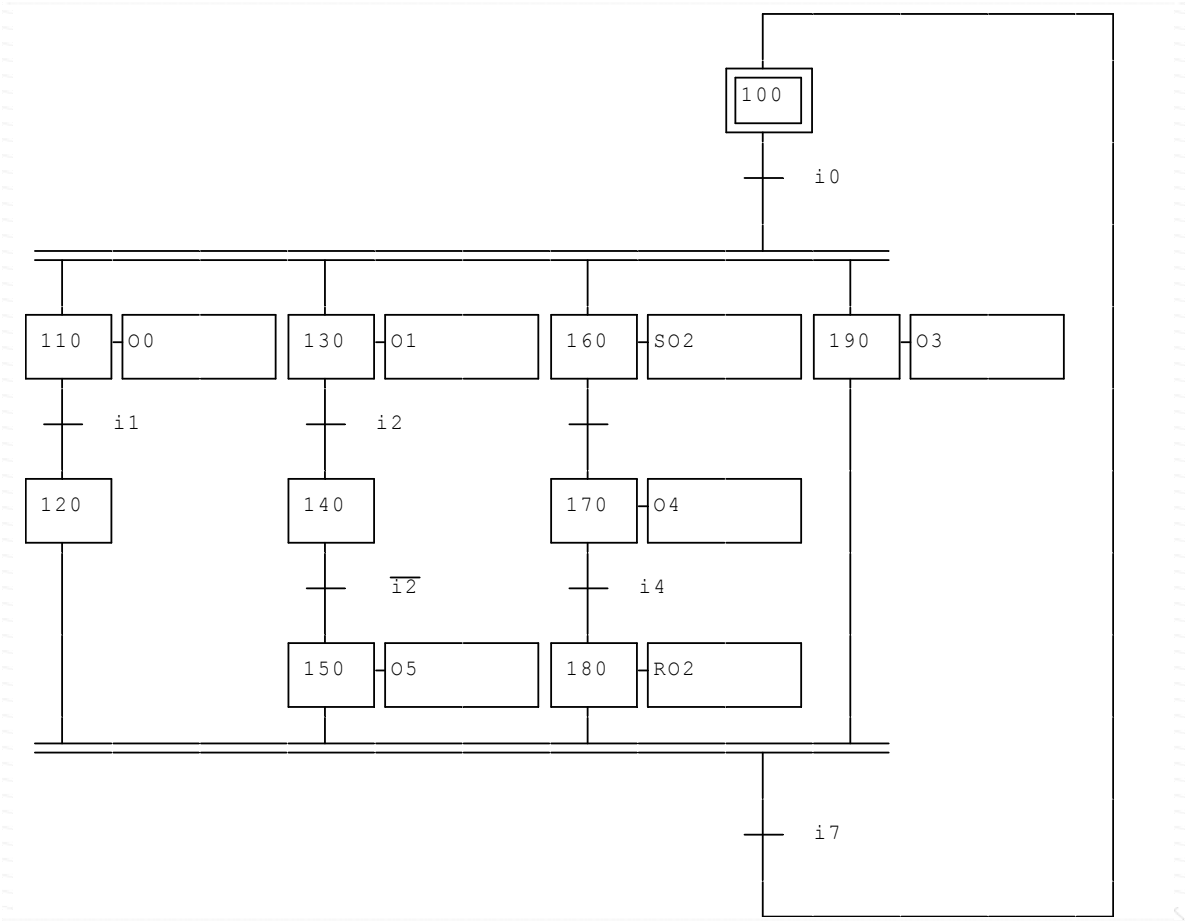
Grafcet avec divergence en OU



exemples\grafcet\sample2.agn

Cet exemple illustre l'utilisation des divergences et convergences en « Ou ». Le nombre de branches n'est limité que par la taille du folio. Il s'agit comme le prévoit la norme, d'un « Ou » non exclusif. Si par exemple, les entrées 1 et 2 sont actives, alors les étapes 120 et 130 seront mises à un.

Grafcet avec divergence en ET

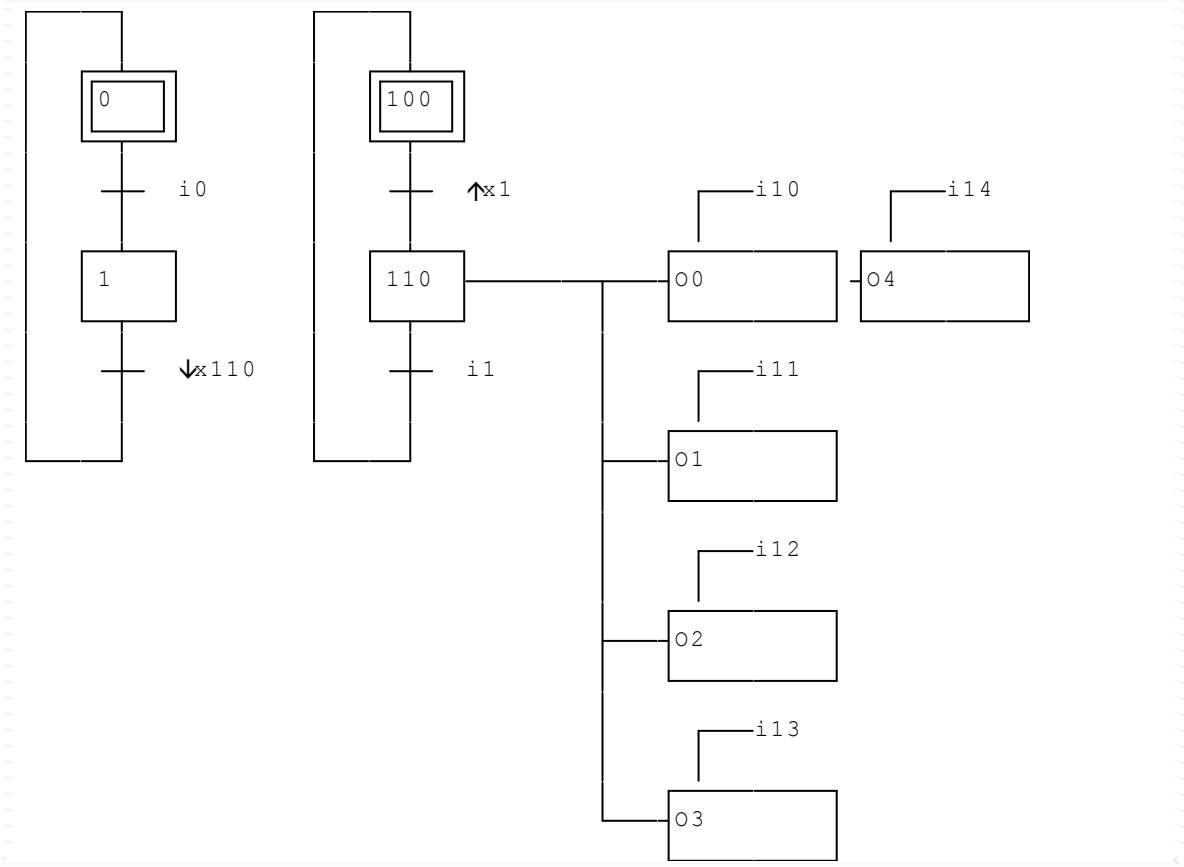


exemples\grafcet\sample3.agn

Cet exemple illustre l'utilisation des divergences et convergences en « Et ». Le nombre de branches n'est limité que par la taille du folio. Notons également au passage les points suivants :

- une étape peut ne pas comporter d'action (cas des étapes 100, 120, et 140),
- les ordres « S » et « R » ont été utilisés avec la sortie o2 (étapes 160 et 180),
- la transition entre l'étape 160 et 170 est laissée blanche, elle est donc toujours vraie, la syntaxe « =1 » aurait pu aussi être utilisée.

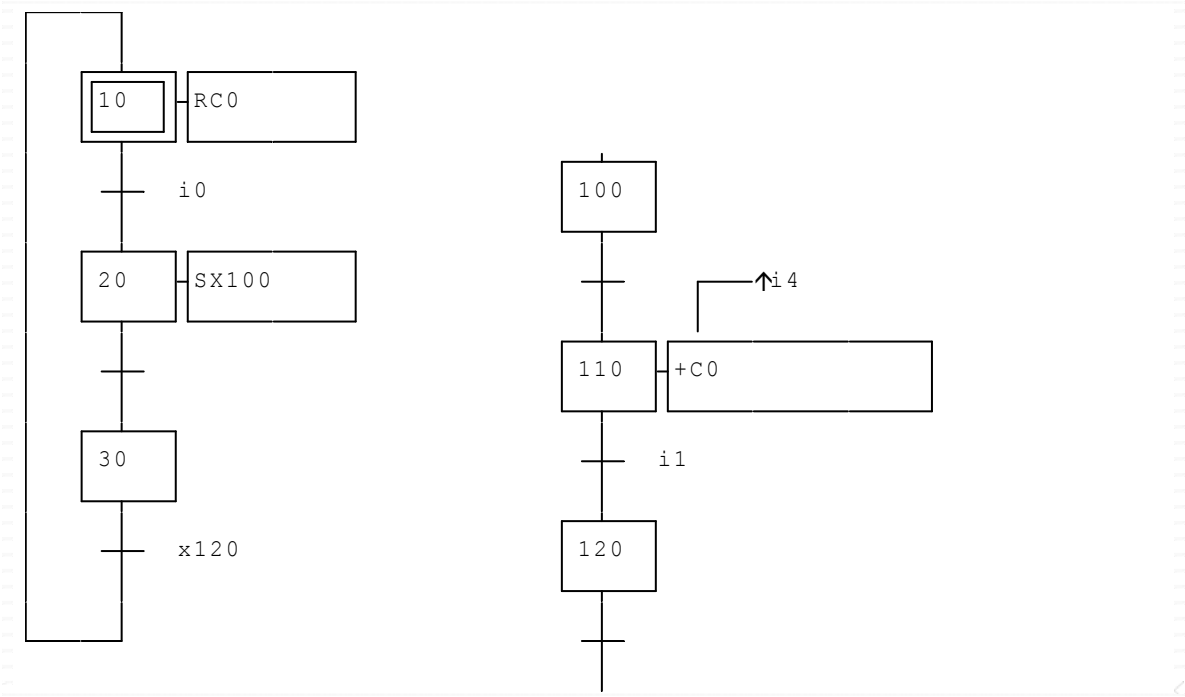
Grafcet et synchronisation



exemples\grafcet\sample4.agn

Cet exemple illustre une des possibilités offertes par AUTOMGEN pour synchroniser plusieurs Grafcets. La transition entre l'étape 100 et 110 « ↑x1 » signifie « attendre un front montant sur l'étape 1 ». La transition « ↓x110 » signifie « attendre un front descendant sur l'étape 110 ». L'exécution pas à pas de ce programme montre l'évolution exacte des variables et de leur front à chaque cycle. Ceci permet de comprendre exactement ce qu'il se passe lors de l'exécution. Notons également l'utilisation d'actions multiples associées à l'étape 110, qui sont ici conditionnées individuellement.

Forçage d'étapes

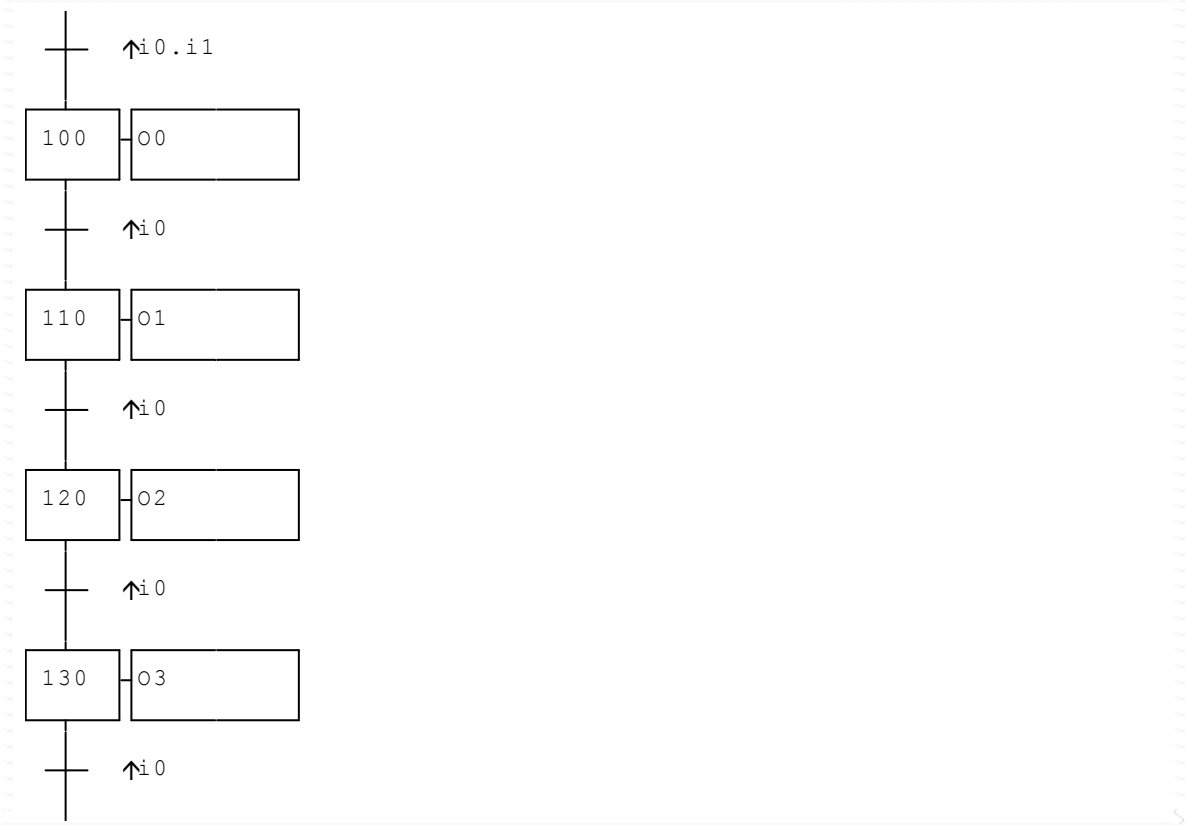


exemples\grafcet\sample5.agn

Dans cet exemple, un ordre « S » (mise à un) a été utilisé pour forcer une étape. AUTOMGEN autorise également le forçage d'un Grafcet entier (voir exemples 8 et 9) Le mode d'exécution pas à pas permet, pour cet exemple aussi, de comprendre de façon précise l'évolution du programme dans le temps. Notons également :

- ⇨ l'utilisation d'un Grafcet non rebouclé (100, 110, 120),
- ⇨ l'utilisation de l'ordre « RC0 » (mise à zéro du compteur 0),
- ⇨ l'utilisation de l'ordre « +C0 » (incrémenter le compteur 0), conditionné par le front montant de l'entrée 4, pour exécuter l'incrémentation du compteur, il faut donc que l'étape 110 soit active et qu'un front montant soit détecté sur l'entrée 4.

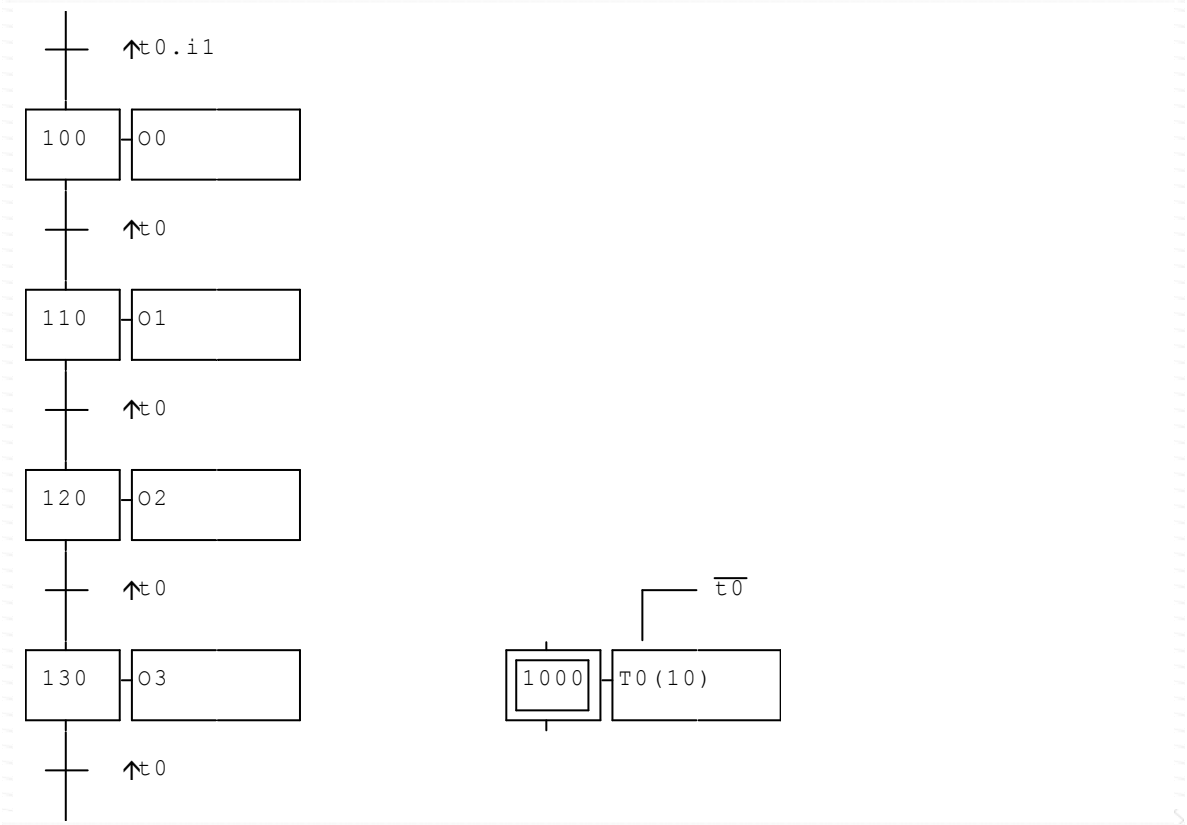
Etapes puits et sources



exemples\grafcet\sample6.agn

Nous avons déjà rencontré des formes similaires, dont la première étape était activée par un autre Grafcet. Ici l'activation de l'étape 100 est réalisée par la transition « ↑i0 . i1 » (front montant de l'entrée 0 et l'entrée 1). Cet exemple représente un registre à décalage. « i1 » est l'information à mémoriser dans le registre et « i0 » est l'horloge qui fait progresser le décalage. L'exemple 7 est une variante qui utilise une temporisation comme horloge.

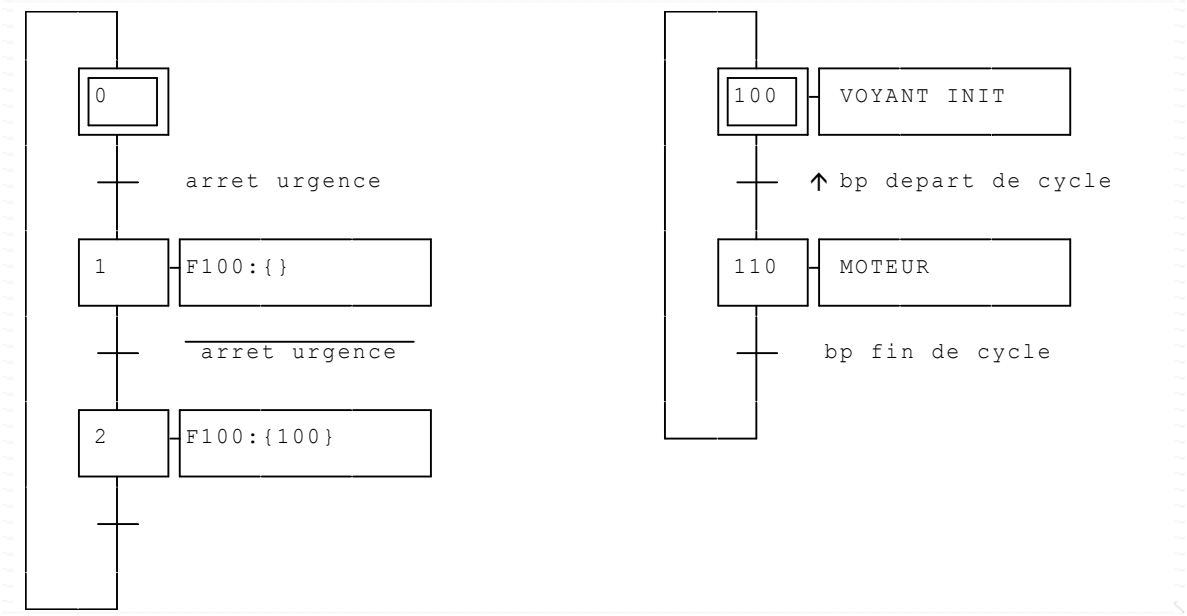
Etapes puits et sources



exemples\grafcet\sample7.agn

Nous retrouvons la structure de registre à décalage utilisé dans l'exemple 6. L'information de décalage est cette fois générée par une temporisation (t0). « ↑t0 » représente le front montant de la temporisation, cette information est vraie pendant un cycle lorsque la temporisation a fini de décompter. L'étape 1000 gère le lancement de la temporisation. On peut résumer l'action de cette étape par la phrase suivante : « activer le décomptage si celui-ci n'est pas terminé, dans le cas contraire, réarmer la temporisation ». Le diagramme de fonctionnement des temporisations de ce manuel vous aidera à comprendre le fonctionnement de ce programme.

Forçage de Grafcets

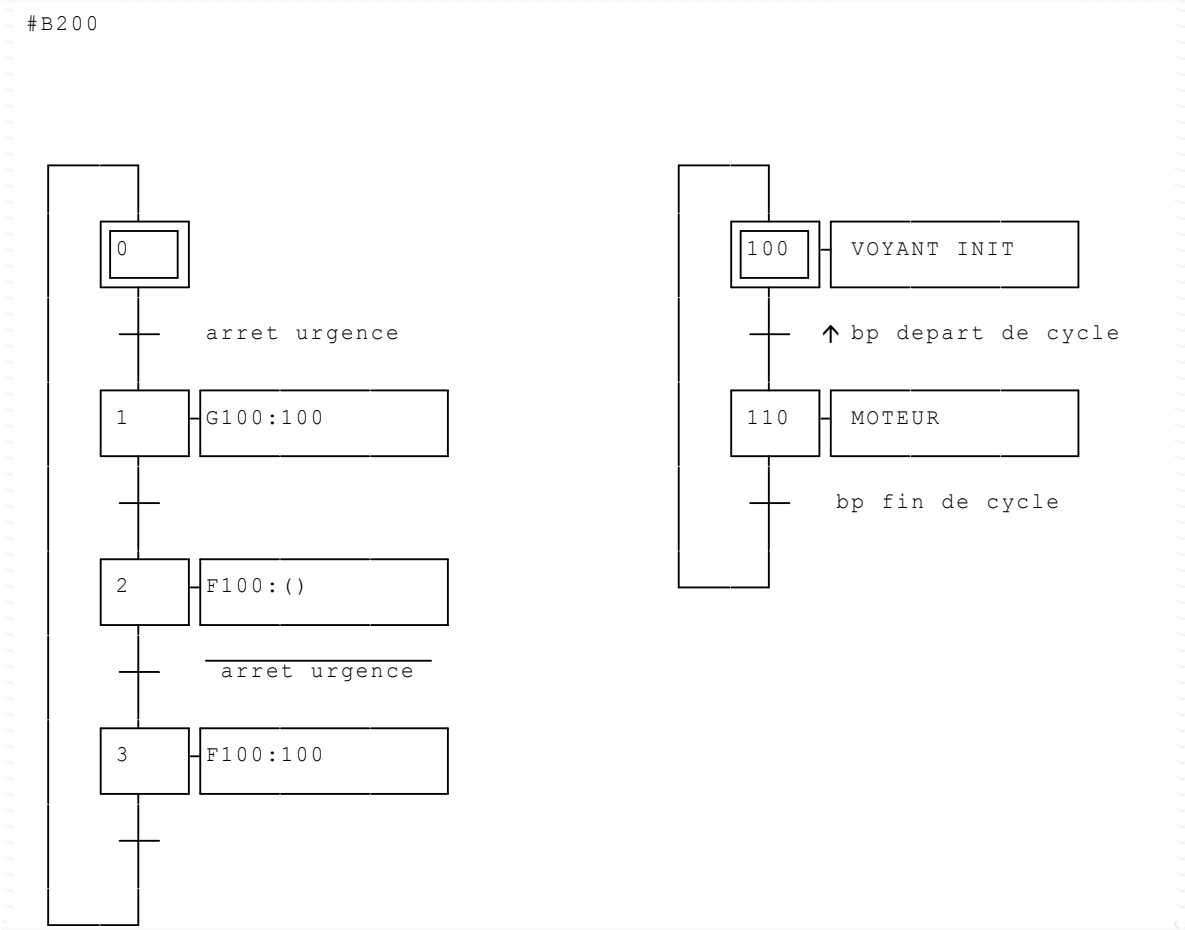


exemples\grafcet\sample8.agn

Cet exemple illustre l'utilisation d'une commande de forçage de Grafcet. L'ordre « F100:{} » signifie « forcer toutes les étapes du Grafcet dont une des étapes porte le numéro 100 à zéro ». L'ordre « F100:{100} » est identique mais force l'étape 100 à 1. Pour cet exemple, nous avons utilisé des symboles :

arret urgence	i0	
bp depart de cycle	i1	
bp fin de cycle	i2	
VOYANT INIT	o0	
MOTEUR	o1	

Mémorisation de Grafcets

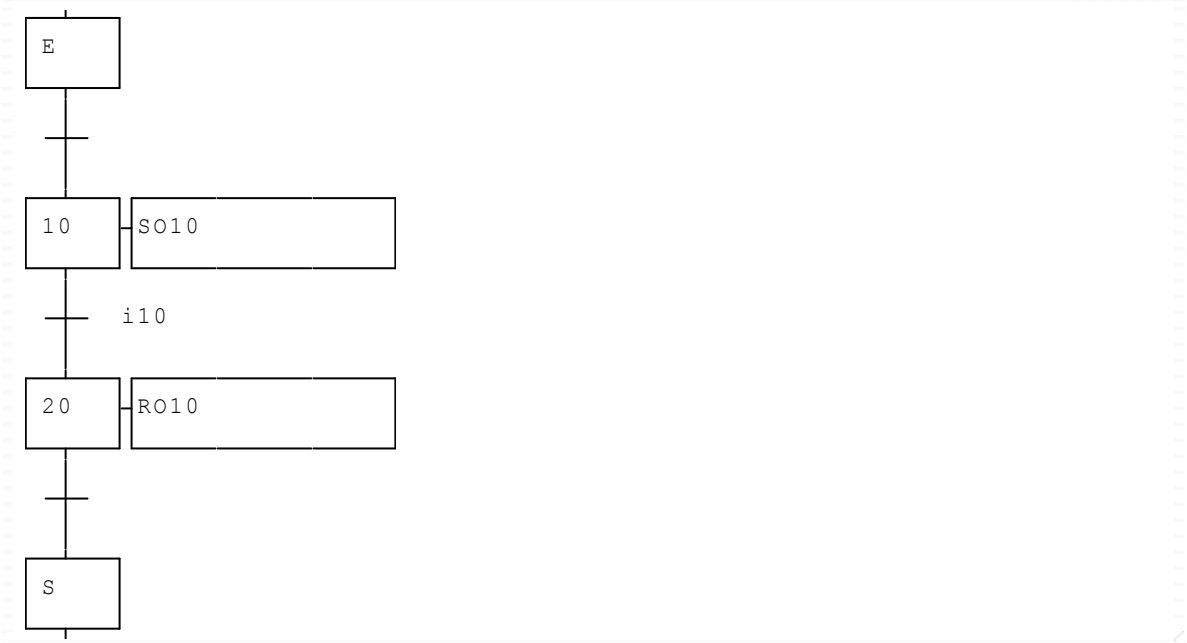


exemples\grafcet\sample9.agn

arret urgence	i0	
bp depart de cycle	i1	
bp fin de cycle	i2	
VOYANT INIT	o0	
MOTEUR	o1	

Cet exemple est une variante du programme précédent. L'ordre « G100:100 » de l'étape 1 mémorise l'état du Grafcet de production avant qu'il ne soit forcé à zéro. A la reprise, le Grafcet de production sera replacé dans l'état où il était avant la coupure, avec l'ordre « F100:100 ». L'état du Grafcet de production est mémorisé à partir du bit 100 (c'est le deuxième paramètre des ordres « F » et « G » qui précise cet emplacement), la directive de compilation « #B200 » réserve les bits u100 à u199 pour ce type d'utilisation. Notons qu'une directive « #B102 » aurait suffit ici puisque le Grafcet de production ne nécessite que deux bits pour être mémorisé (un bit par étape).

Grafcet et macro-étapes

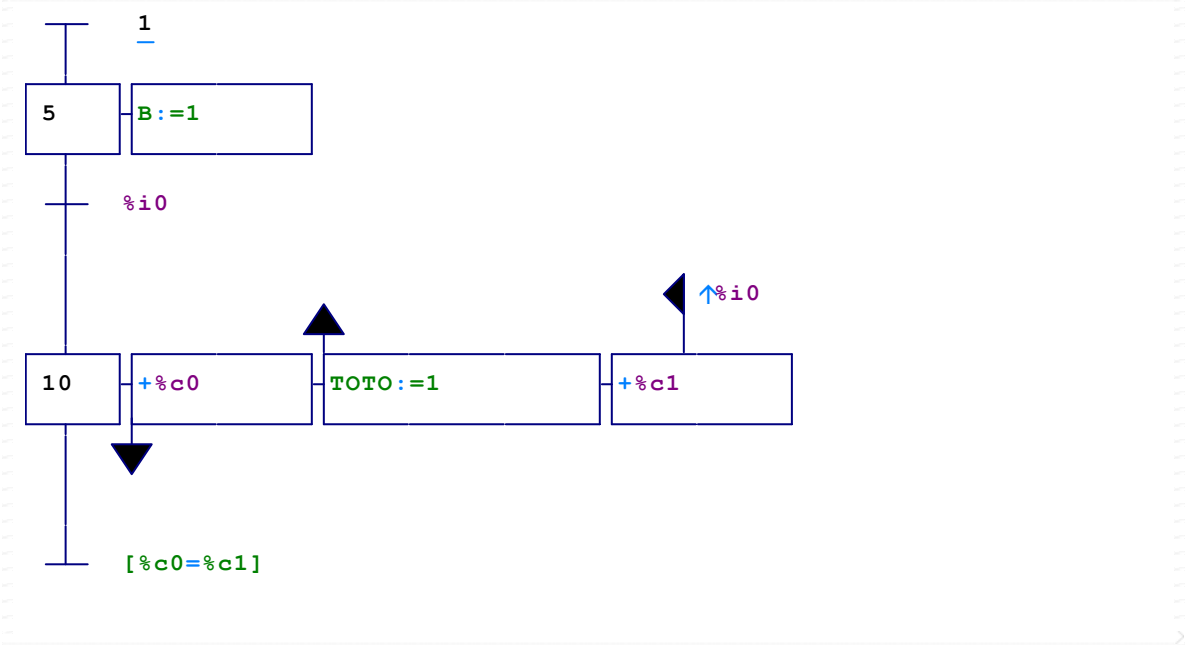


exemples\grafcet\sample11.agn

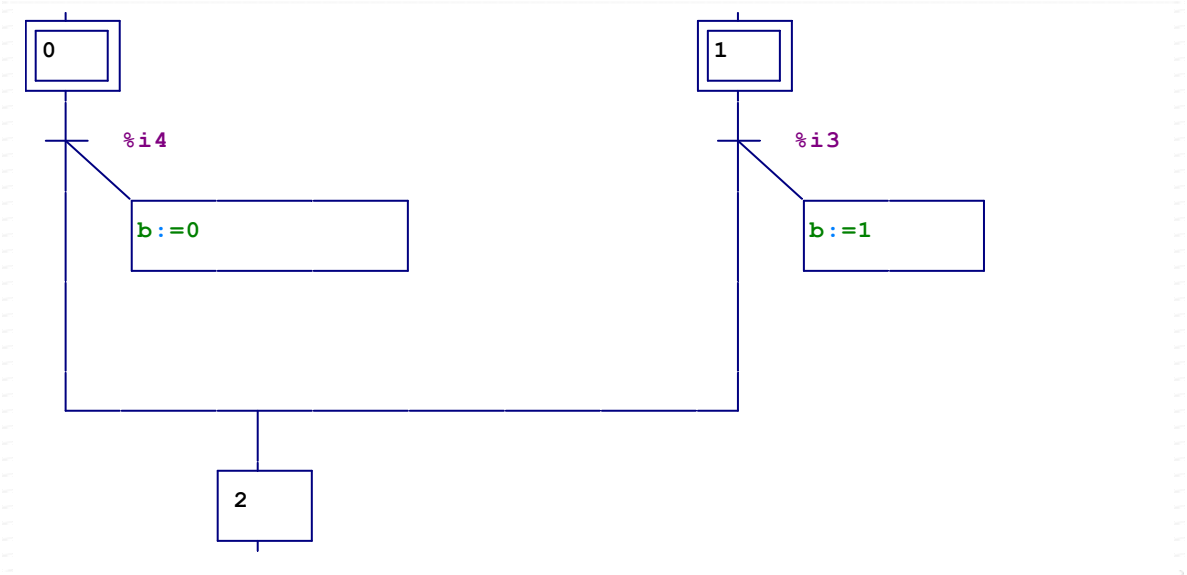
Cet exemple illustre l'utilisation des macro-étapes. Les folio « Macro étape 1 » et « Macro étape 3 » représentent l'expansion des macro-

étapes avec les étapes d'entrées et de sorties. Les étapes 1 et 3 du folio « Programme principal » sont définies comme macro-étapes. L'accès aux expansions de macro-étapes en visualisation est réalisé en cliquant avec le bouton gauche de la souris sur les macro-étapes.

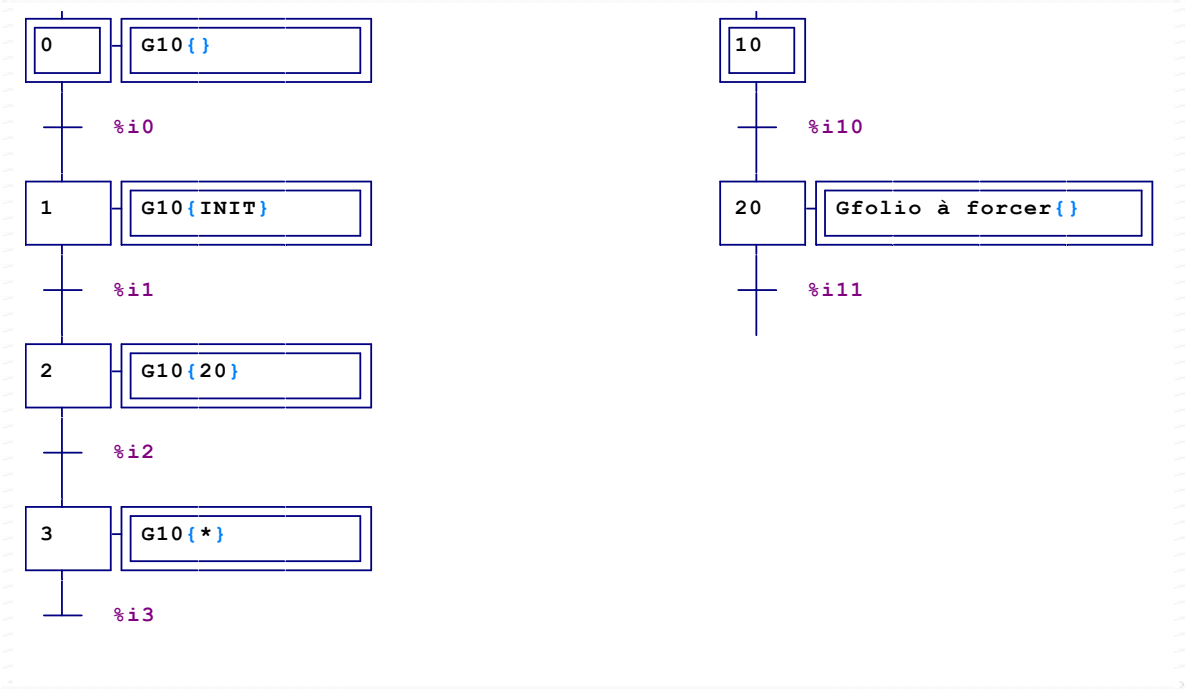
Eléments de la norme 60848



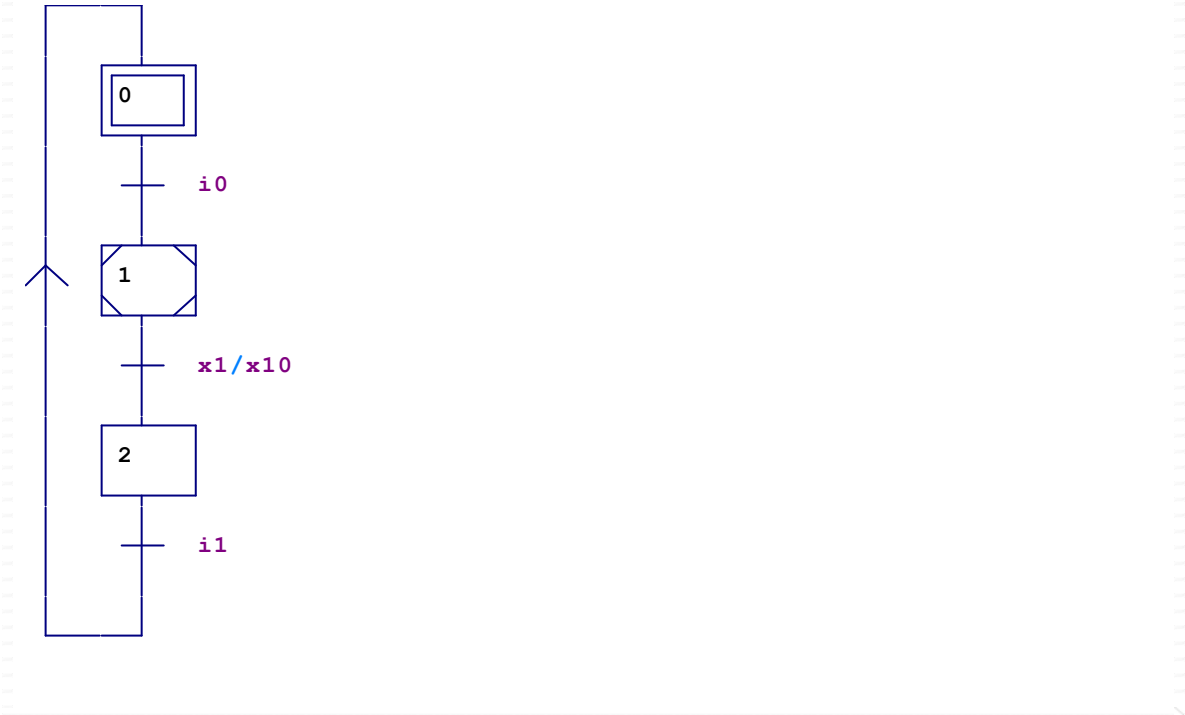
exemples\grafcet\norme 60848 exemple 1.agn



exemples\grafcet\norme 60848 exemple 2.agn

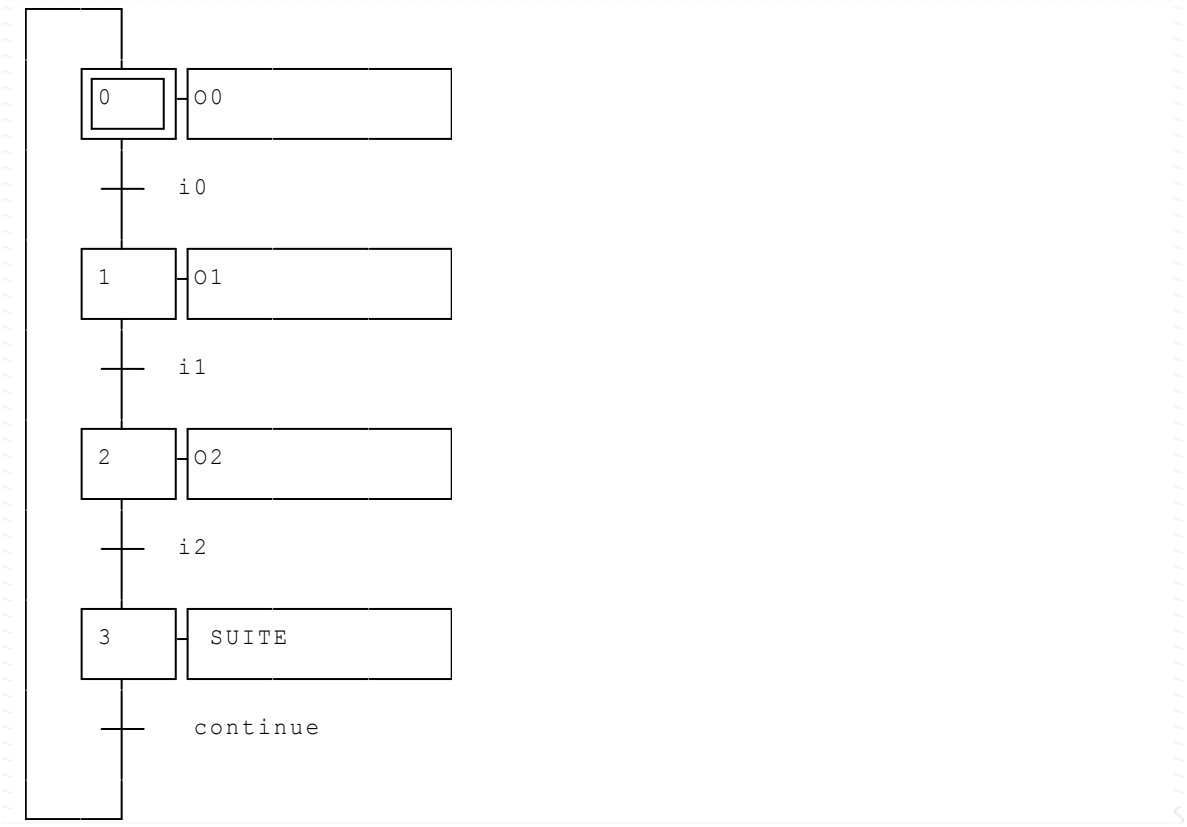


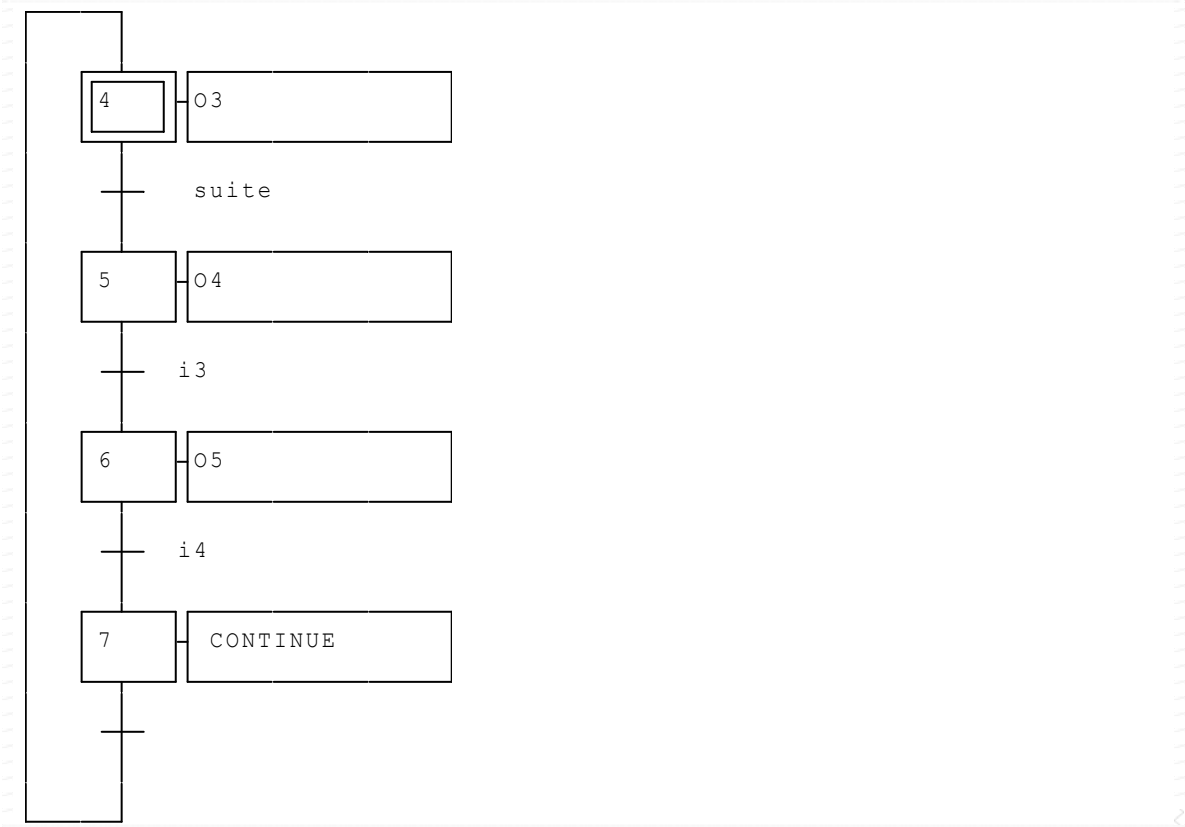
exemples\grafcet\norme 60848 exemple 3.agn



exemples\grafcet\norme 60848 exemple 4.agn

Folios chaînés

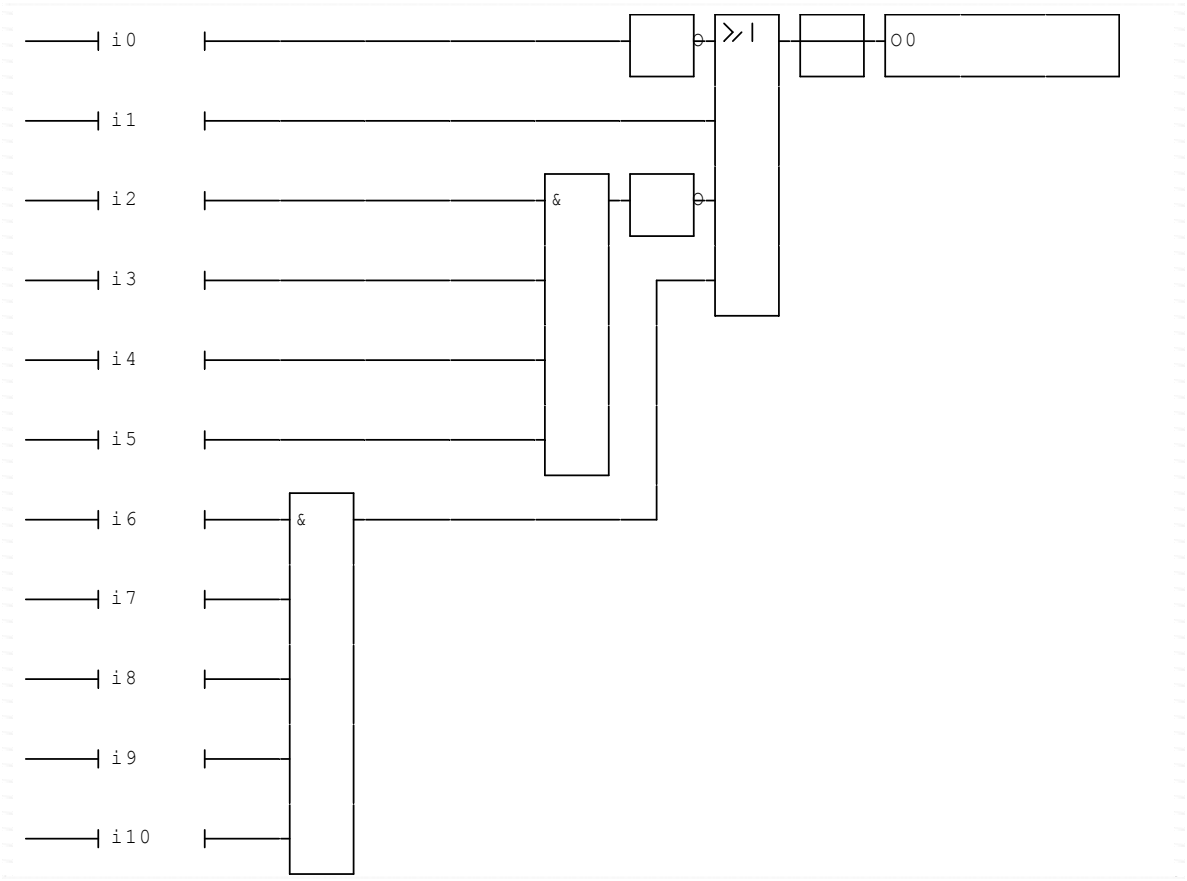




exemples\grafcet\sample12.agn

Dans cet exemple, deux folios ont été utilisés pour écrire le programme. Les symboles « _SUITE_ » et « _CONTINUE_ » ont été déclarés comme bits (voir le fichier des symboles) et permettent de faire le lien entre les deux Grafcets (c'est une autre technique de synchronisation utilisable avec AUTOMGEN).

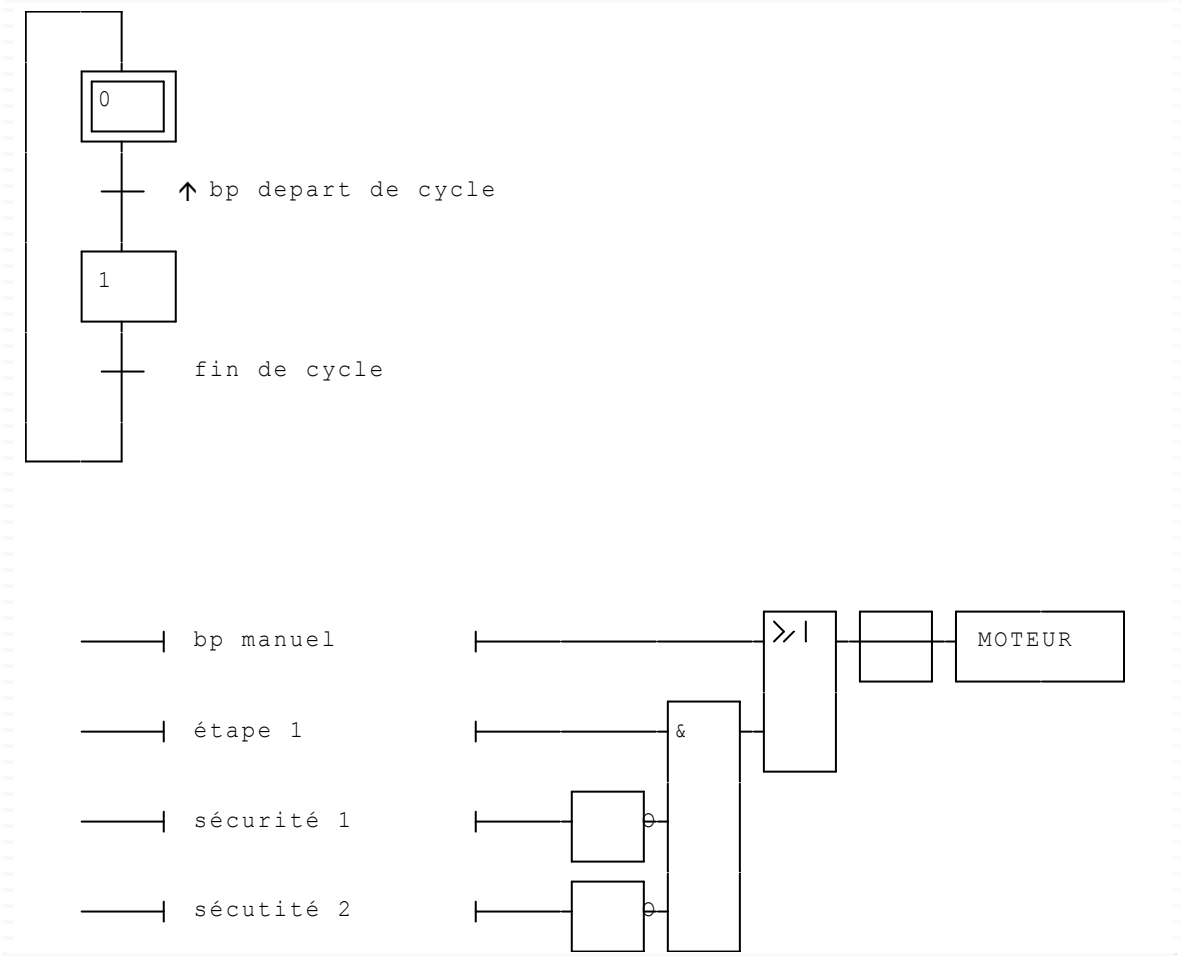
Logigramme



exemples\logigramme\sample14.agn

Cet exemple développé en logigrammes montre l'utilisation des différents blocs : le bloc d'affectation associé à la touche [0] à gauche du rectangle d'action, le bloc « pas » associé à la touche [1] qui complémente un signal, ainsi que les blocs d'ancrage de tests et les fonctions « Et » et « Ou ».

Grafcet et Logigramme

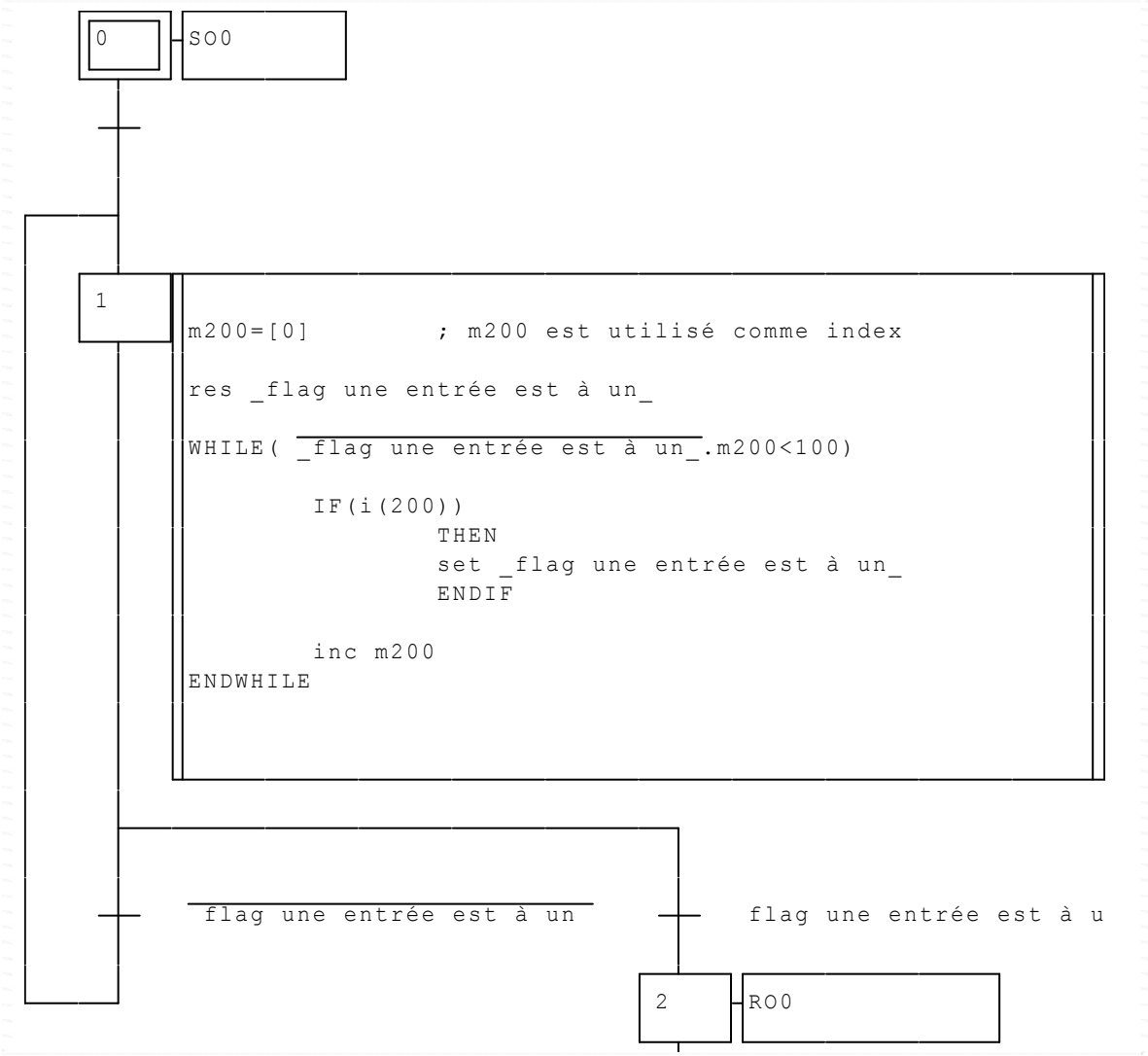


exemples\logigramme\exempl15.agn

Dans cet exemple un Grafcet et un Logigramme sont utilisés conjointement. Le symbole « _étape 1_ » utilisé dans le logigramme est associé à la variable « x1 ».

Ce type de programmation laisse apparaître clairement les conditions d'activation d'une sortie.

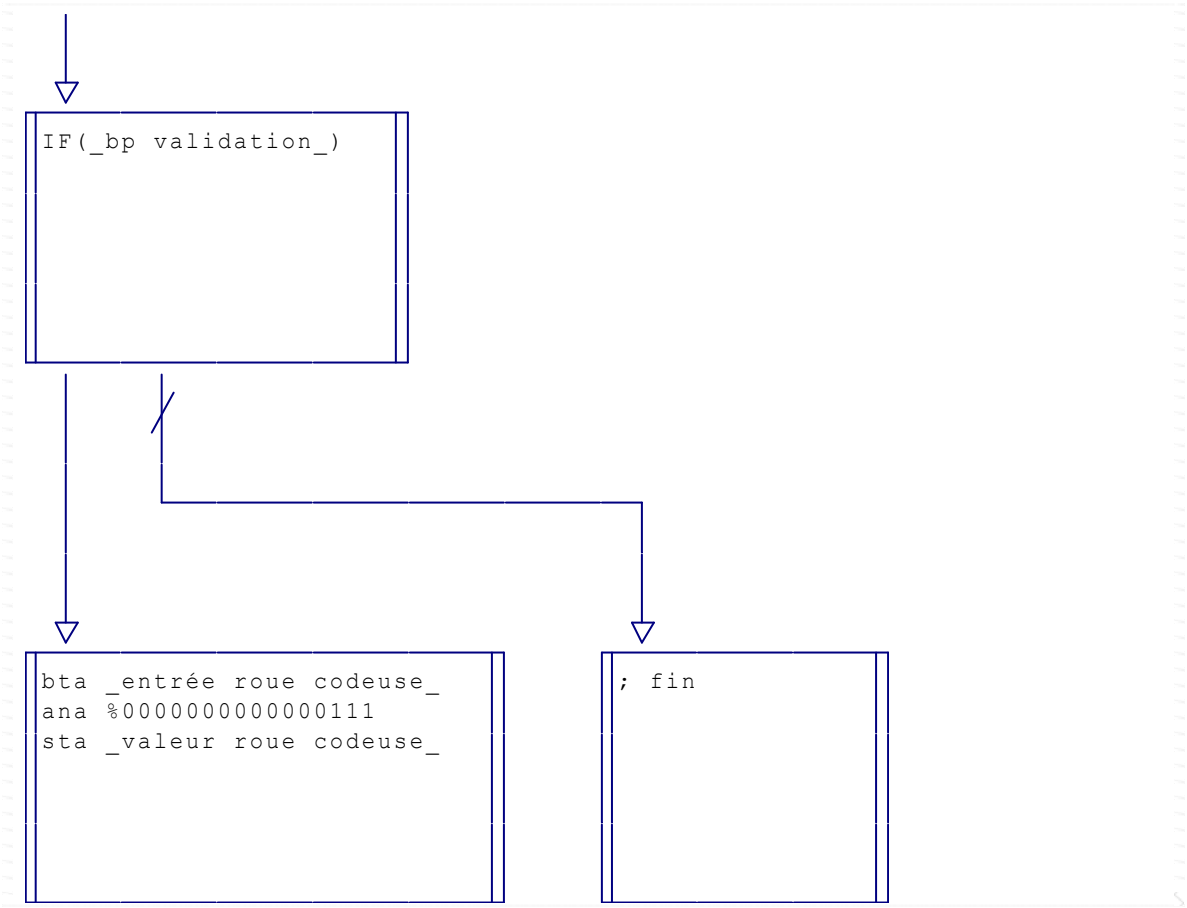
Boîte de langage littéral



exemples\lit\sample16.agn

Ce programme qui associe Grafcet et boîte de langage littéral a pour objet de tester les entrées i0 à i99. Si une de ces entrées est à un, alors l'étape 2 est activée et le Grafcet se retrouve dans un état où toute évolution est interdite. Le symbole « _flag une entrée est à un_ » est associé au bit u500. Un adressage indexé a été utilisé pour balayer les 100 entrées. Notons également l'emploi simultané du langage littéral bas niveau et étendu.

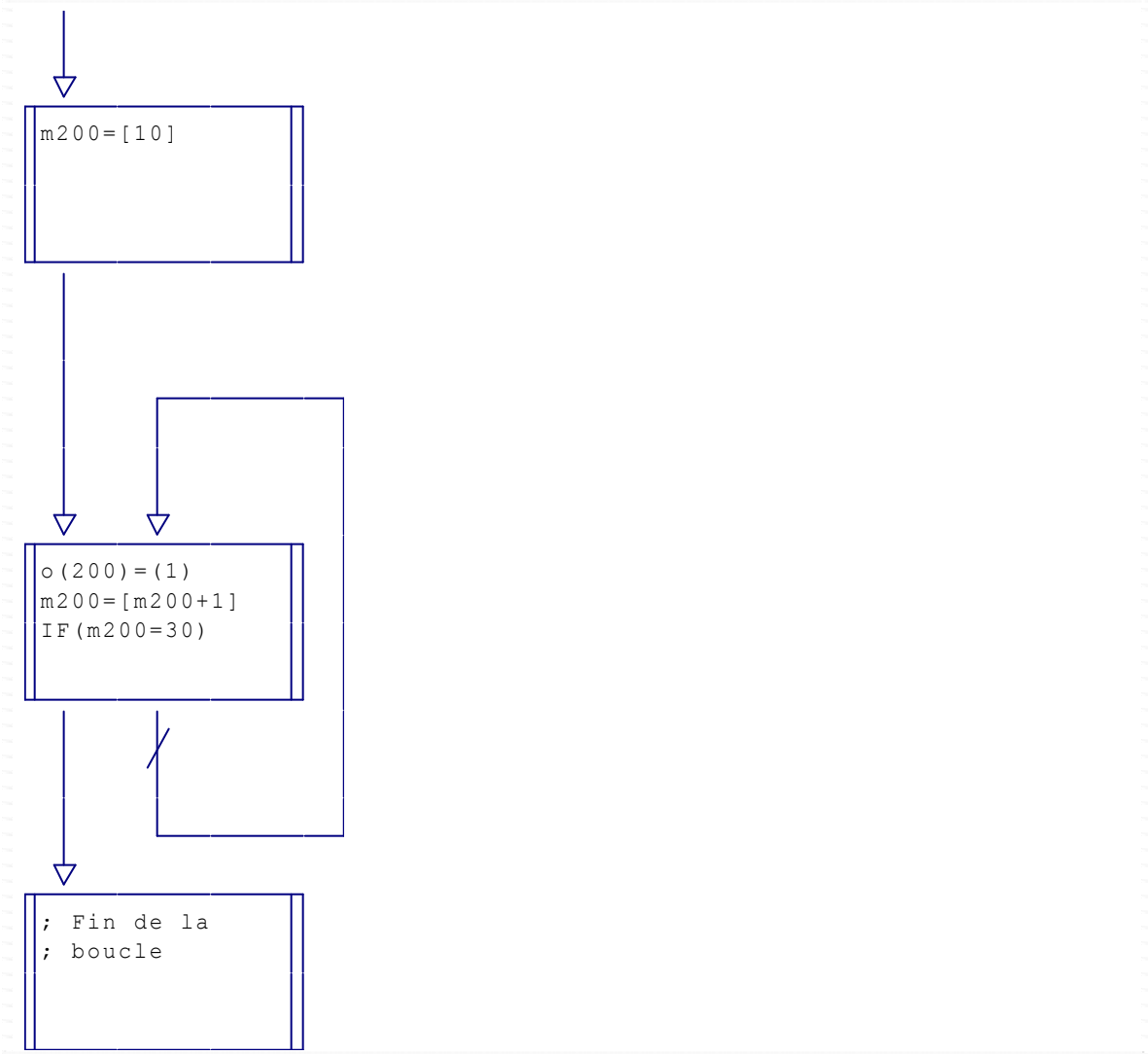
Organigramme



exemples\organigramme\sample18.agn

Cet exemple illustre l'utilisation d'un organigramme pour effectuer un traitement algorithmique et numérique. Ici trois entrées provenant d'une roue codeuse sont lues et stockées dans un mot si une entrée de validation est active.

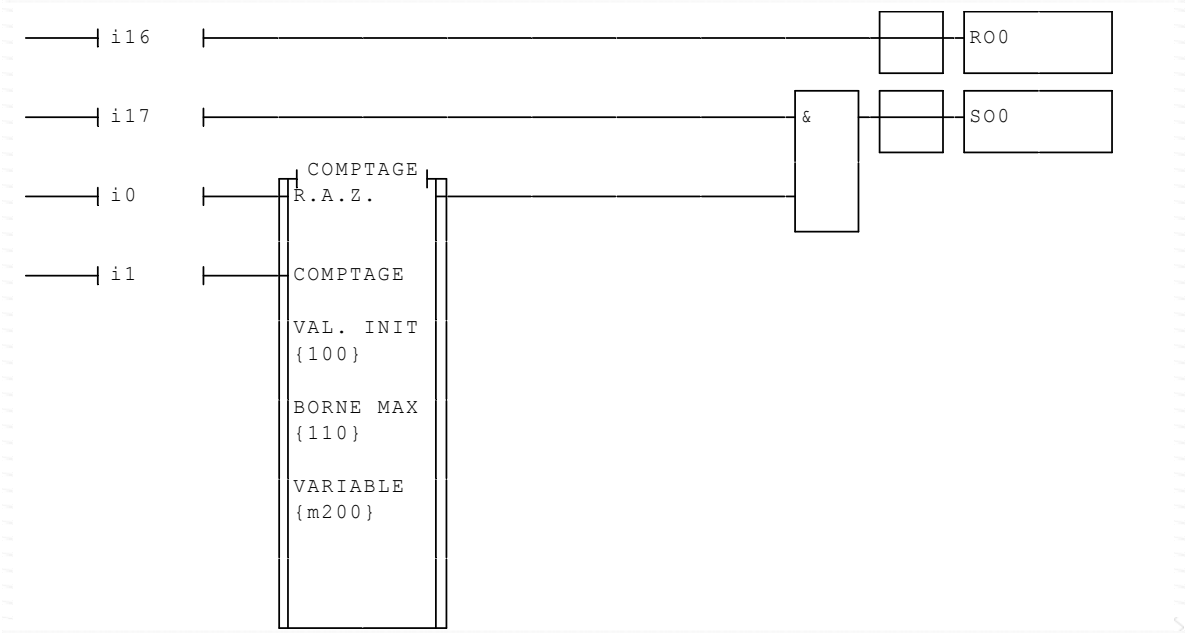
Organigramme



exemples\organigramme\sample19.agn

Ce deuxième exemple d'organigramme réalise une structure de boucle qui permet de forcer à un une série de sorties (o10 à o29) avec un adressage indirect (« o(200) »).

Bloc fonctionnel



exemples\bf\sample20.agn

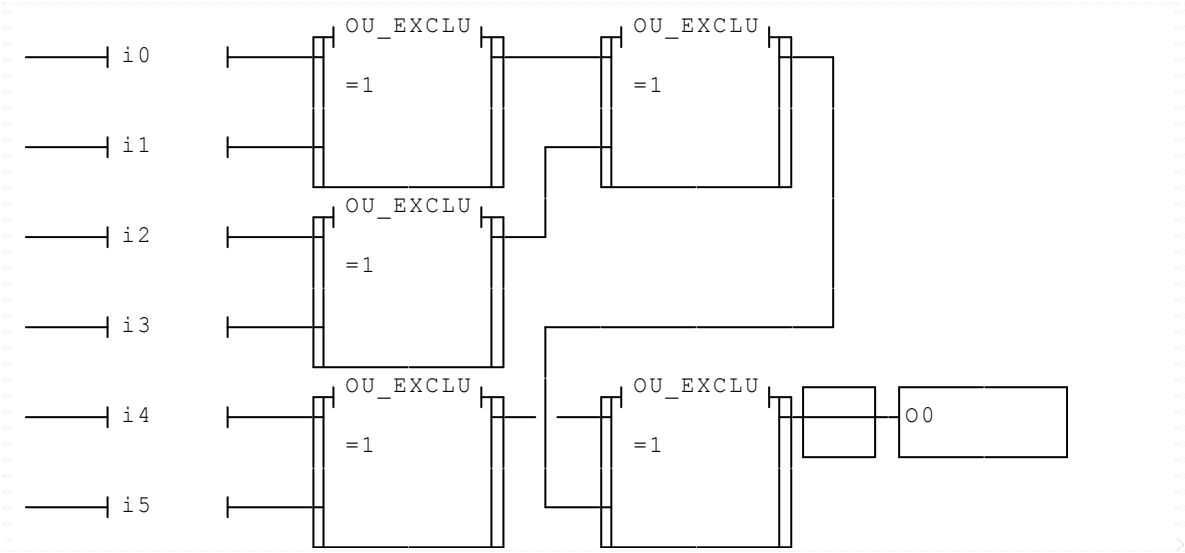
```
; Gestion de l'entrée de RAZ
IF({I0})
        THEN
        {?2}=[{?0}]
        ENDIF

; Gestion de l'entrée de comptage

IF(#{I1})
        THEN
        {?2}=[{?2}+1]
        ENDIF

; Teste la borne maxi


IF({?2}={?1})
        THEN
        {O0}=(1)
        {?2}=[{?0}]
        ENDIF
        ELSE
        {O0}=(0)
        ENDIF
```

comptage.lib (inclus dans les ressources du projet)

Cet exemple illustre l'utilisation d'un bloc fonctionnel. Les fonctions du bloc « COMPTAGE » que nous avons définies ici sont les suivantes :

- ⇨ le comptage se fera en partant d'une valeur d'init et se terminera à une valeur de borne maximale,
- ⇨ lorsque la variable de comptage atteindra la borne maximale elle sera forcée à la valeur d'init et la sortie du bloc passera à un pendant un cycle programme,
- ⇨ le bloc possédera une entrée booléenne de RAZ et une entrée de comptage sur front montant.

Bloc fonctionnel



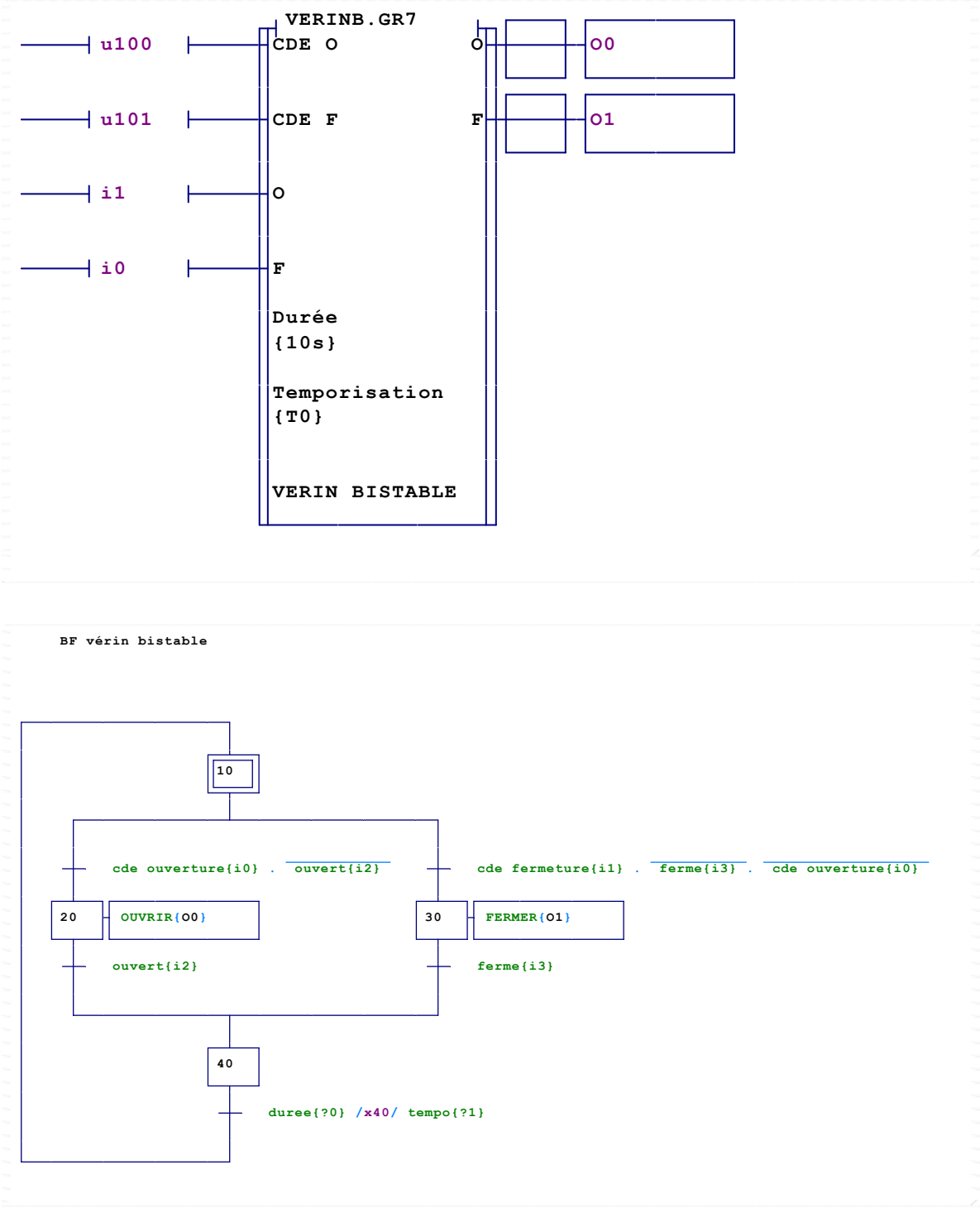
exemples\bf\sample21.agn

```
; Ou exclusif
neq {o0} orr /{i0} eor {i1} orr {i0} eor /{i1}
```

ou_exclu.lib (inclus dans les ressources du projet)

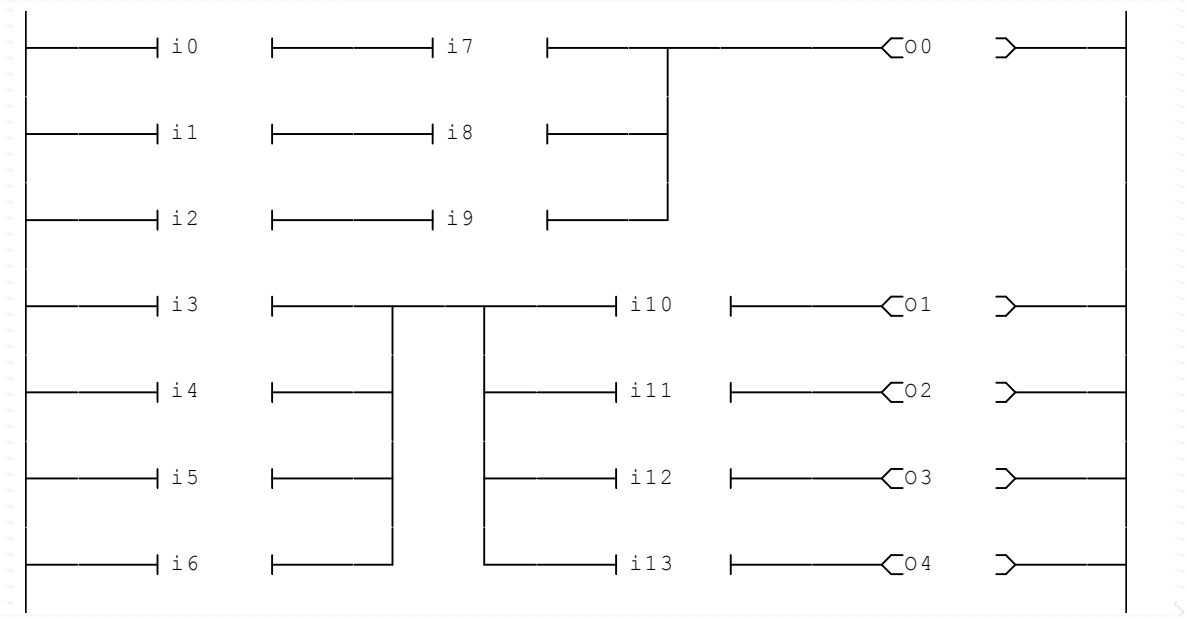
Ce deuxième exemple de bloc fonctionnel illustre l'utilisation multiple d'un même bloc. Le bloc « OU_EXCLU » réalise un ou exclusif entre les deux entrées booléennes. Cet exemple utilise 5 blocs pour réaliser un ou exclusif entre 6 entrées (i0 à i5). Le fichier « OU_EXCLU.LIB » listé dessous régit le fonctionnement du bloc. L'équation booléenne du ou exclusif est la suivante : « (i0./i1)+(/i0.i1) ». La forme équivalente utilisée ici permet de coder l'équation sur une seule ligne de langage littéral bas niveau sans utiliser de variables intermédiaires.

Bloc fonctionnel évolué



exemples\bf\bf géré par un Grafcet.agn

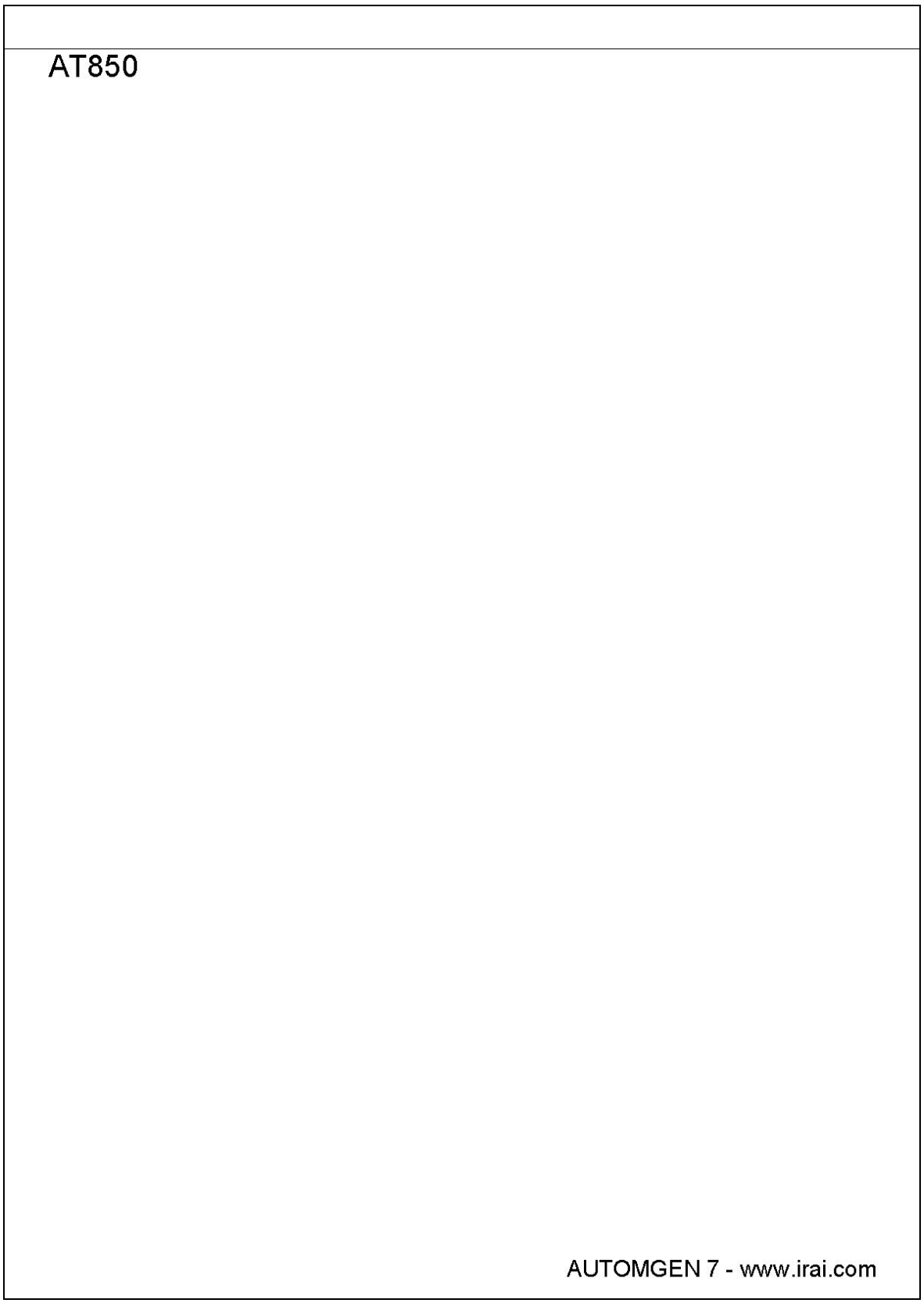
Ladder



exemples\laddersample22.agn

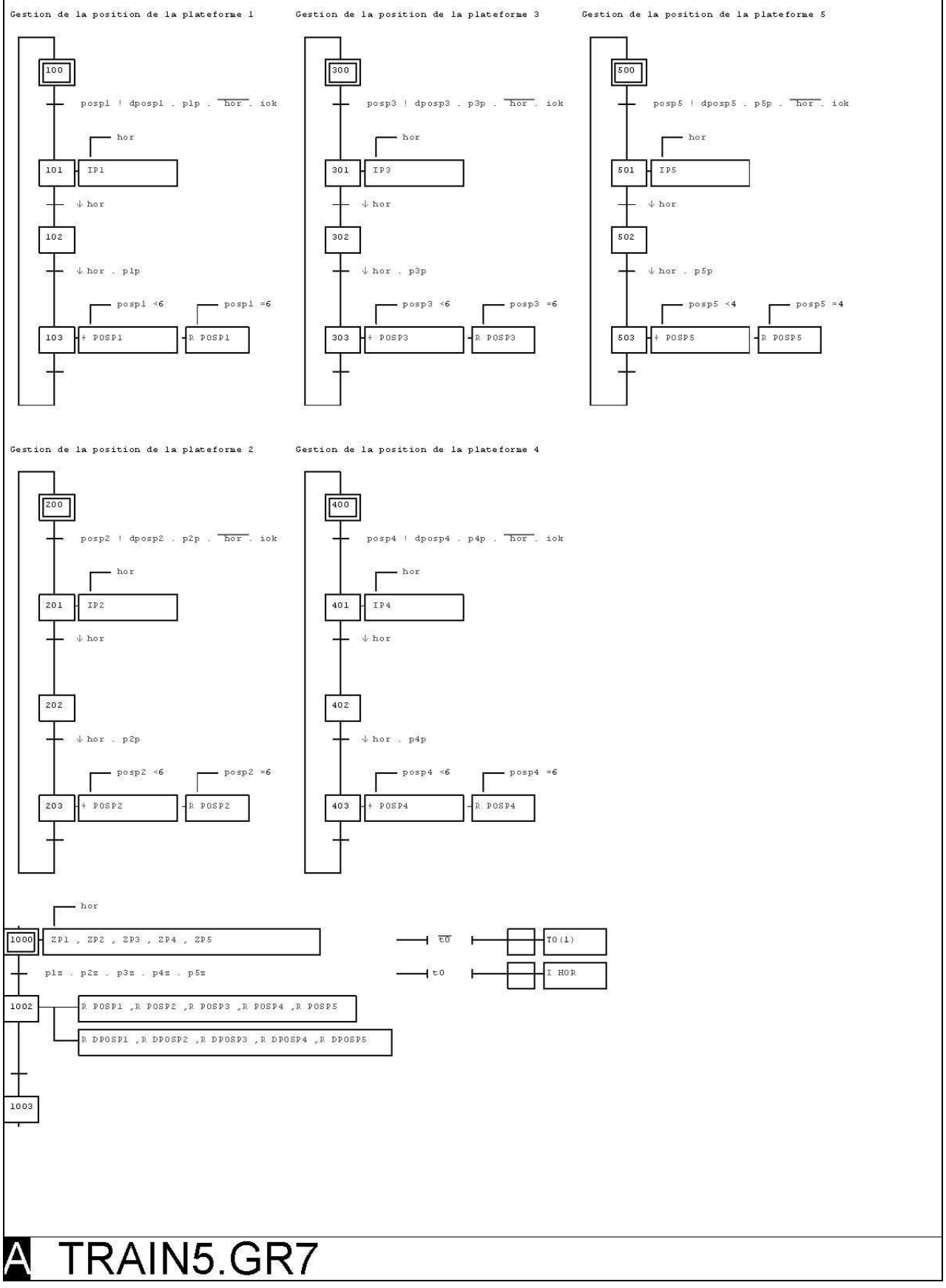
Cet exemple illustre l'utilisation de la programmation en ladder.

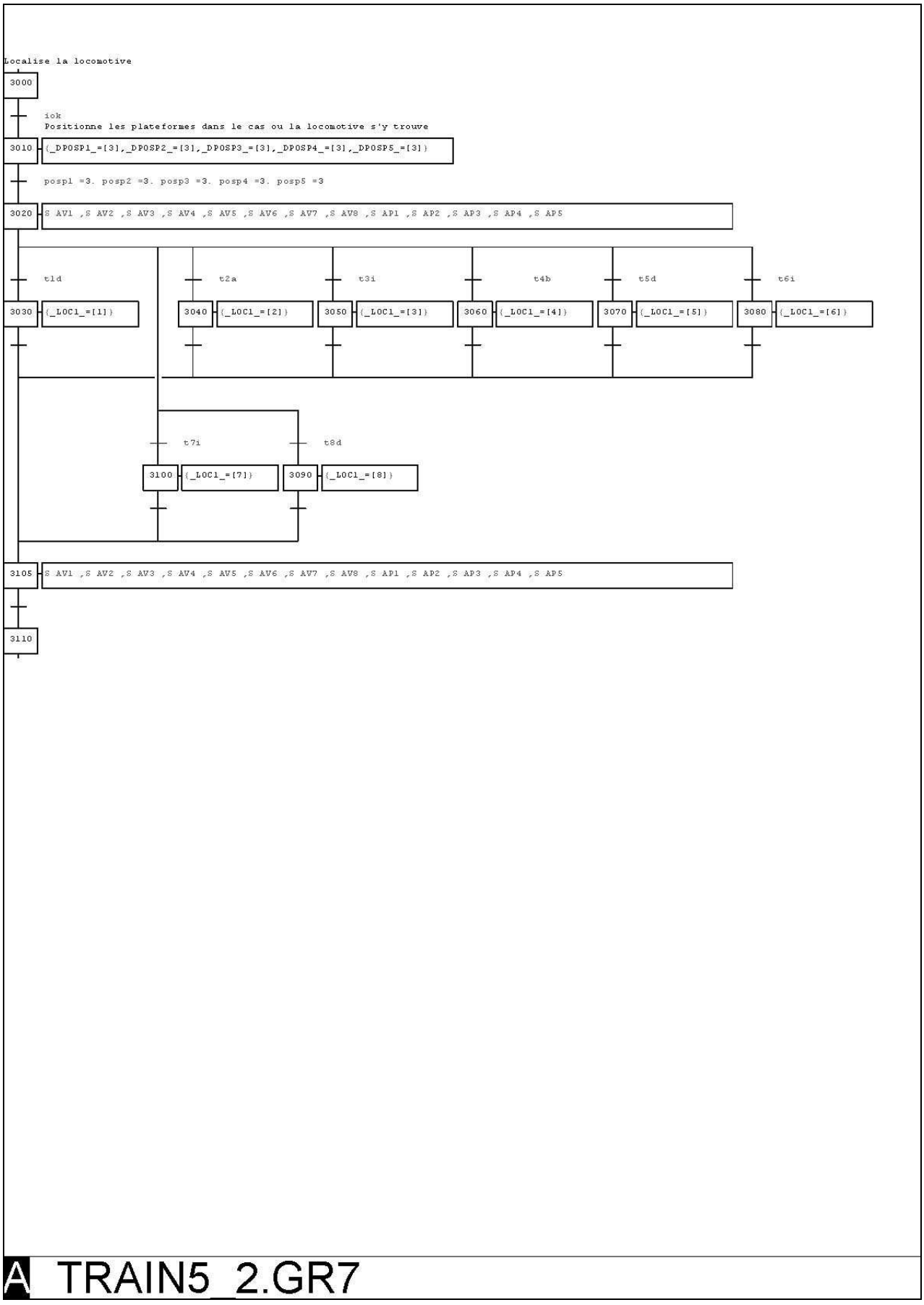
Exemple développé sur une maquette de train

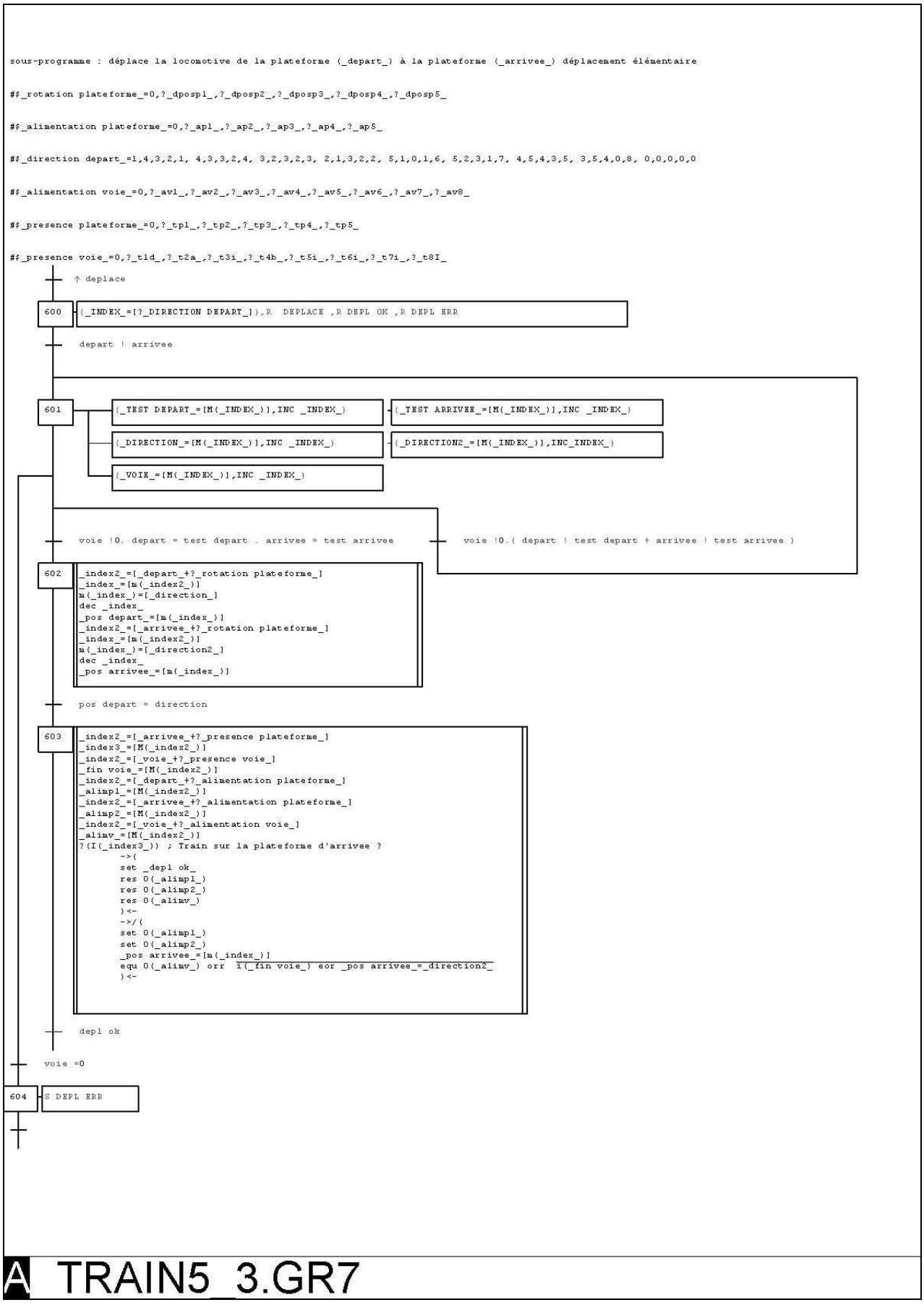


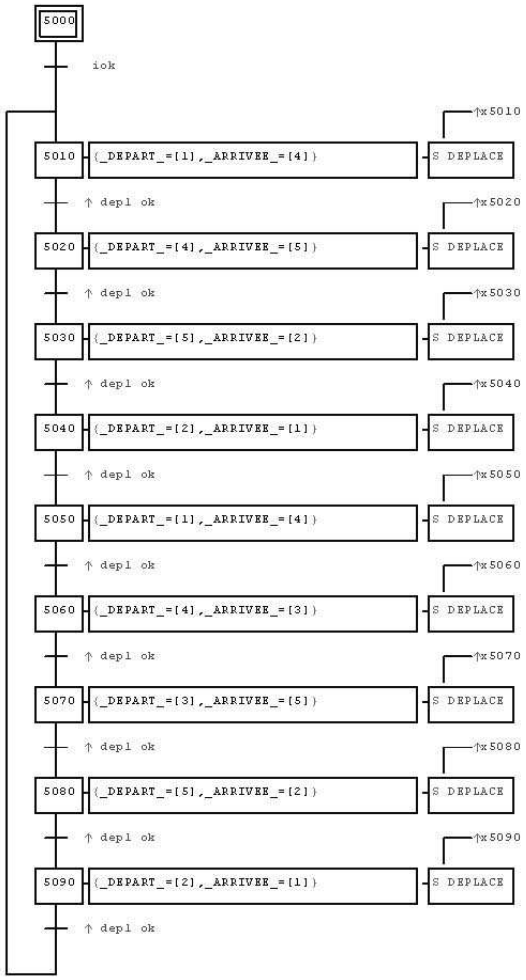
AV1	O0	alimentation voie 1
AV2	O1	alimentation voie 2
AV3	O2	alimentation voie 3
AV4	O3	alimentation voie 4
AV5	O4	alimentation voie 5
AV6	O5	alimentation voie 6
AV7	O6	alimentation voie 7
AV8	O7	alimentation voie 8
AP1	O8	alimentation plateforme 1
AP2	O9	alimentation plateforme 2
AP3	O10	alimentation plateforme 3
AP4	O11	alimentation plateforme 4
AP5	O12	alimentation plateforme 5
IP1	O13	rotation plateforme 1
IP2	O14	rotation plateforme 2
IP3	O15	rotation plateforme 3
IP4	O16	rotation plateforme 4
IP5	O17	rotation plateforme 5
ZP1	O18	initialisation plateforme 1
ZP2	O19	initialisation plateforme 2
ZP3	O20	initialisation plateforme 3
ZP4	O21	initialisation plateforme 4
ZP5	O22	initialisation plateforme 5
DV1	O23	direction voie 1
DV2	O24	direction voie 2
DV3	O25	direction voie 3
DV4	O26	direction voie 4
DV5	O27	direction voie 5
DV6	O28	direction voie 6
DV7	O29	direction voie 7
DV8	O30	direction voie 8
S1D	O31	feu droit voie 1
S1I	O32	feu gauche voie 1
S2A	O33	feu haut voie 2
S2B	O34	feu bas voie 2
S3D	O35	feu droit voie 3
S3I	O36	feu gauche voie 3
S4A	O37	feu haut voie 4
S4B	O38	feu bas voie 4
S5D	O39	feu droit voie 5
S5I	O40	feu gauche voie 5
S6D	O41	feu droit voie 6
S6I	O42	feu gauche voie 6
S7D	O43	feu droit voie 7
S7I	O44	feu gauche voie 7
S8D	O45	feu droit voie 8
S8I	O46	feu gauche voie 8
T1D	i0	train droit voie 1
T1I	i1	train gauche voie 1
T2A	i2	train haut voie 2
T2B	i3	train bas voie 2
T3D	i4	train droit voie 3
T3I	i5	train gauche voie 3
T4A	i6	train haut voie 4
T4B	i7	train bas voie 4
T5D	i8	train droit voie 5

1/1 Symboles





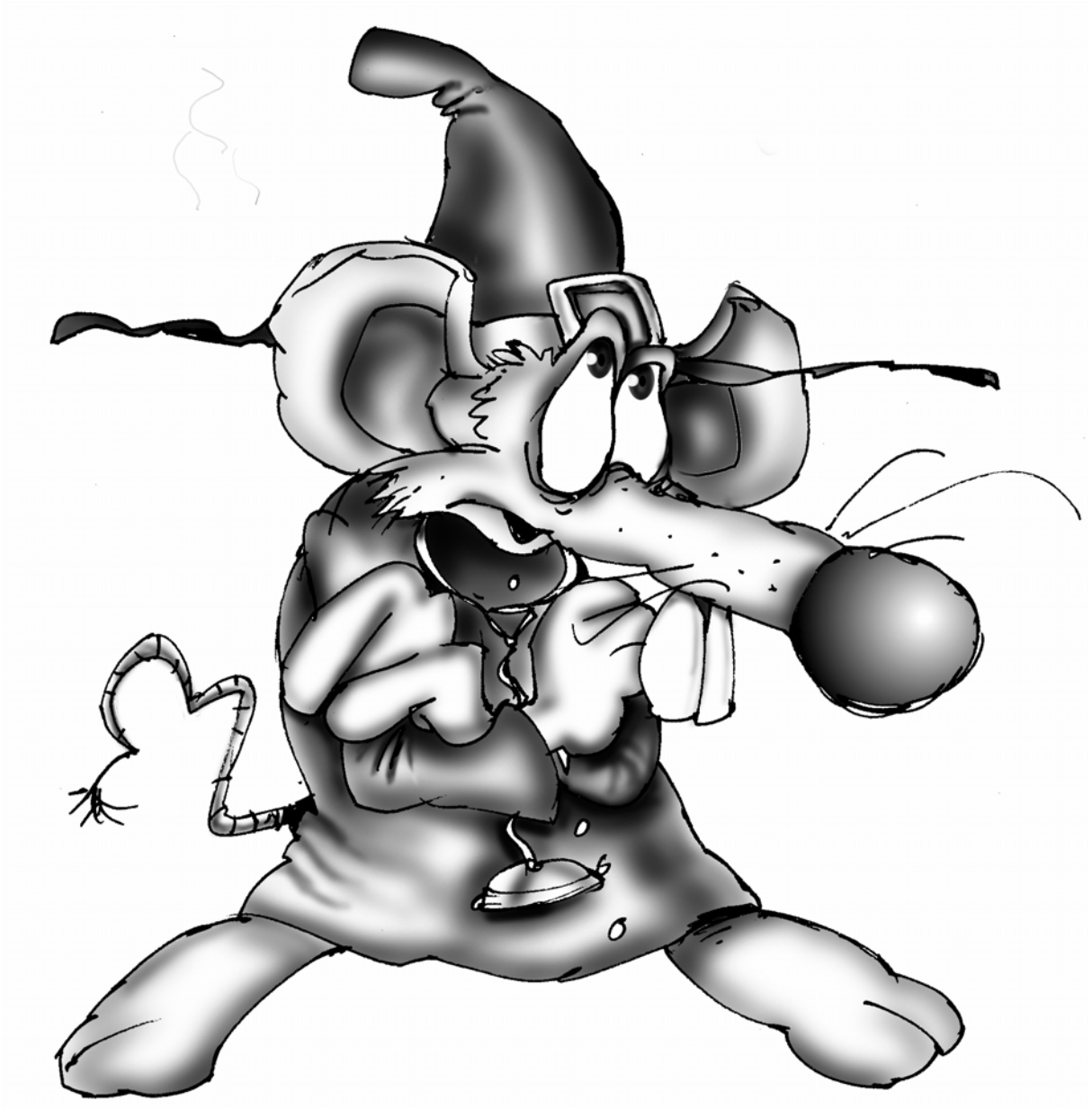

sous-programme : déplace la locomotive de la plateforme (_depart_) à la plateforme (_arrivee_) déplacement élémentaire
#$_rotation plateforme_=0,?_dposp1_,?_dposp2_,?_dposp3_,?_dposp4_,?_dposp5_
#$_alimentation plateforme_=0,?_ap1_,?_ap2_,?_ap3_,?_ap4_,?_ap5_
#$_direction depart_=1,4,3,2,1, 4,3,3,2,4, 3,2,3,2,3, 2,1,3,2,2, 5,1,0,1,6, 5,2,3,1,7, 4,5,4,3,5, 3,5,4,0,8, 0,0,0,0,0
#$_alimentation voie_=0,?_av1_,?_av2_,?_av3_,?_av4_,?_av5_,?_av6_,?_av7_,?_av8_
#$_presence plateforme_=0,?_tp1_,?_tp2_,?_tp3_,?_tp4_,?_tp5_
#$_presence voie_=0,?_t1d_,?_t2a_,?_t3i_,?_t4b_,?_t5i_,?_t6i_,?_t7i_,?_t8I_
↑ deplace
600
(_INDEX_=[?_DIRECTION DEPART_]),R DEPLACE ,R DEPL OK ,R DEPL ERR
depart ! arrivee
601
(_TEST DEPART_=[M(_INDEX_)],INC _INDEX_)
(_TEST ARRIVEE_=[M(_INDEX_)],INC _INDEX_)
(_DIRECTION_=[M(_INDEX_)],INC _INDEX_)
(_DIRECTION2_=[M(_INDEX_)],INC_INDEX_)
(_VOIE_=[M(_INDEX_)],INC _INDEX_)
voie !0. depart = test depart . arrivee = test arrivee
voie !0.(depart ! test depart + arrivee ! test arrivee)
602
index2=[_depart_+?_rotation plateforme_]
index=[m(_index2_)]
m(_index_)=[_direction_]
dec _index_
pos depart=[m(_index_)]
index2=[_arrivee_+?_rotation plateforme_]
index=[m(_index2_)]
m(_index_)=[_direction2_]
dec _index_
pos arrivee=[m(_index_)]
pos depart = direction
603
index2=[_arrivee_+?_presence plateforme_]
index3=[M(_index2_)]
index2=[_voie_+?_presence voie_]
fin voie=[M(_index2_)]
index2=[_depart_+?_alimentation plateforme_]
alimp1=[M(_index2_)]
index2=[_arrivee_+?_alimentation plateforme_]
alimp2=[M(_index2_)]
index2=[_voie_+?_alimentation voie_]
alimv=[M(_index2_)]
?(I(_index3_)) ; Train sur la plateforme d'arrivee ?
->(
set _depl ok_
res 0(_alimp1_)
res 0(_alimp2_)
res 0(_alimv_)
)<-
->/(
set 0(_alimp1_)
set 0(_alimp2_)
pos arrivee=[m(_index_)]
equ 0(_alimv_) orr I(_fin voie_) eor _pos arrivee_=_direction2_
)<-
depl ok
voie =0
604
S DEPL ERR
A TRAIN5_3.GR7



A TRAIN5_4.GR7

Les aventures de Docteur R.

Manuel pédagogique à l'usage des utilisateurs d'AUTOMGEN



Dessins par TABAN

Distribution

Docteur R. …… Monsieur R.

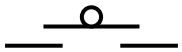
Docteur R. au royaume de la domotique

Nous allons aborder différents exemples pouvant directement s'appliquer dans un projet de domotique. D'un premier abord simple, ces exemples nous permettront d'appréhender différents aspects de la base des automatismes et de l'apprentissage d'AUTOMGEN et d'IRIS.

Ce symbole évoquera dans ce qui suit une partie commande (un automate programmable par exemple.

Est-il besoin de préciser que ceci

évoquera une ampoule,

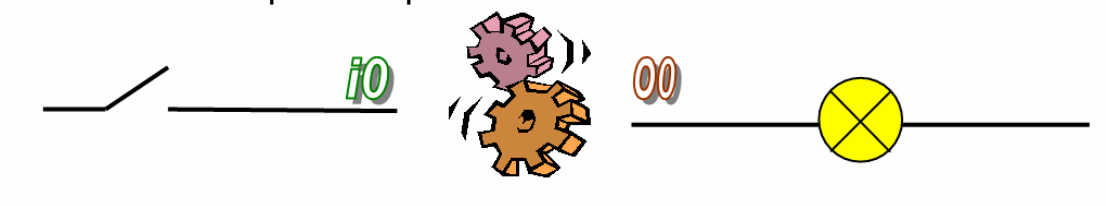


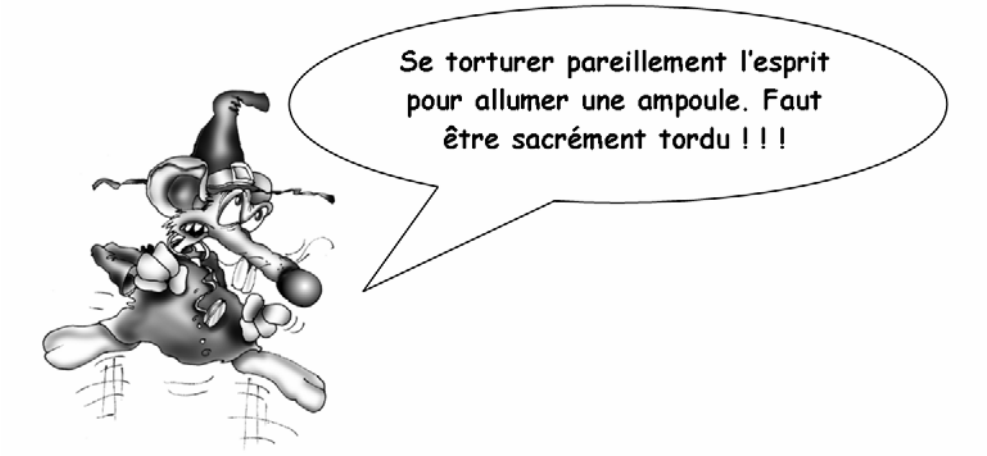
un bouton poussoir

et cela un interrupteur ?

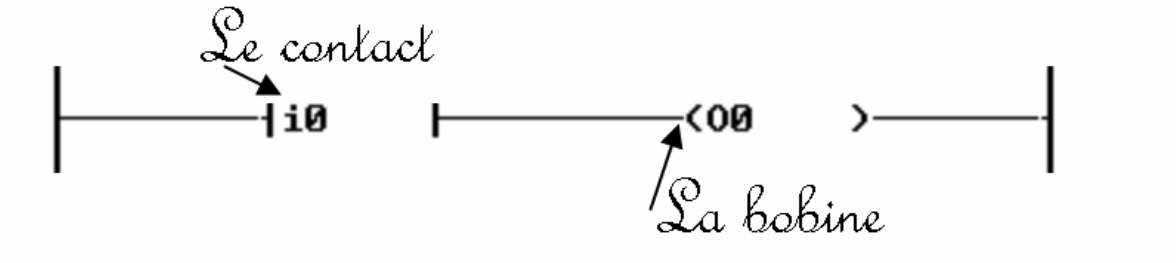
Premier exemple : « qui de l'interrupteur ou de l'ampoule était le premier … »

Un simple interrupteur et une simple ampoule : l'interrupteur est câblé sur l'entrée i0, l'ampoule sur la sortie o0. Si l'interrupteur est fermé alors l'ampoule s'allume, si l'interrupteur est ouvert l'ampoule s'éteint. Difficile de faire plus simple.



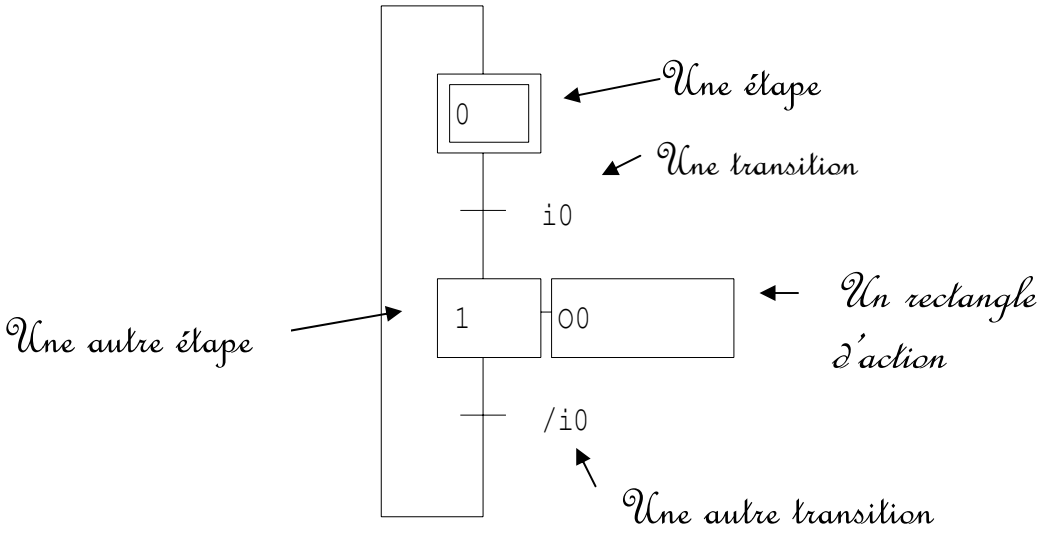


Solution 1 : le langage naturel de l'électricien : le ladder



Le ladder est la transcription la plus directe du schéma électrique. Le contact reçoit le nom de l'entrée où est câblé l'interrupteur, la bobine le nom de la sortie où est câblée l'ampoule.

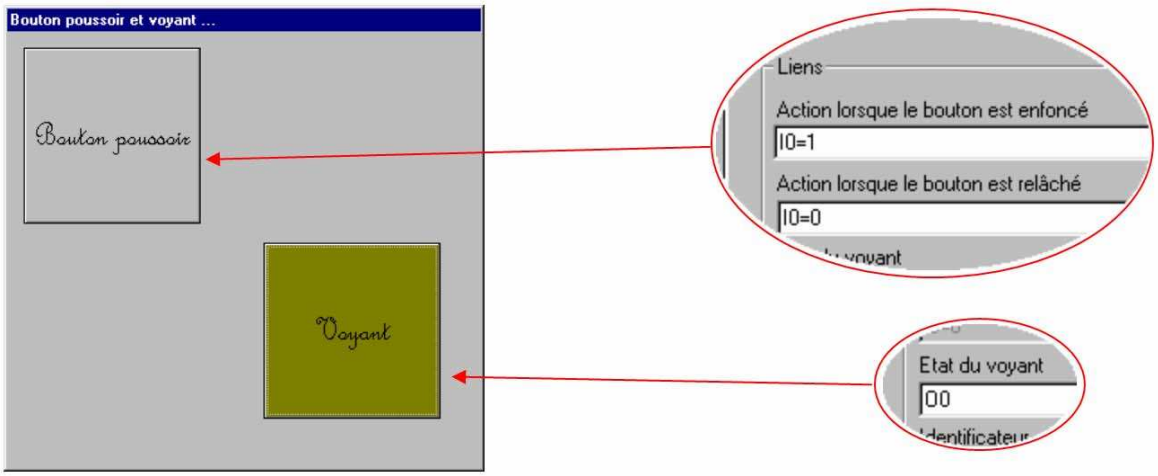
Solution 2 : le langage séquentiel de l'automaticien : le Grafcet

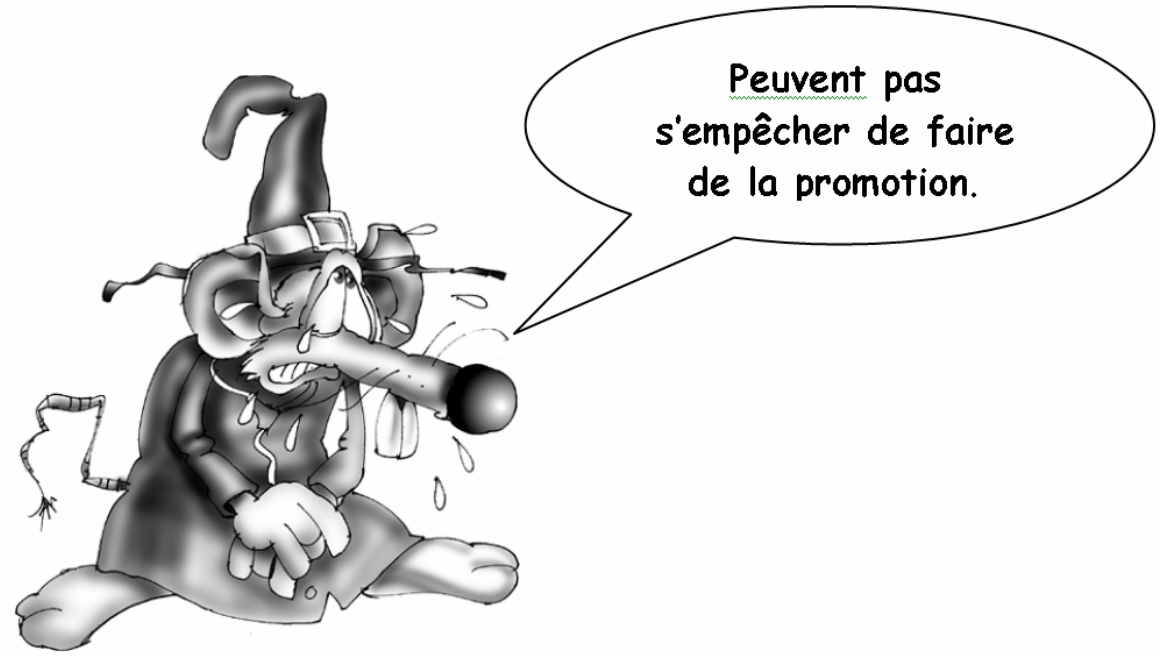


Le Grafcet est basé sur la notion d'état. Pour notre problème nous pouvons dire qu'il y a deux états : l'état allumé et l'état éteint. Chaque étape représente un état : ici l'étape 0 représente l'état éteint et l'étape 1 l'état allumé. Reste à déterminer la condition qui fait évoluer de l'état éteint à l'état allumé : ici l'interrupteur fermé (noté i0) puis la condition qui fait passer de l'état allumé à l'état éteint : ici l'interrupteur ouvert (noté / i0). Les conditions sont écrites à droite de l'élément noté transition. Le rectangle associé à l'étape 1 nommé rectangle d'action contient le nom de la sortie O0 (sortie où est câblée notre ampoule).

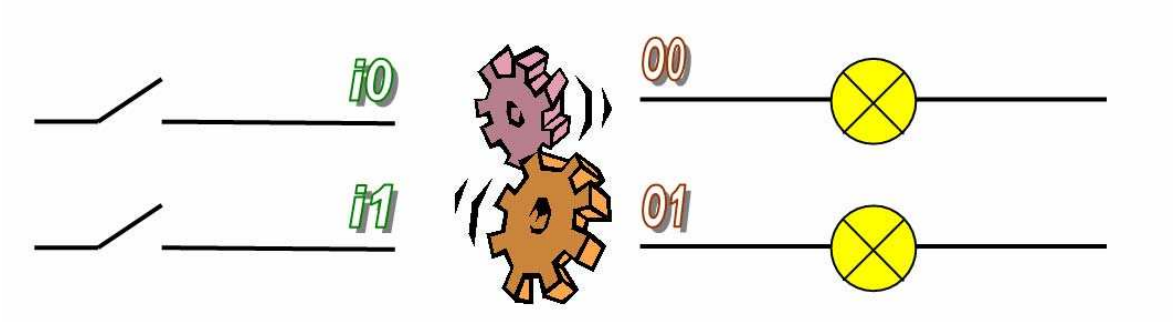
Ainsi, à tout instant l'état de l'ampoule est le même que celui de l'étape 1.

Les heureux possesseurs d'IRIS peuvent utiliser deux objets BPVOYANT pour simuler ce premier exemple.



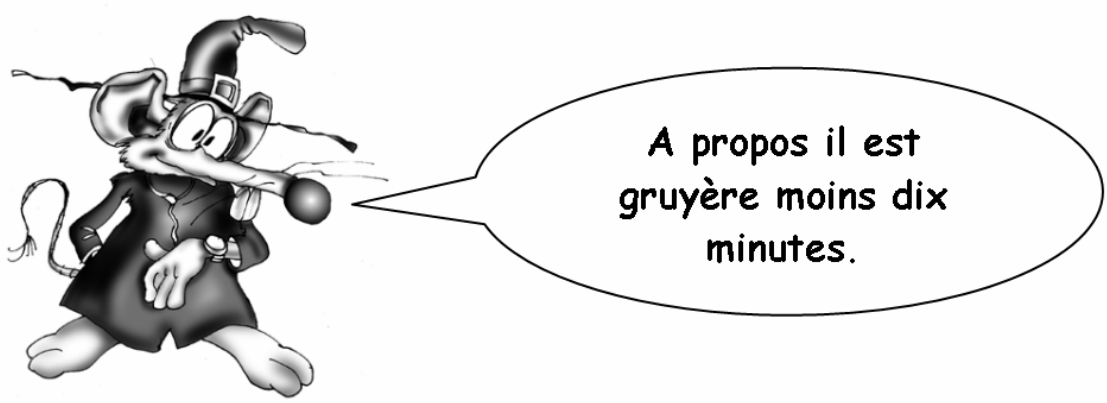


A vous de jouer …



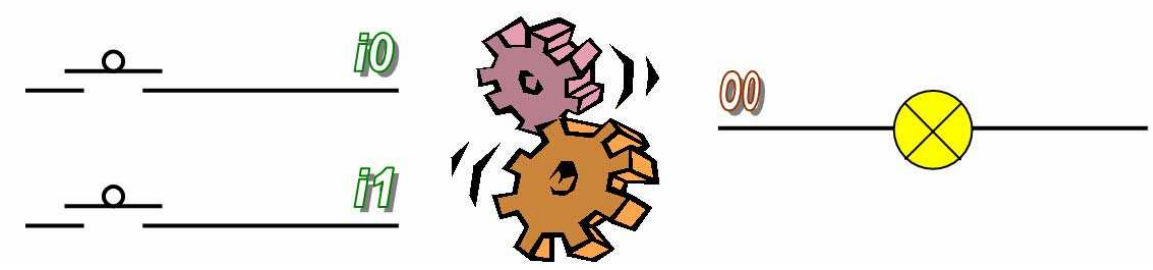
L'interrupteur 1 allume l'ampoule 1, l'interrupteur 2 l'ampoule numéro 2. Une solution Grafcet vous est proposée à la fin de ce document.

Deuxième exemple : « temporisations, minuteries et autres amusements temporels… »

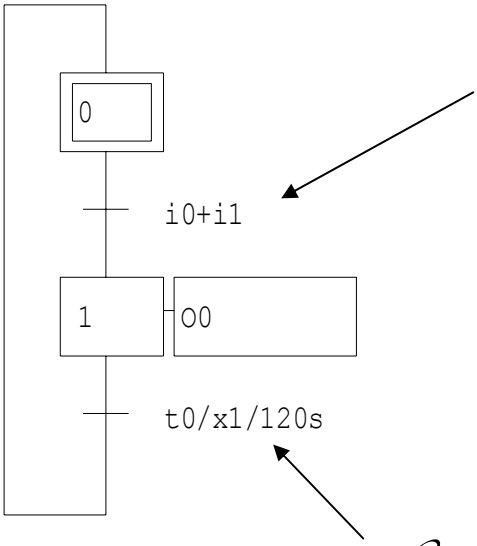


Comme vous vous en doutez certainement, la notion de temporisation est utilisée lorsqu'un programme doit, d'une façon ou d'une autre, effectuer des actions en tenant compte d'une donnée relative au temps. Attendre un certain temps avant de faire une action ou faire une action pendant un certain temps par exemple.

Notre deuxième exemple est le suivant : un couloir est équipé d'une ampoule et de deux boutons poussoirs. L'appui sur un des deux boutons poussoirs provoque l'allumage de l'ampoule pendant 2 minutes (d'après Dr R. c'est largement suffisant pour traverser le couloir).



Solution 1 : la simplicité

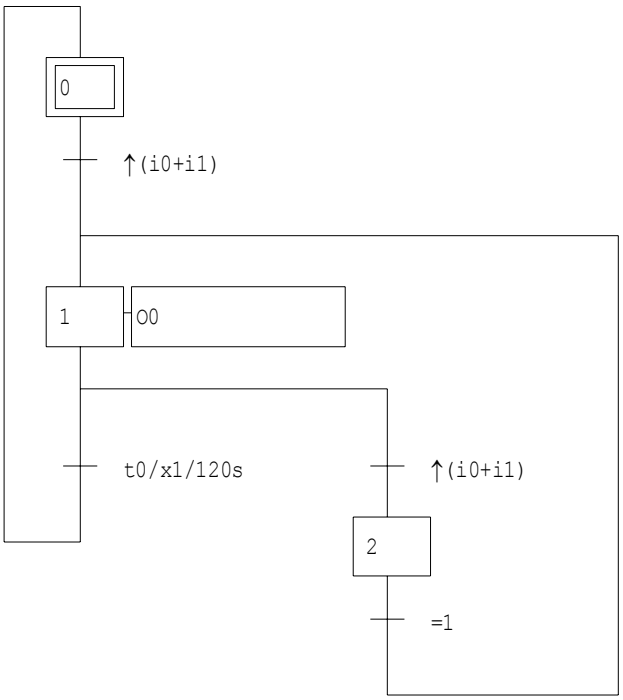


Allumer l'ampoule si le bouton poussoir 1 est pressé ***ou*** *si le bouton poussoir 2 est pressé. « Ou » s'écrit « + » dans une transition.*

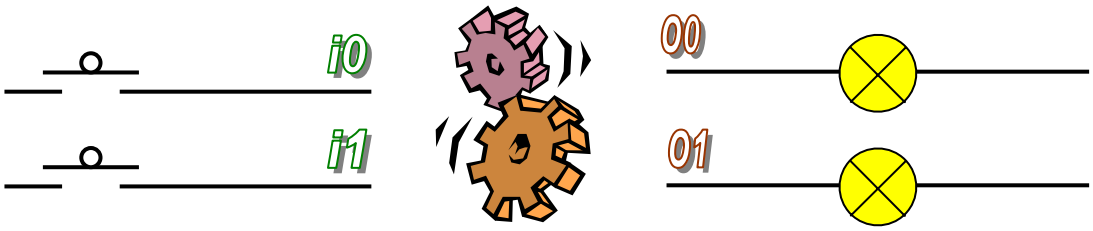
Attendre 2 minutes (120 secondes) en utilisant la temporisation 0, c'est l'étape 1 qui lance la temporisation.

Solution 2 : amélioration

Un problème posé par cette solution est que si l'on appuie sur un bouton poussoir pendant que l'ampoule est allumée, alors on ne réarme pas la temporisation. Ainsi Dr R. croyant avoir réarmé la minuterie s'est retrouvé dans le noir la semaine dernière.



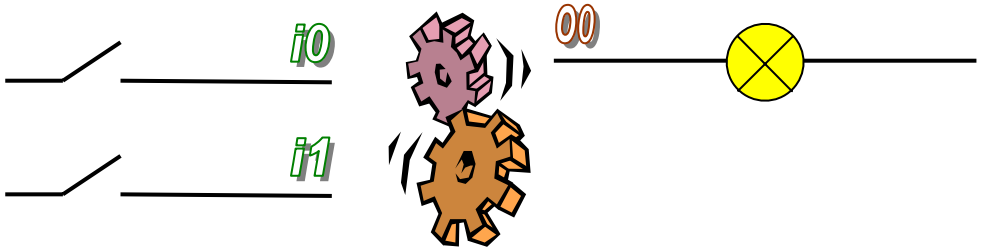
Et si vous vous lanciez dans l'écriture d'un vrai programme …



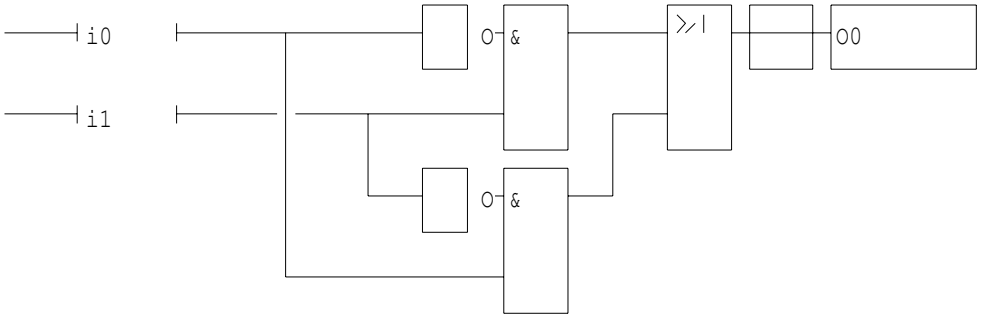
Une gestion intelligente de l'éclairage du couloir : une ampoule a été placée à chaque extrémité. Lorsqu'on appuie sur un interrupteur : les deux ampoules s'allument, puis l'ampoule se trouvant du côté de l'interrupteur pressé s'éteint au bout de 30 secondes et enfin l'autre ampoule au bout d'une minute.

Troisième exemple : « variation sur le thème du va et vient… »

Rappelons le principe au combien génial du va et vient : deux interrupteurs permettent d'allumer ou d'éteindre la même ampoule.

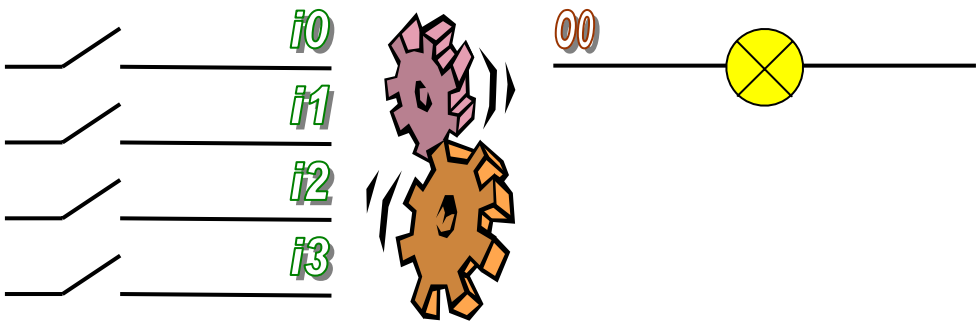


Voici une solution en logigramme :

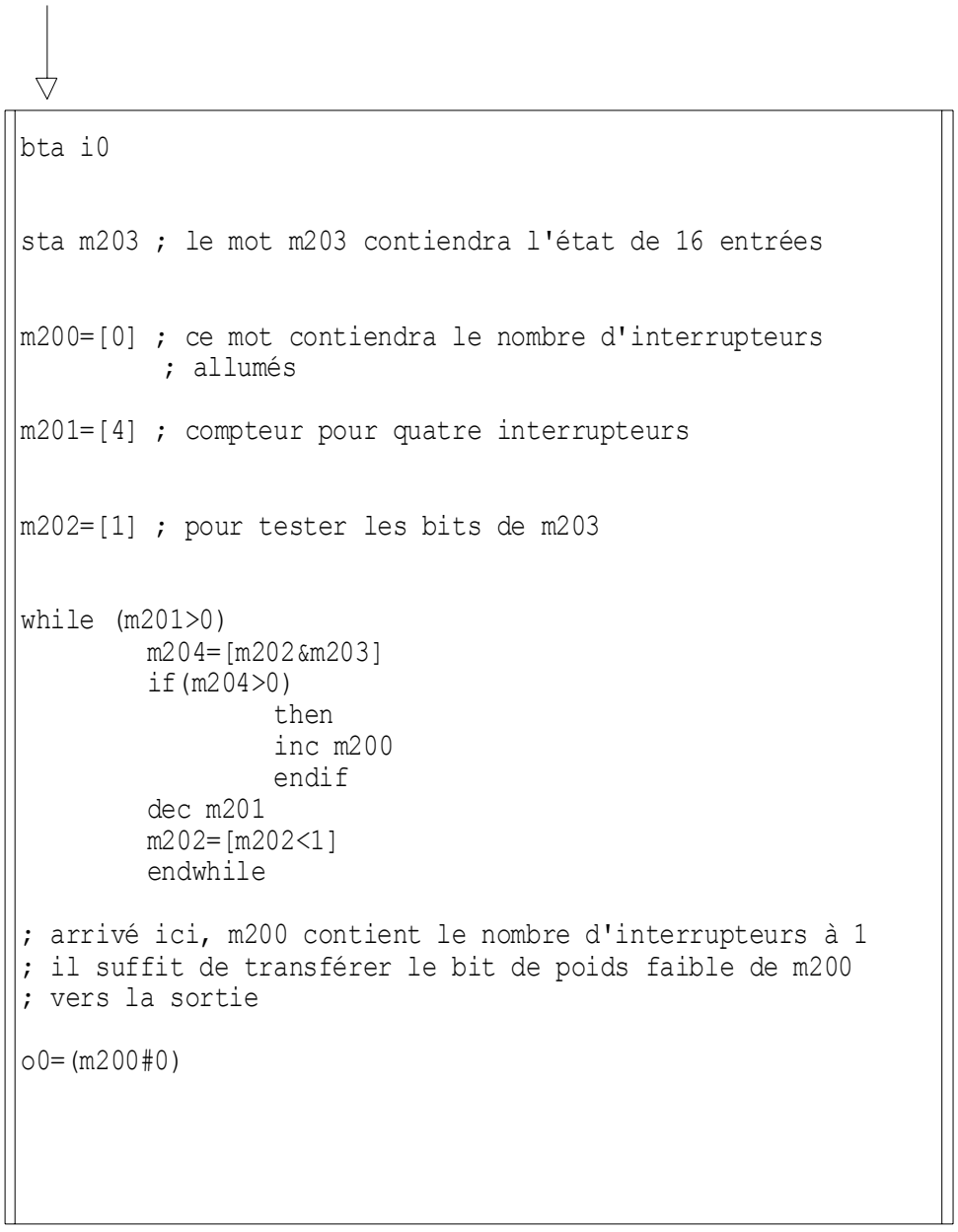


Les puristes auront reconnu l'équation booléenne du ou exclusif.

Les choses deviennent réellement intéressantes si l'on souhaite conserver les propriétés du va et vient avec un nombre d'interrupteurs supérieur à 2.

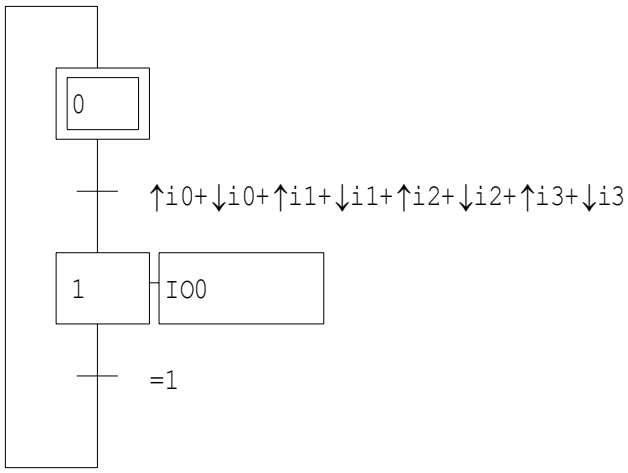


Une solution utilisant le langage littéral d'AUTOMGEN.

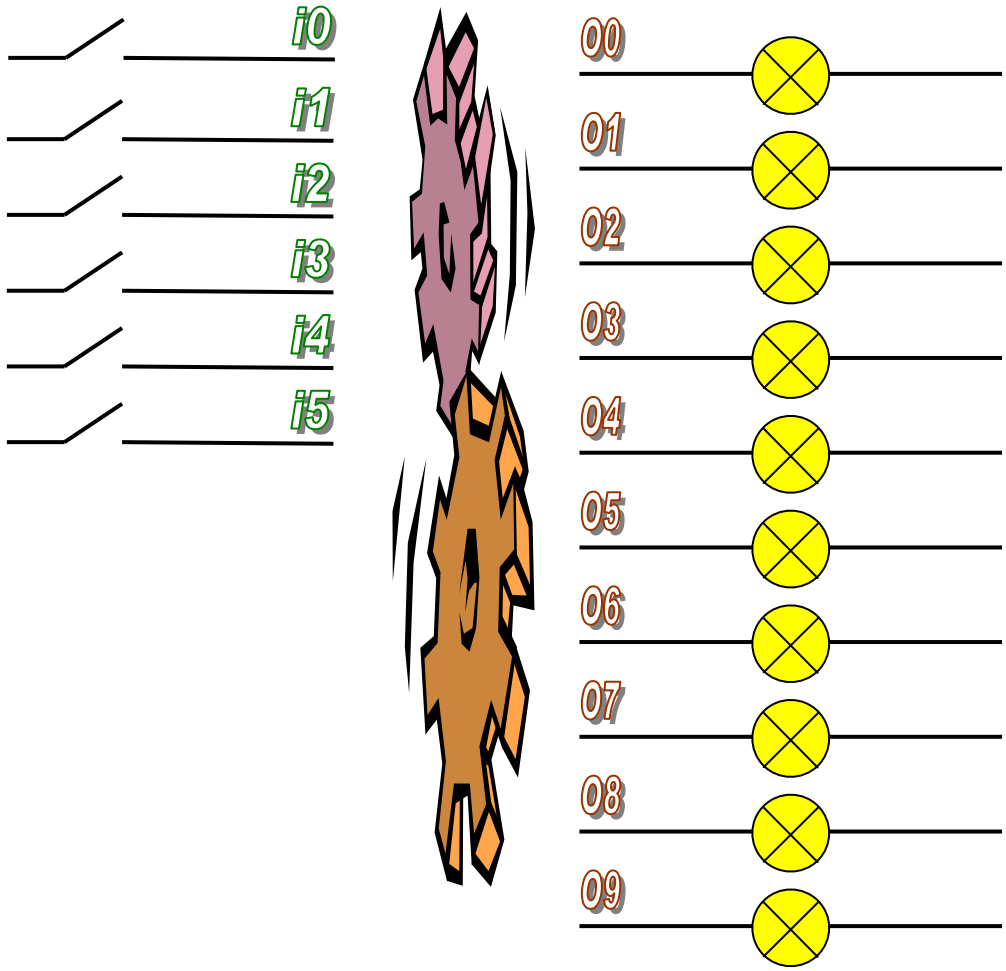


```
bta i0


sta m203 ; le mot m203 contiendra l'état de 16 entrées


m200=[0] ; ce mot contiendra le nombre d'interrupteurs
          ; allumés

m201=[4] ; compteur pour quatre interrupteurs


m202=[1] ; pour tester les bits de m203


while (m201>0)
         m204=[m202&m203]
         if(m204>0)
                  then
                  inc m200
                  endif
         dec m201
         m202=[m202<1]
         endwhile

; arrivé ici, m200 contient le nombre d'interrupteurs à 1
; il suffit de transférer le bit de poids faible de m200
; vers la sortie

o0=(m200#0)
```


Une autre plus astucieuse :



« IO0 » signifie « inverser l'état de la sortie 0 ».
Essayez ceci :
Une grande pièce avec 6 interrupteurs et 10 ampoules. Chaque interrupteur permet d'éclairer plus ou moins la pièce (en clair de passer d'un état où tout est éteint à un état où une ampoule est allumée, puis deux, etc …).

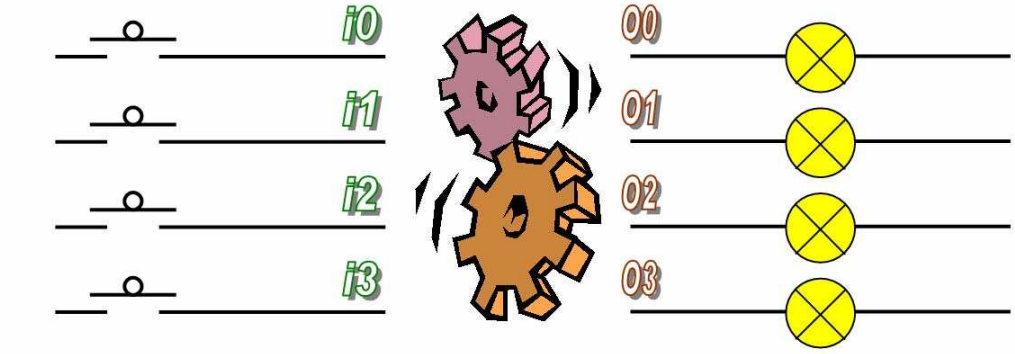


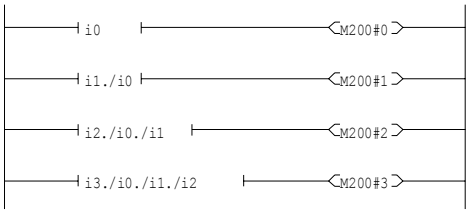
Quatrième exemple : « Et le bouton poussoir devint intelligent … »

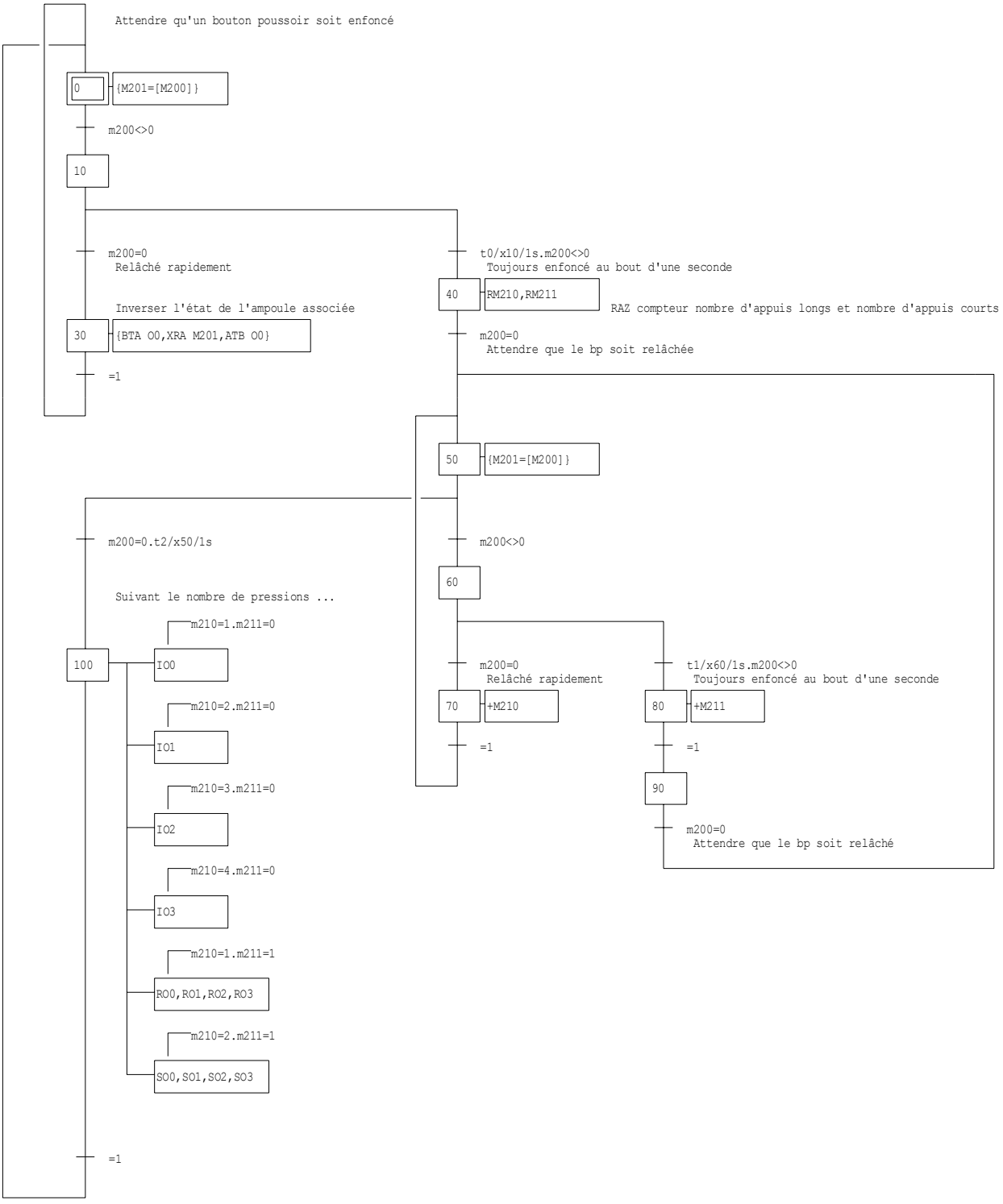
Dans tous les exemples précédents, les boutons poussoirs n'ont réalisés qu'une seule fonction. Entendez par là que la personne qui les manipule n'a que deux choix : ne pas appuyer dessus ou appuyer dessus pour obtenir une fonction (allumer ou éteindre). Imaginons un bouton poussoir « plus performant » capable de recevoir deux types de pression : une pression courte (arbitrairement moins de 1 seconde) ou une pression longue (arbitrairement au moins une seconde).

Pour cet exemple quatre boutons poussoirs et quatre ampoules. Par défaut, en utilisation normale chacun des boutons poussoirs est associé à une ampoule. Une pression courte sur un bouton poussoir allume ou éteint l'ampoule associée. Chaque bouton poussoir doit permettre de piloter chaque ampoule ou la totalité des ampoules. Le tableau ci-dessous résume le fonctionnement.

Type d'action sur les boutons poussoirs	**Résultat**
Une pression courte	L'ampoule associée change d'état
Une pression longue et une pression courte	L'ampoule numéro 1 change d'état
Une pression longue et deux pressions courtes	L'ampoule numéro 2 change d'état
Une pression longue et trois pressions courtes	L'ampoule numéro 3 change d'état
Une pression longue et quatre pressions courtes	L'ampoule numéro 4 change d'état
Deux pressions longues et une pression courte	Toutes les ampoules s'éteignent
Deux pressions longues et deux pressions courtes	Toutes les ampoules s'allument



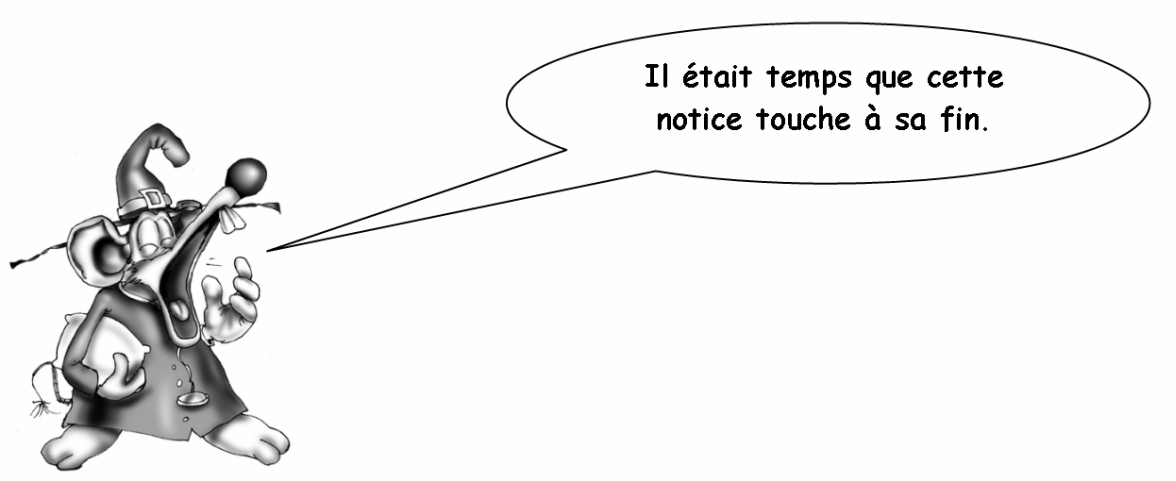




Ceci termine ce manuel pédagogique. Nous espérons qu'il vous aura permis de découvrir les possibilités d'AUTOMGEN.

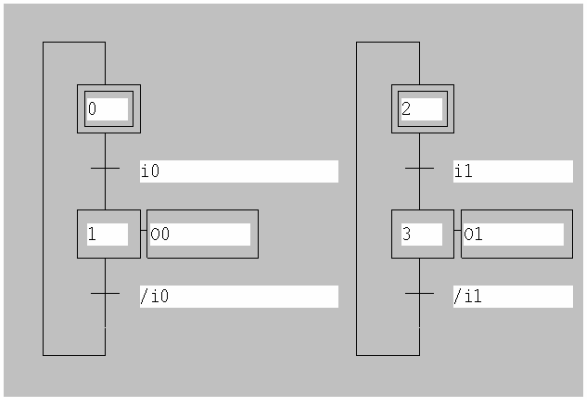
Nous vous proposons un ultime exercice.

Automatiser l'appartement de votre tante Hortense en respectant son goût immodéré pour les interrupteurs nickelés.



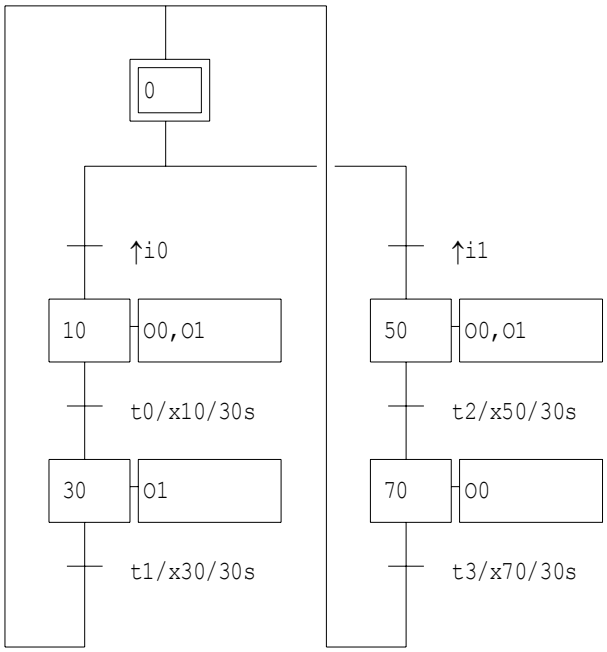
Les solutions …

« qui de l'interrupteur ou de l'ampoule était le premier … »

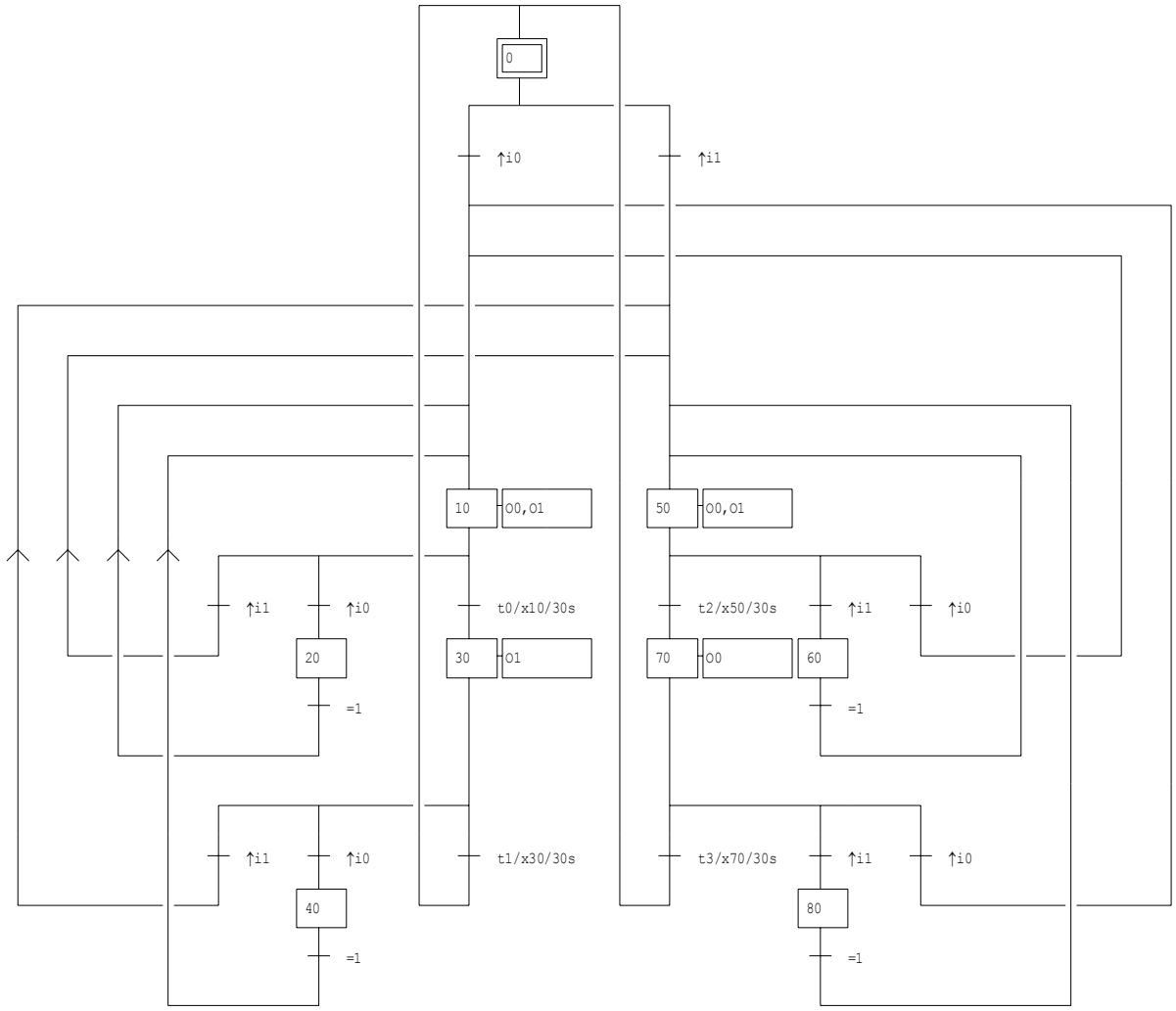


Il suffit d'écrire deux Grafcets identiques. Chacun s'occupe indépendamment d'un interrupteur et d'une ampoule.

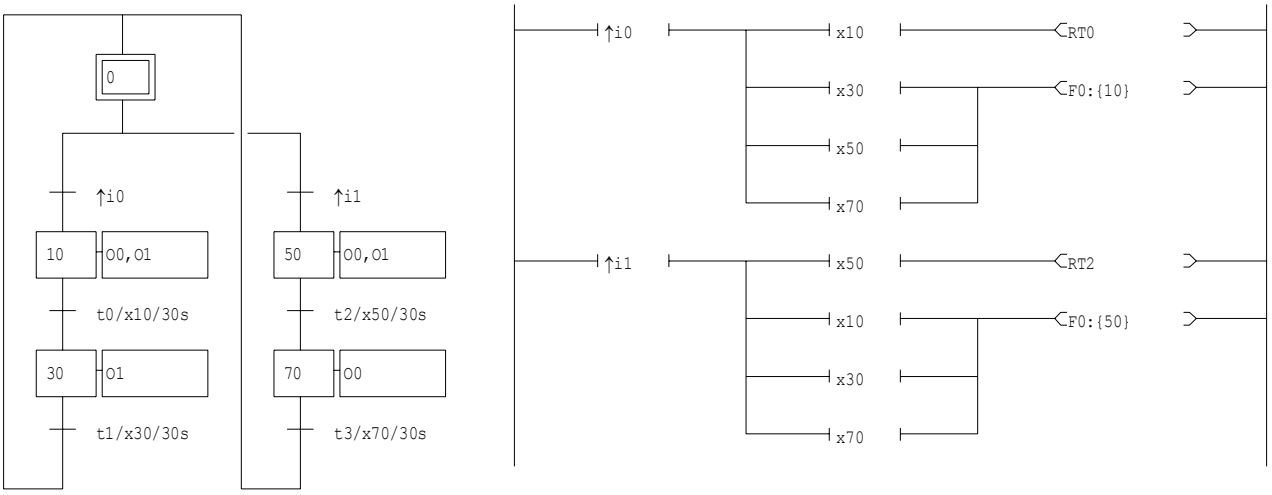
« temporisations, minuteries et autres amusements temporels… »



Une version simple sans la gestion du réarmement de la minuterie.

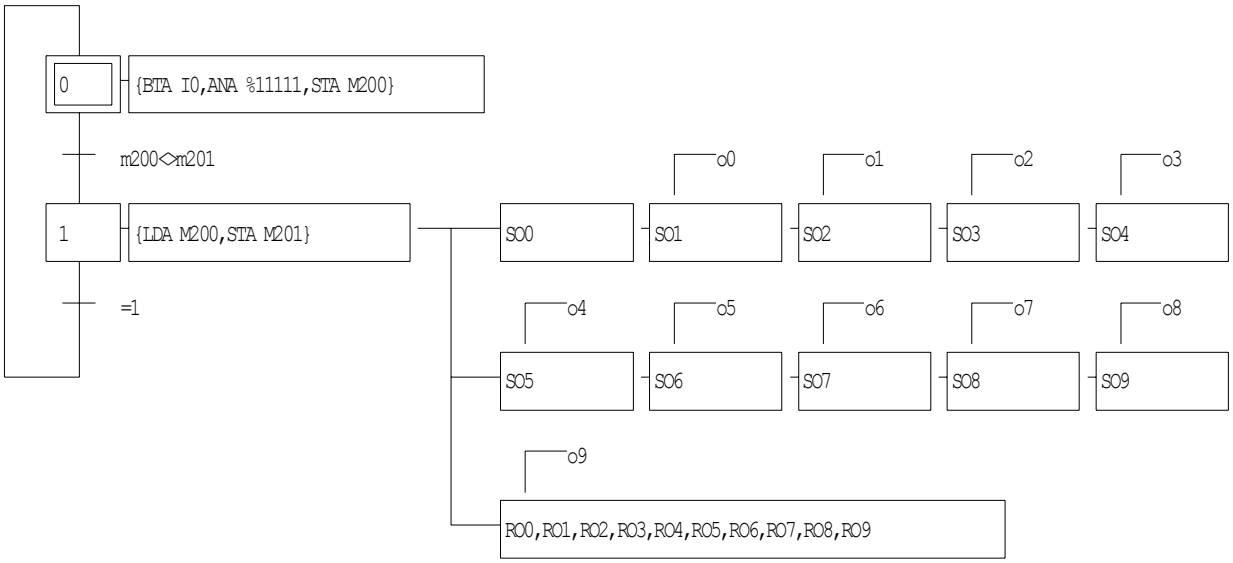


Le réarmement de la minuterie rend le programme très complexe.



Une troisième solution utilisant Grafcet, langage ladder et forçages de Grafcet. Le programme reste lisible.

« variation sur le thème du va et vient …»

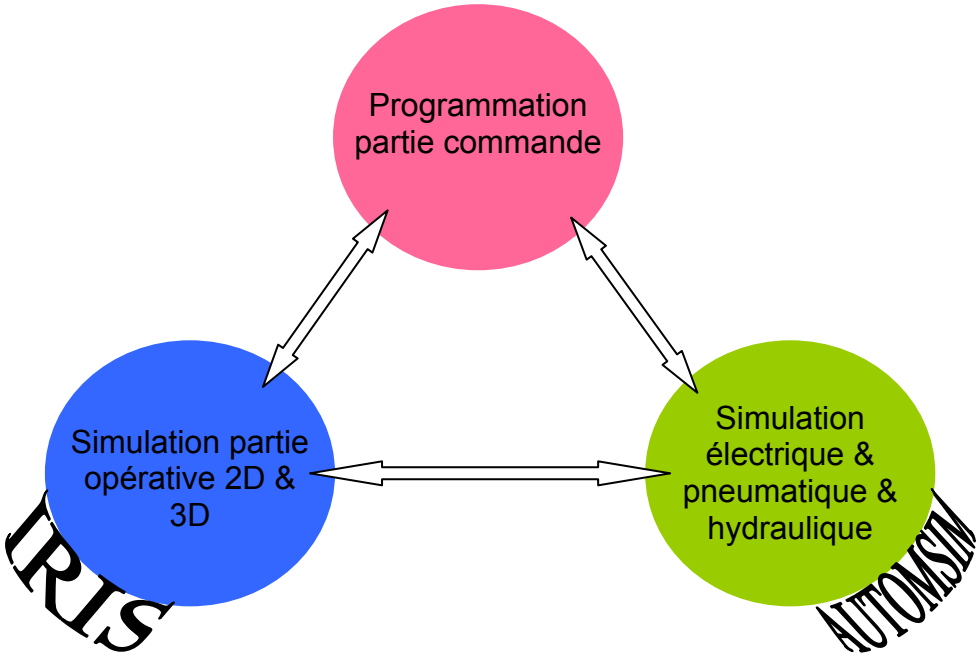


AUTOMSIM

Introduction

AUTOMSIM est un module de simulation pneumatique / électrique / hydraulique.

Il peut être utilisé de façon autonome ou en complément des fonctionnalités d'AUTOMGEN[8] :

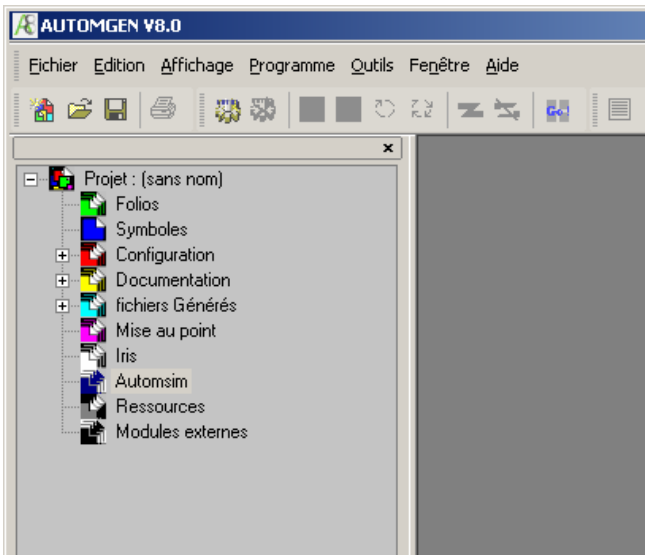


Installation

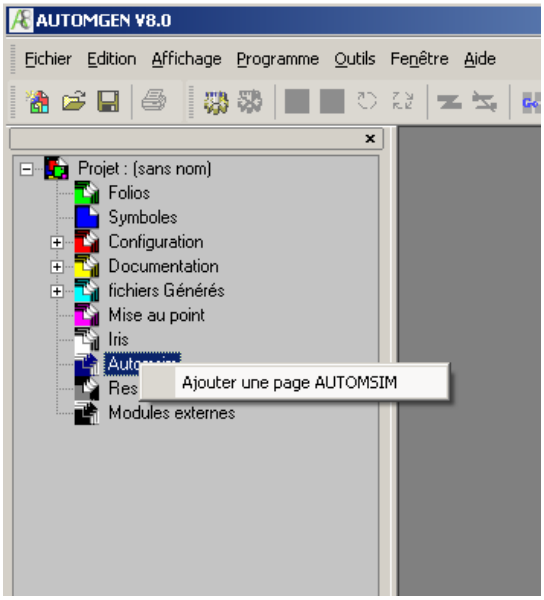
Pour installer AUTOMSIM, installez AUTOMGEN[8]. Dans les options, vérifiez que « AUTOMSIM » est validé.

Prise en main

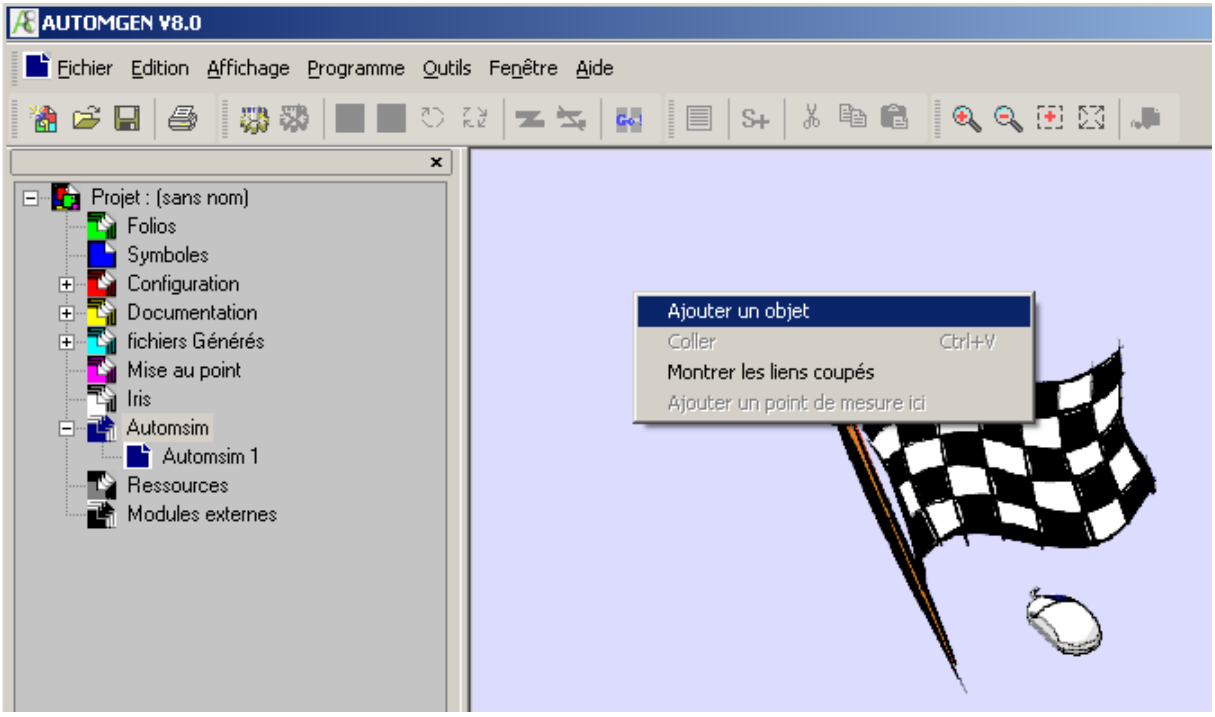
Réalisons un simple exemple : un vérin + un distributeur.



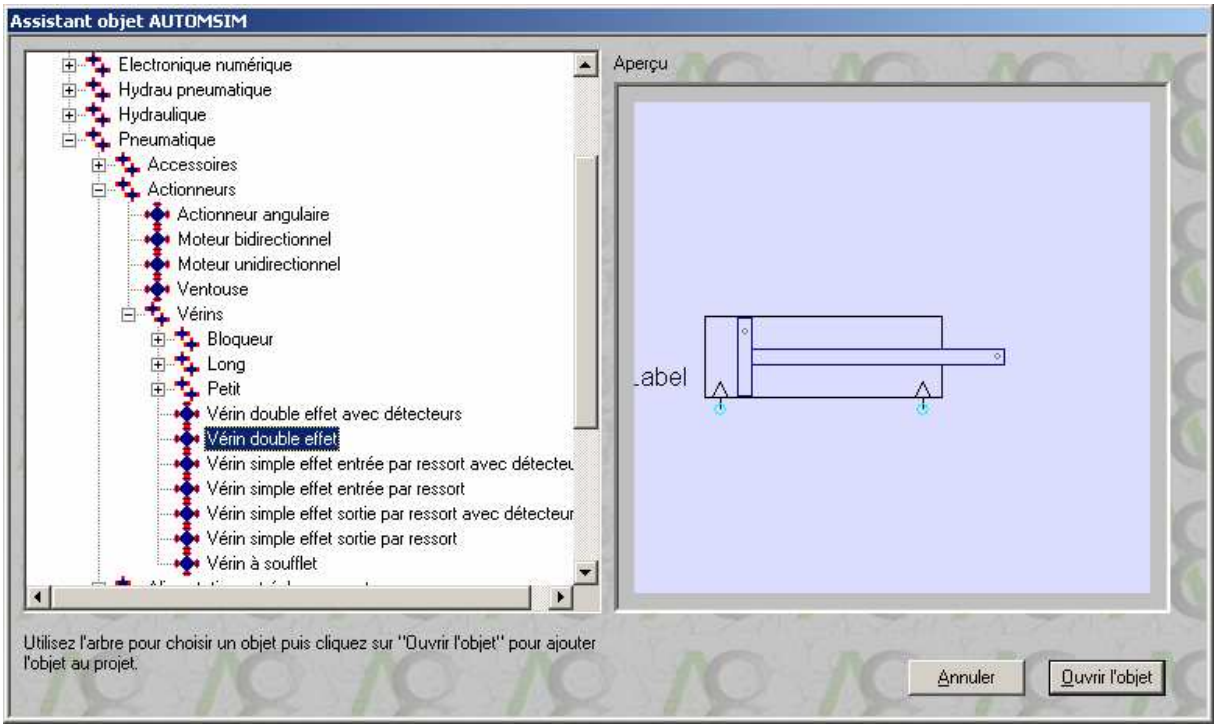
Clic droit de la souris sur « AUTOMSIM »



Choisir « Ajouter une page AUTOMSIM »



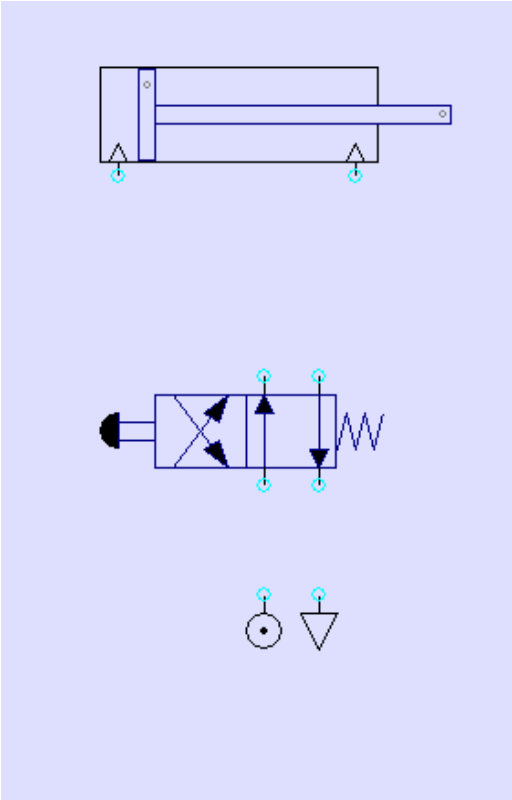
Clic droit de la souris sur le folio AUTOMSIM (partie droite) puis choisir « Ajouter un objet »



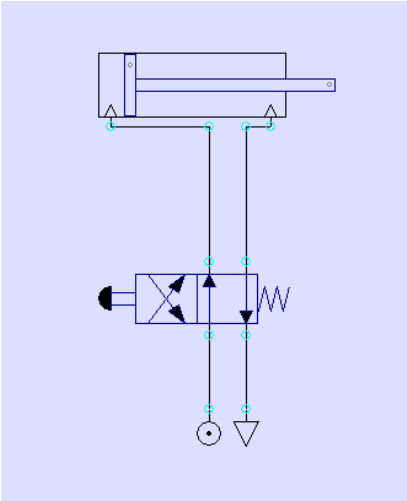
Choisir « vérin double effet », puis cliquez sur « Ouvrir l'objet ».

Répétez les opérations ci-dessus pour ajouter un distributeur 4/2 à pilotage manuel monostable, une alimentation et un échappement pneumatique.

Vous devez obtenir ceci :



Créez les connexions entre les différents éléments : déplacez le curseur au dessus des connexions (les ronds bleus clairs), enfoncez le bouton gauche de la souris puis relâchez le, déplacez le curseur de la souris jusqu'à la connexion où le lien doit être connecté, enfoncez le bouton gauche de la souris puis relâchez le. Renouvelez ceci pour chaque connexion jusqu'à obtenir le résultat suivant :



Cliquez sur le bouton « GO » dans la barre d'outils.

La tige du vérin sort. Pour la faire rentrer, cliquez sur la commande manuelle du distributeur.

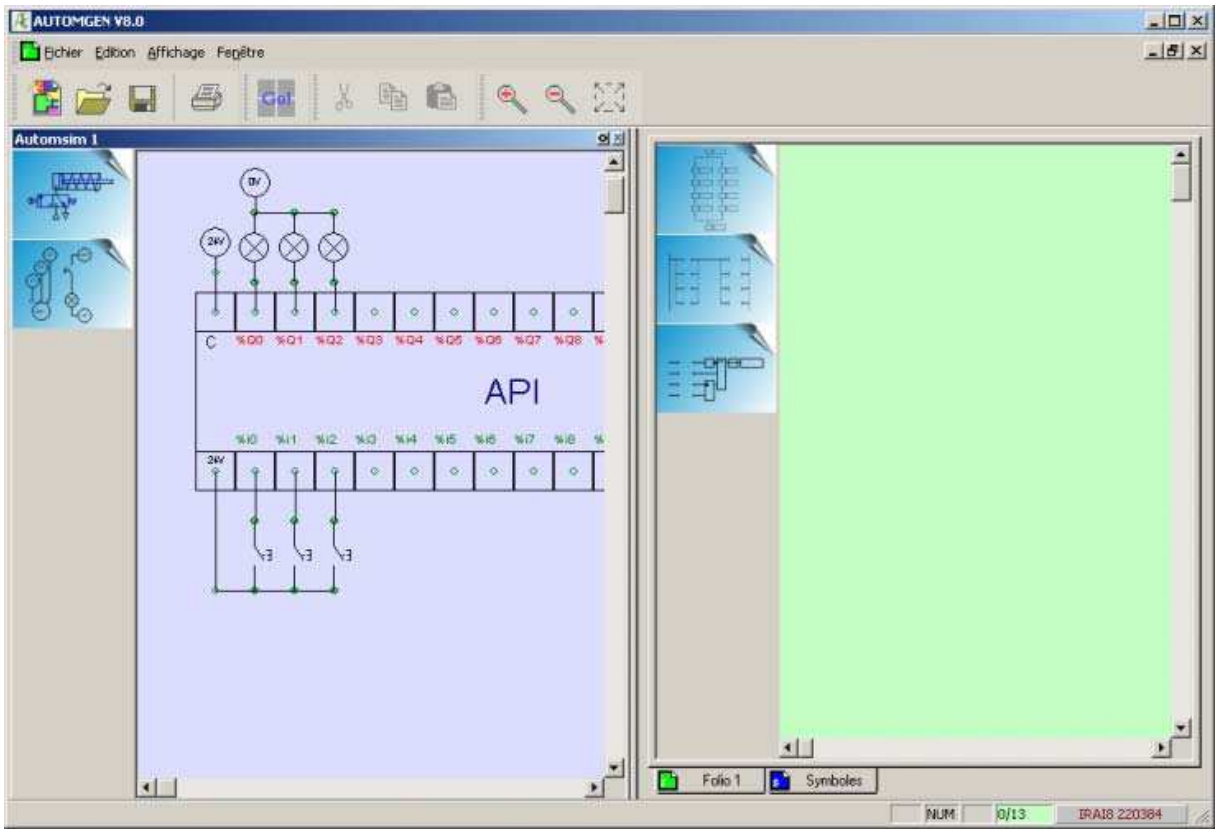
Pendant le fonctionnement, vous pouvez faire des modifications, ajouter des objets, les déplacer, etc…

AUTOMSIM ne nécessite pas de stopper la simulation !

Pour mettre fin à la simulation, cliquez de nouveau sur « GO ».

Mode « débutant » d'AUTOMGEN

Le mode « Débutant » d'AUTOMGEN permet d'utiliser des palettes simplifiées pour AUTOMSIM. Le mode « Débutant, automatisme, électricité et pneumatique » permet d'utiliser un environnement où un automate programmable est déjà « pré câblé » :



Pour faire référence aux entrées et aux sorties de l'automate, les syntaxes %In (avec n=numéro de l'entrée) ou %Qn (avec n=numéro de la sortie) peuvent être utilisées. Il est également possible de « traîner » le nom d'une entrée ou d'une sortie en cliquant sur le nom de celle-ci sur le schéma AUTOMSIM et en déplaçant le curseur vers une transition ou un rectangle d'action du folio AUTOMGEN. La programmation peut ainsi se faire uniquement à la souris.

Utilisation d'AUTOMSIM

Organisation des applications

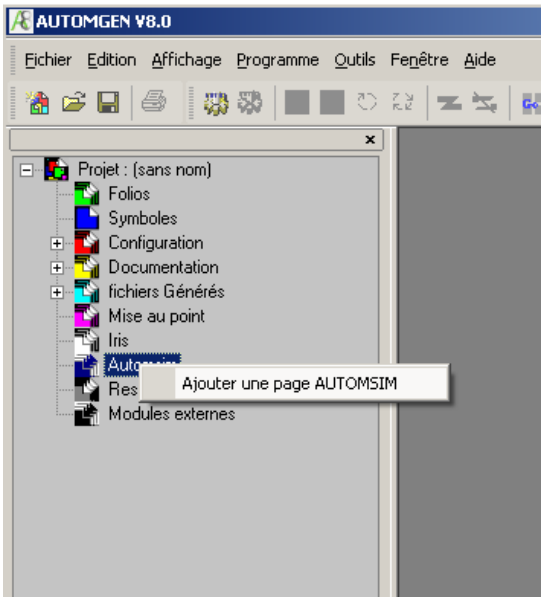
Les applications AUTOMSIM sont écrites sur un ou plusieurs folios qui apparaissent dans l'arborescence d'AUTOMGEN[8]. Sur ce ou ces folios sont ensuite placés des objets : un objet = un élément tel qu'un vérin ou un contact électrique.

Ouvrir une application existante

Le sous répertoire « Exemples / automsim » du répertoire d'installation d'AUTOMGEN[8] contient des exemples réalisés avec AUTOMSIM.

Créer un folio AUTOMSIM

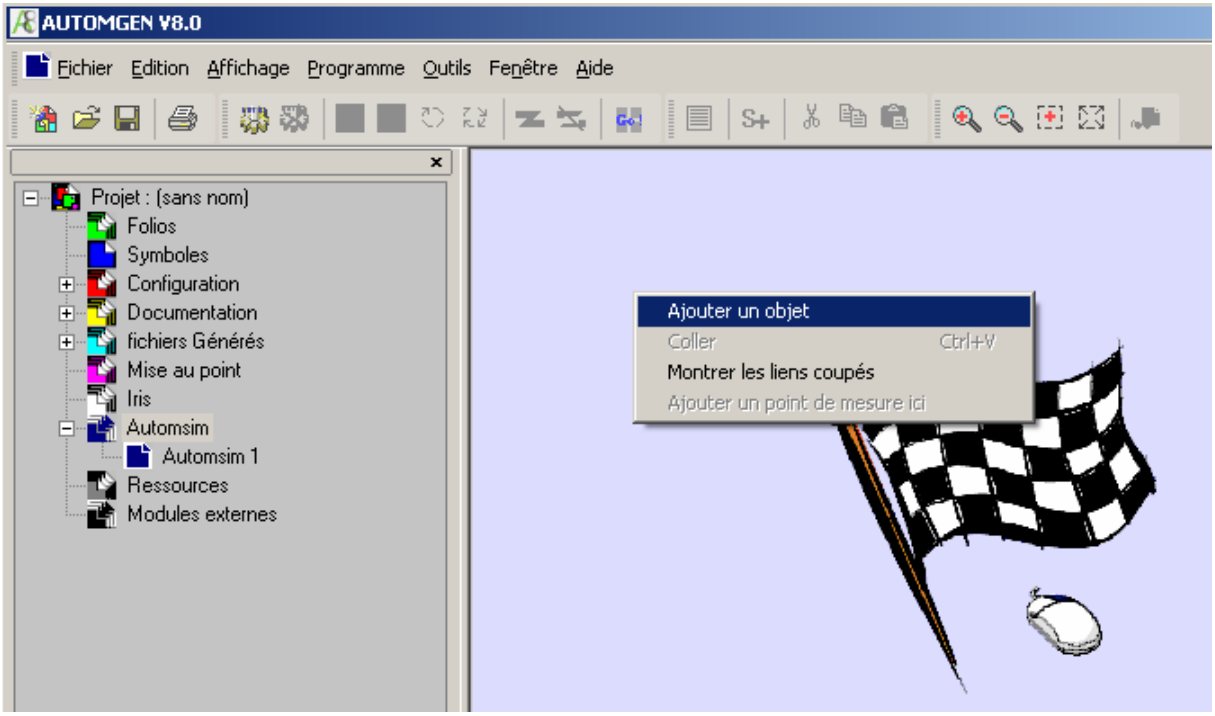
Pour ajouter un folio AUTOMSIM dans l'arborescence d'un projet, cliquez avec le bouton droit de la souris sur l'élément « AUTOMSIM » dans l'arborescence, puis choisissez « Ajouter une page AUTOMSIM ».



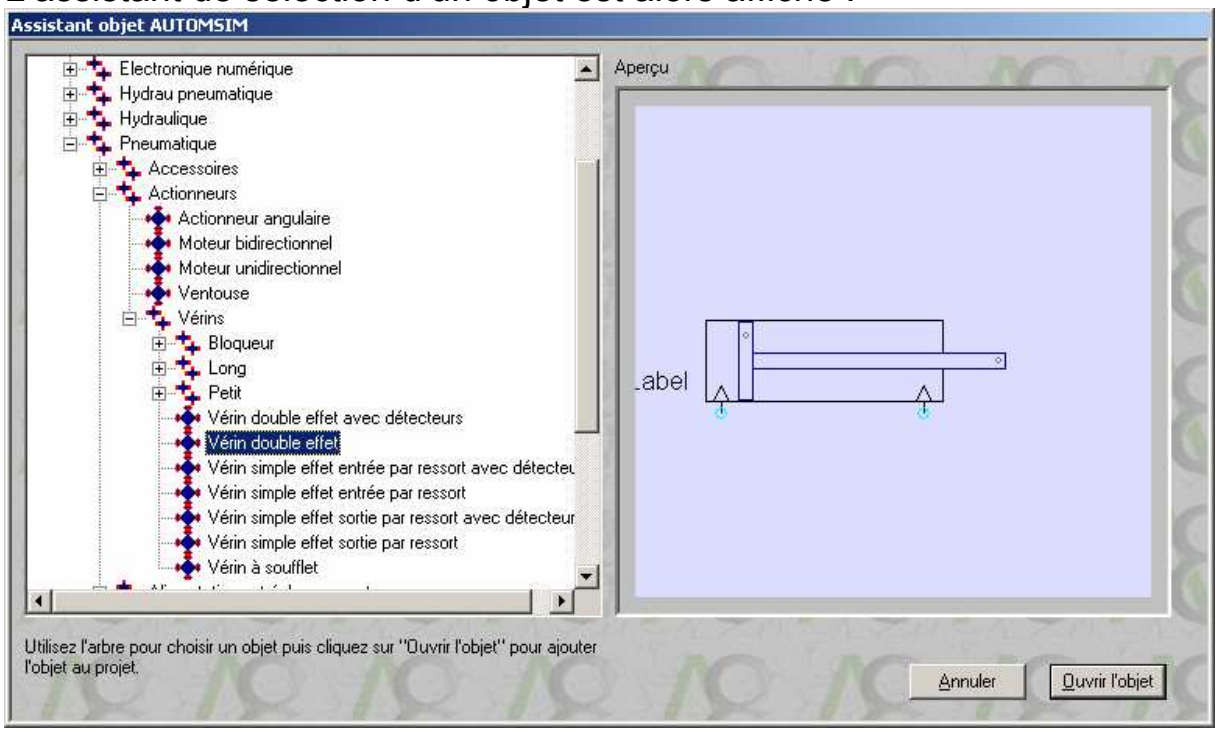
Un folio AUTOMSIM est alors créé.

Ajouter un objet sur un folio AUTOMSIM

Cliquez avec le bouton droit de la souris sur le folio AUTOMSIM (affiché à droite ci-dessous) et choisissez « Ajoutez un objet ».



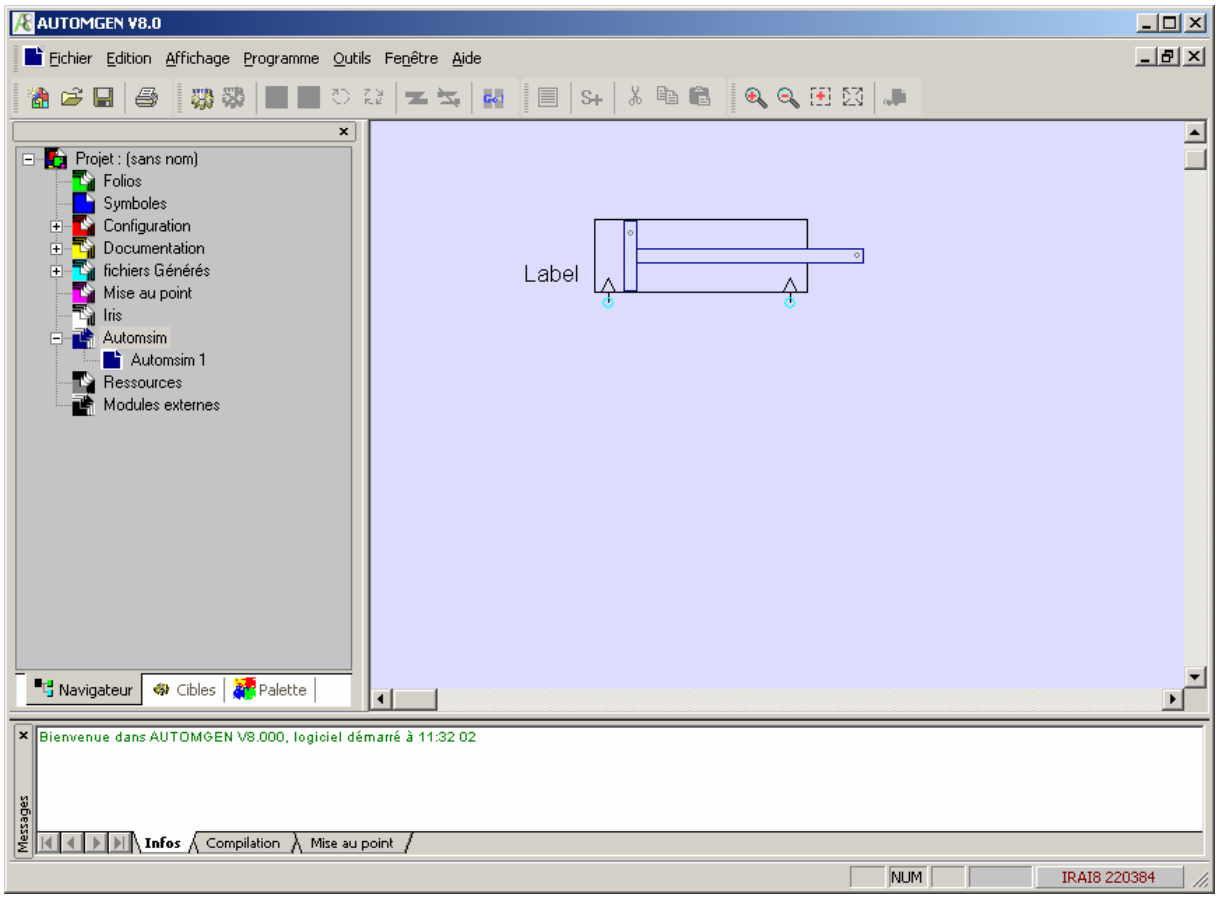
L'assistant de sélection d'un objet est alors affiché :



Cet assistant montre un aperçu de l'objet en bas de la fenêtre. Pour ajouter l'objet sur le folio AUTOMSIM, cliquez sur « Ouvrir l'objet ».

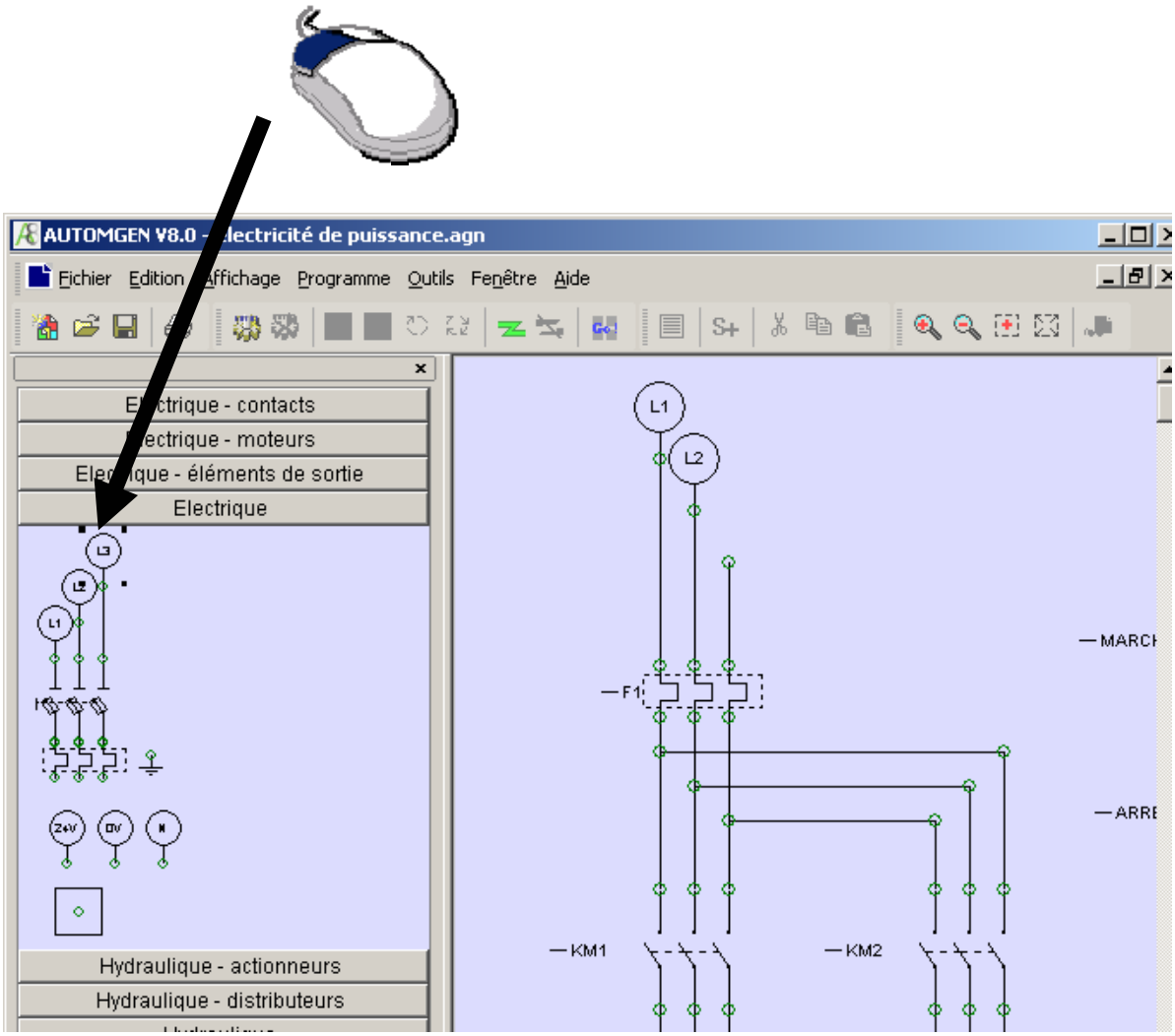
Déplacez ensuite la souris pour positionner l'objet sur le folio AUTOMSIM puis enfoncez le bouton gauche de la souris et relâchez-le pour déposer l'objet.

Vous obtenez le résultat suivant :

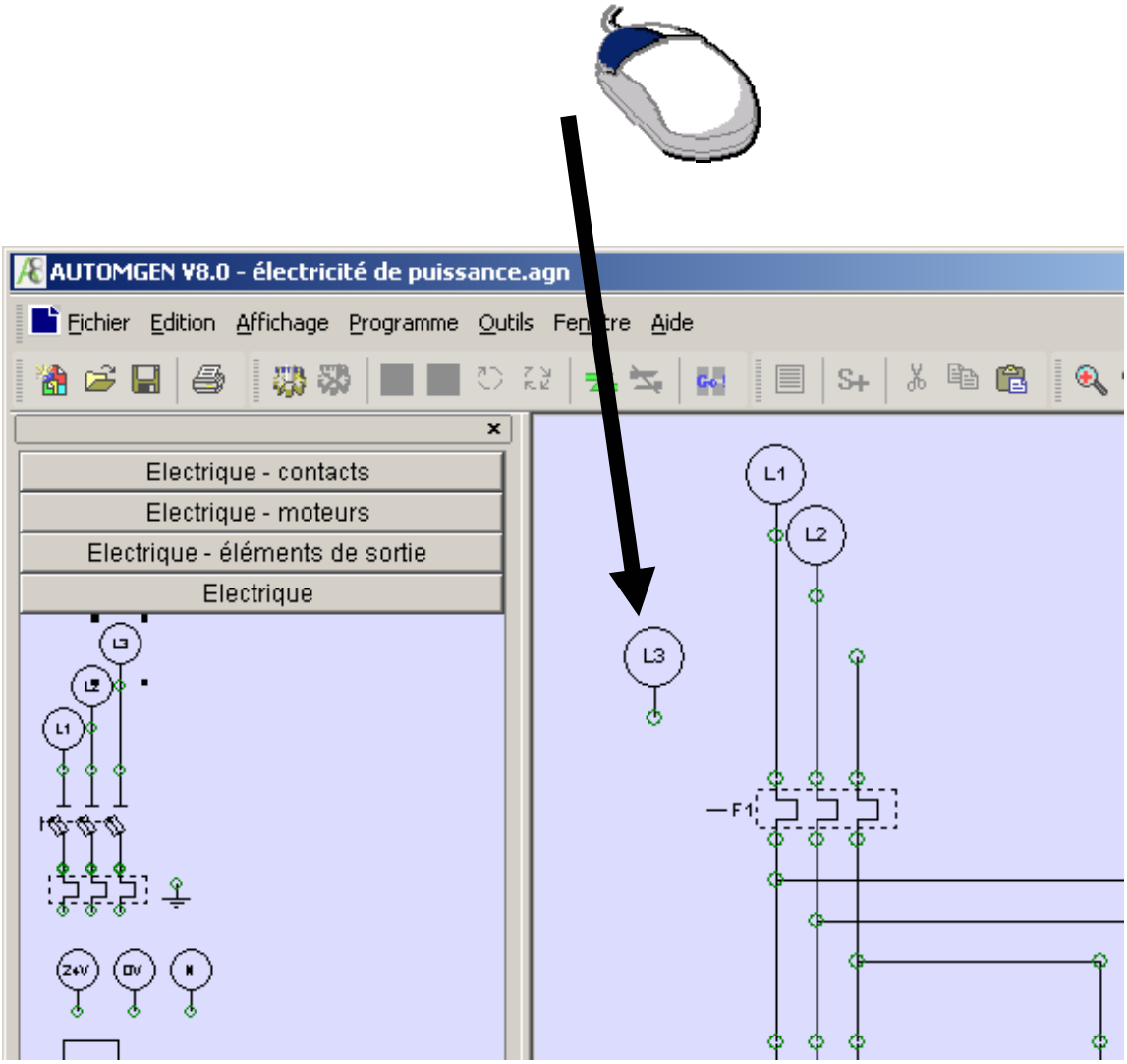


Utiliser la palette

1- cliquez sur le ou les objets dans la palette (ils apparaissent comme sélectionnés : encadrés de carrés noirs).

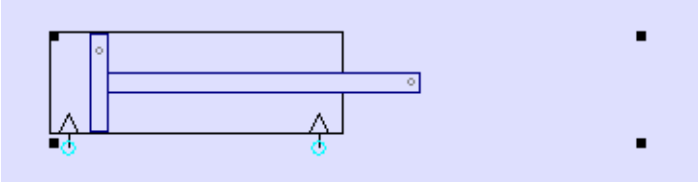


2- cliquez sur le ou les objets sélectionnés, laissez le bouton enfoncé et traînez l'objet sur le folio.



Sélectionner un ou plusieurs objets

Pour sélectionner un objet, déplacez le curseur de la souris au dessus de l'objet, enfoncez le bouton gauche de la souris et relâchez le. Des carrés noirs apparaissent autour des objets lorsqu'ils sont sélectionnés :



Pour désélectionner un objet, répétez la même opération.

Pour sélectionner plusieurs objets : maintenez la touche SHIFT du clavier enfoncée et sélectionnez plusieurs objets avec la méthode décrite ci-dessus.

Pour sélectionner plusieurs objets se trouvant dans une même zone : enfoncer le bouton gauche de la souris, déplacez le curseur de la souris - un rectangle de sélection se dessine - relâchez le bouton gauche de la souris lorsque le rectangle de sélection a la taille désirée.

Pour sélectionner un objet se trouvant sous un autre objet (on peut superposer plusieurs objets) cliquez plusieurs fois avec le bouton gauche de la souris sur les objets qui se recouvrent : à chaque clic, la sélection est déplacée d'un objet à l'autre.

Déplacer un ou plusieurs objets

Déplacez le curseur au dessus d'un ou plusieurs objets sélectionnés – le curseur de la souris prend l'aspect de quatre flèches de direction – enfoncez le bouton gauche de la souris, déplacez les objets en déplaçant la souris, relâchez le bouton gauche de la souris lorsque l'emplacement désiré pour les objets est atteint.

Effacer un ou plusieurs objets

Déplacez le curseur au dessus d'un ou plusieurs objets sélectionnés, enfoncez puis relâchez le bouton droit de la souris et sélectionnez « Effacer ».

Modifier l'orientation d'un ou plusieurs objets

Déplacez le curseur au dessus d'un ou plusieurs objets sélectionnés, enfoncez puis relâchez le bouton droit de la souris et sélectionnez la valeur souhaitée dans le menu « Rotation ».

Copier/couper un ou plusieurs objets vers le presse-papier

Déplacez le curseur au dessus d'un ou plusieurs objets sélectionnés, enfoncez puis relâchez le bouton droit de la souris et sélectionnez « Copier » ou « Coller ».

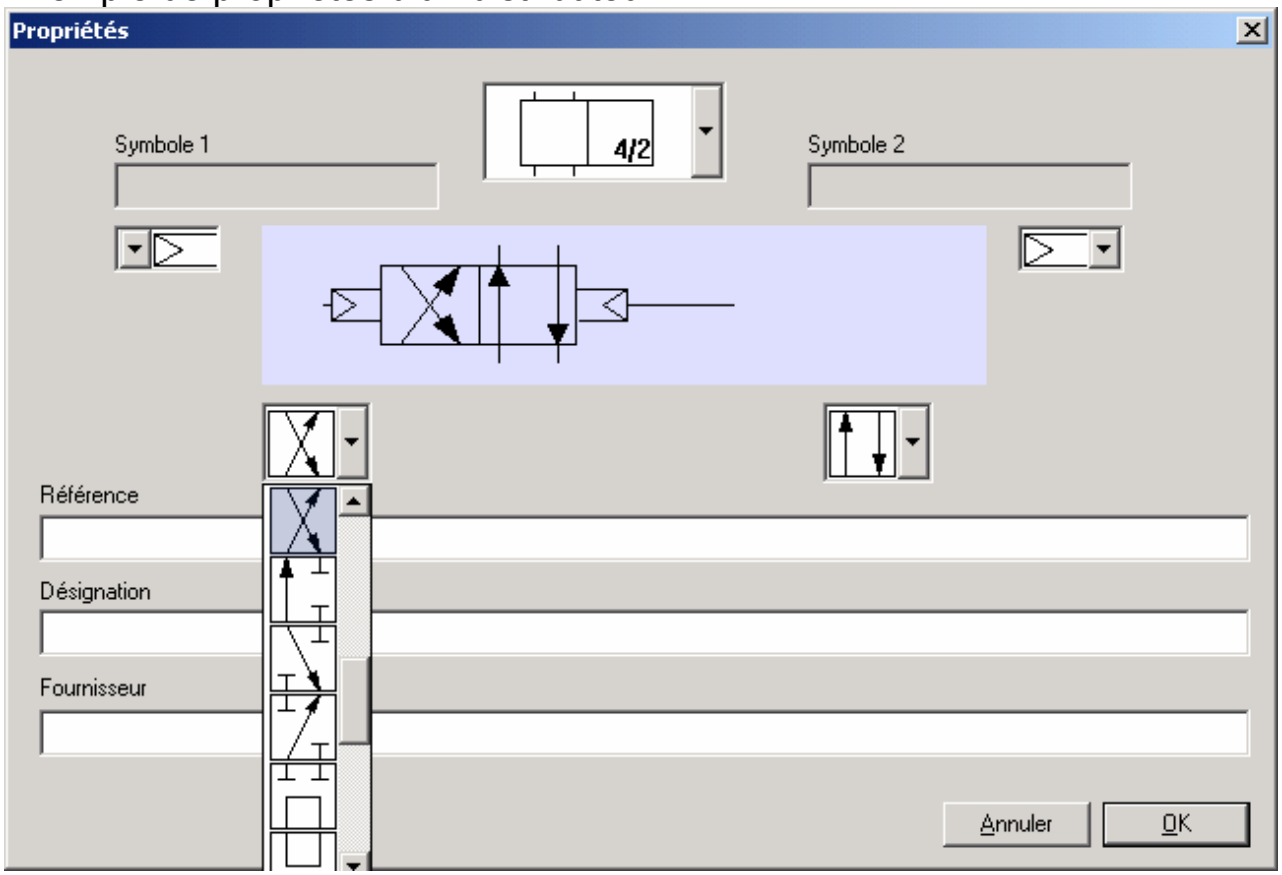
Coller un ou plusieurs objets depuis le presse-papier

Enfoncez puis relâchez le bouton droit de la souris sur un endroit vierge du folio AUTOMSIM et choisissez « Coller » dans le menu.

Modifier les propriétés d'un objet

Déplacez le curseur au dessus d'un ou plusieurs objets sélectionnés, enfoncez puis relâchez le bouton droit de la souris et sélectionnez « Propriétés ».

Exemple de propriétés d'un distributeur :



Exporter un ou plusieurs objets

Déplacez le curseur au dessus d'un ou plusieurs objets sélectionnés, enfoncez puis relâchez le bouton droit de la souris et sélectionnez « Exporter ».

Les objets sont exportés vers des fichiers portant l'extension .ASO.

En exportant vers le sous répertoire « automsim/lib » du répertoire d'installation d'AUTOMGEN, les nouveaux objets ainsi créés apparaissent dans l'assistant d'AUTOMSIM. Le nom du fichier est le nom affiché dans l'assistant. Si le nom doit contenir le caractère '/', substituez ce caractère par '@' dans le nom du fichier.

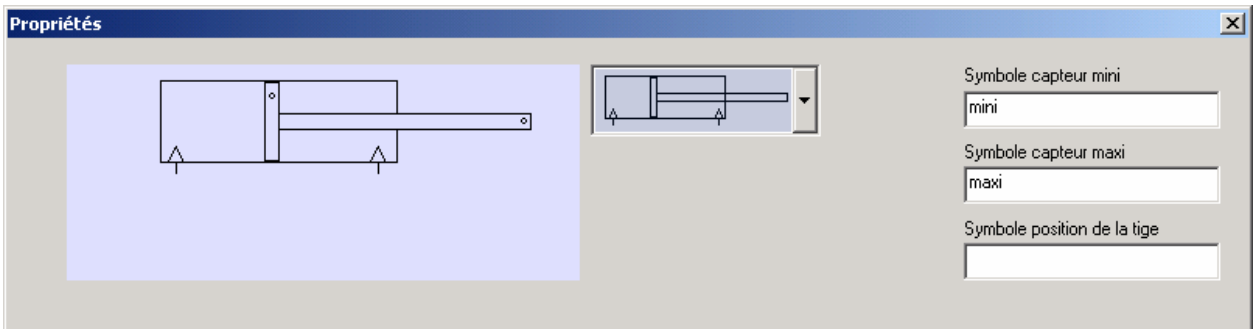
Fonctionnalités avancées

Interactions entre les objets

Les interactions entre les objets AUTOMSIM sont réalisées soient par des liens visuels définis sur les folios (une conduite pneumatique ou électrique reliant deux objets par exemple) soit par un symbole. Un symbole est un nom générique « capteur mini » par exemple. Un symbole peut être un nom quelconque à l'exception des mots clés réservés pour les noms de variables AUTOMGEN (voir le manuel de référence langage d'AUTOMGEN) et des symboles utilisés dans la table des symboles d'AUTOMGEN.

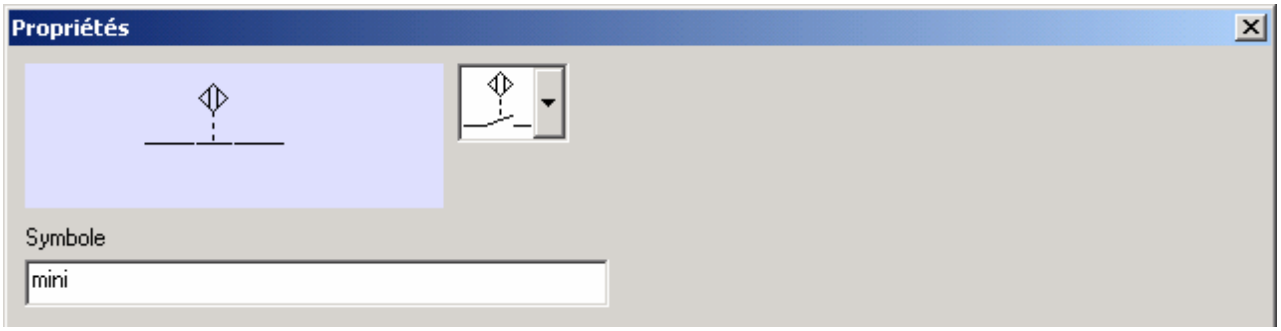
Créer des capteurs associés à un vérin

Les fins de courses mini et maxi d'un vérin peuvent être configurés dans les propriétés du vérin. Exemple :

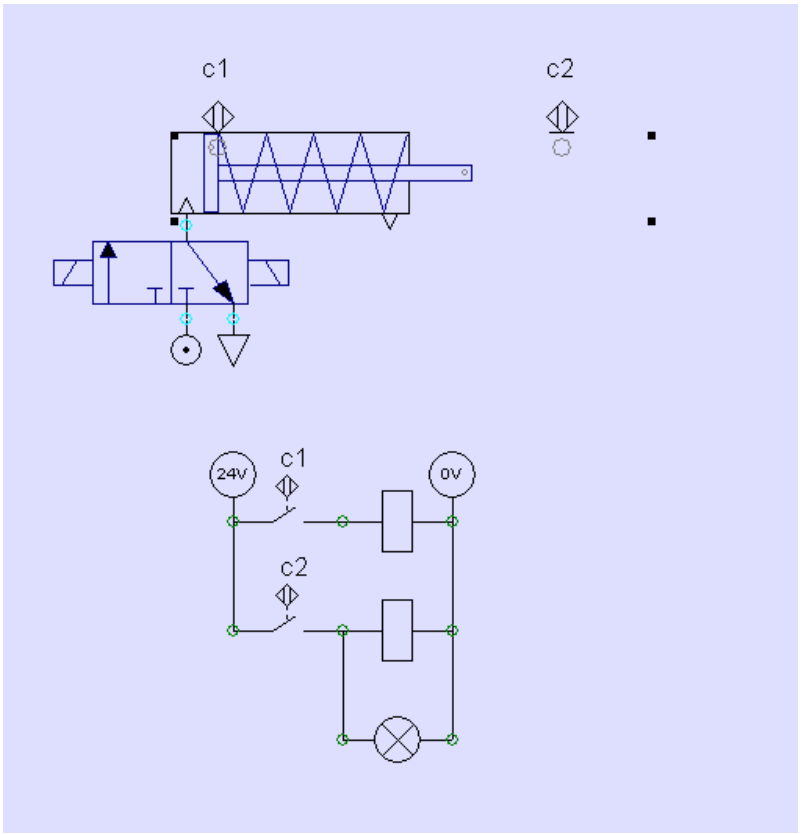


Les symboles utilisés pourront être référencés dans des contacts électriques.

Par exemple :



Les capteurs peuvent également être positionnés directement sur le folio AUTOMSIM. Par exemple :

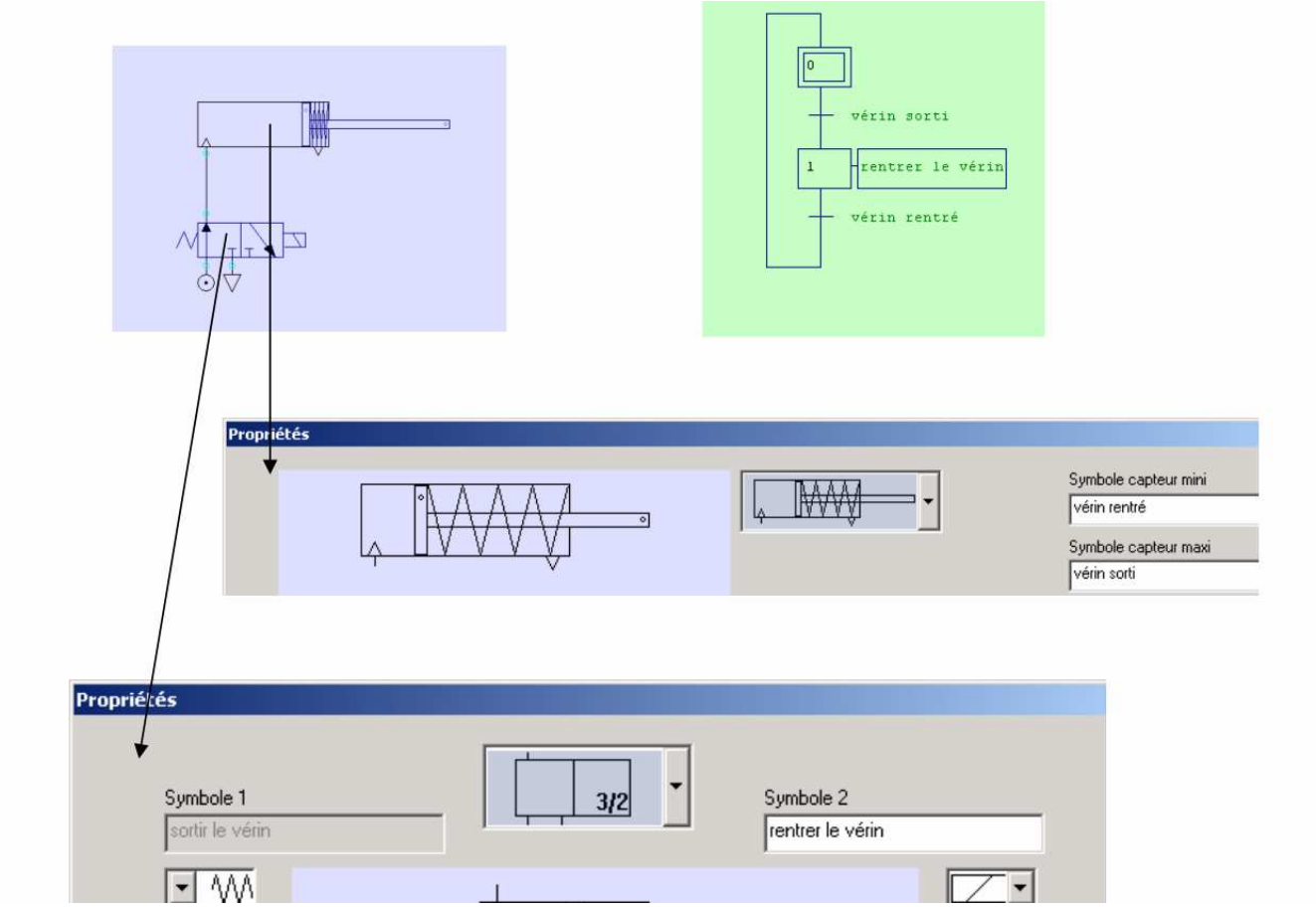


Le cercle gris associé aux objets capteur doit coïncider avec le point gris se trouvant sur le piston ou la tige du vérin pour que le capteur soit activé.

Interactions entre les objets AUTOMSIM et le programme d'automatisme

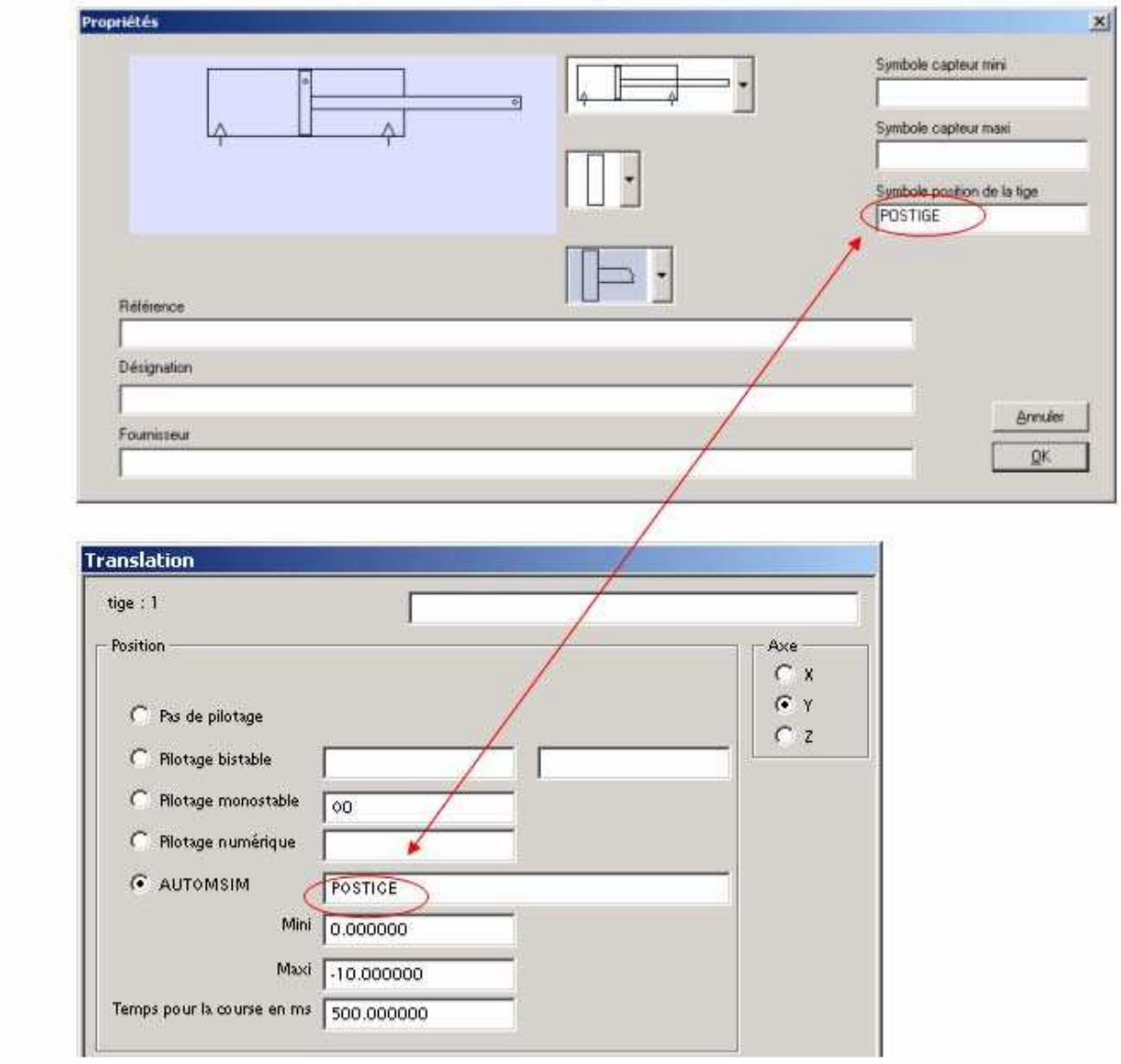
Comme vue ci avant, les symboles utilisés dans les objets AUTOMSIM permettent d'échanger des informations entre les objets. Ces symboles ne doivent être ni des noms de variables AUTOMGEN ni des noms de symboles AUTOMGEN tant que l'on souhaite dialoguer uniquement entre objets AUTOMSIM. Si on utilise un nom de variable AUTOMGEN où un symbole AUTOMGEN, alors les objets AUTOMSIM font référence aux variables AUTOMGEN et peuvent donc selon le cas, lire ou écrire des variables de l'application d'automatisme.

Exemple :



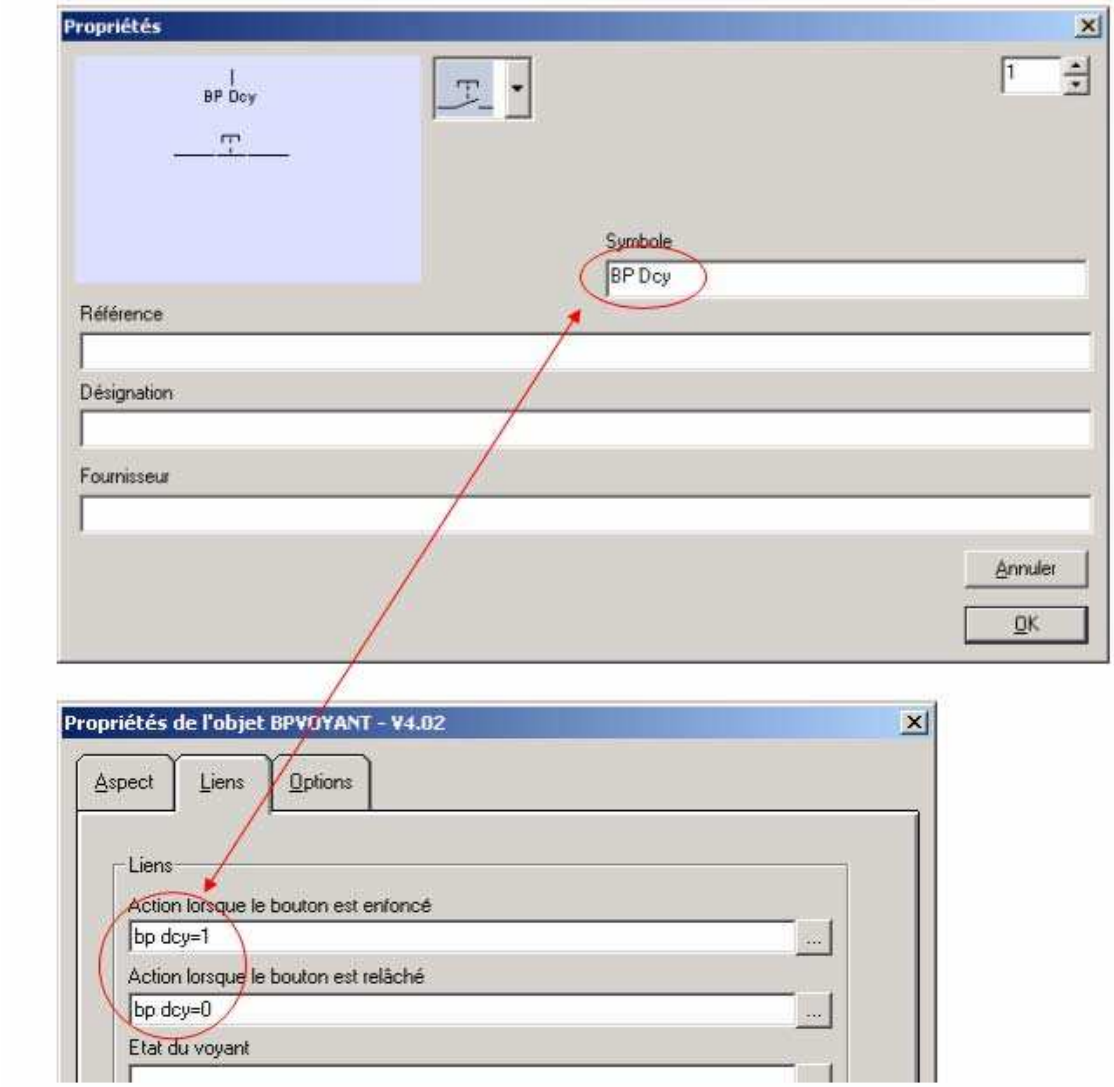
Interactions entre les objets AUTOMSIM et le simulateur de partie opérative IRIS 3D

Dans les comportements IRIS 3D « Translations » et « Rotations », le type « AUTOMSIM » permet de faire référence à la position d'un objet vérin d'AUTOMSIM (voir l'exemple complet2.agn).



Interactions entre les objets AUTOMSIM et les objets de supervision IRIS2D

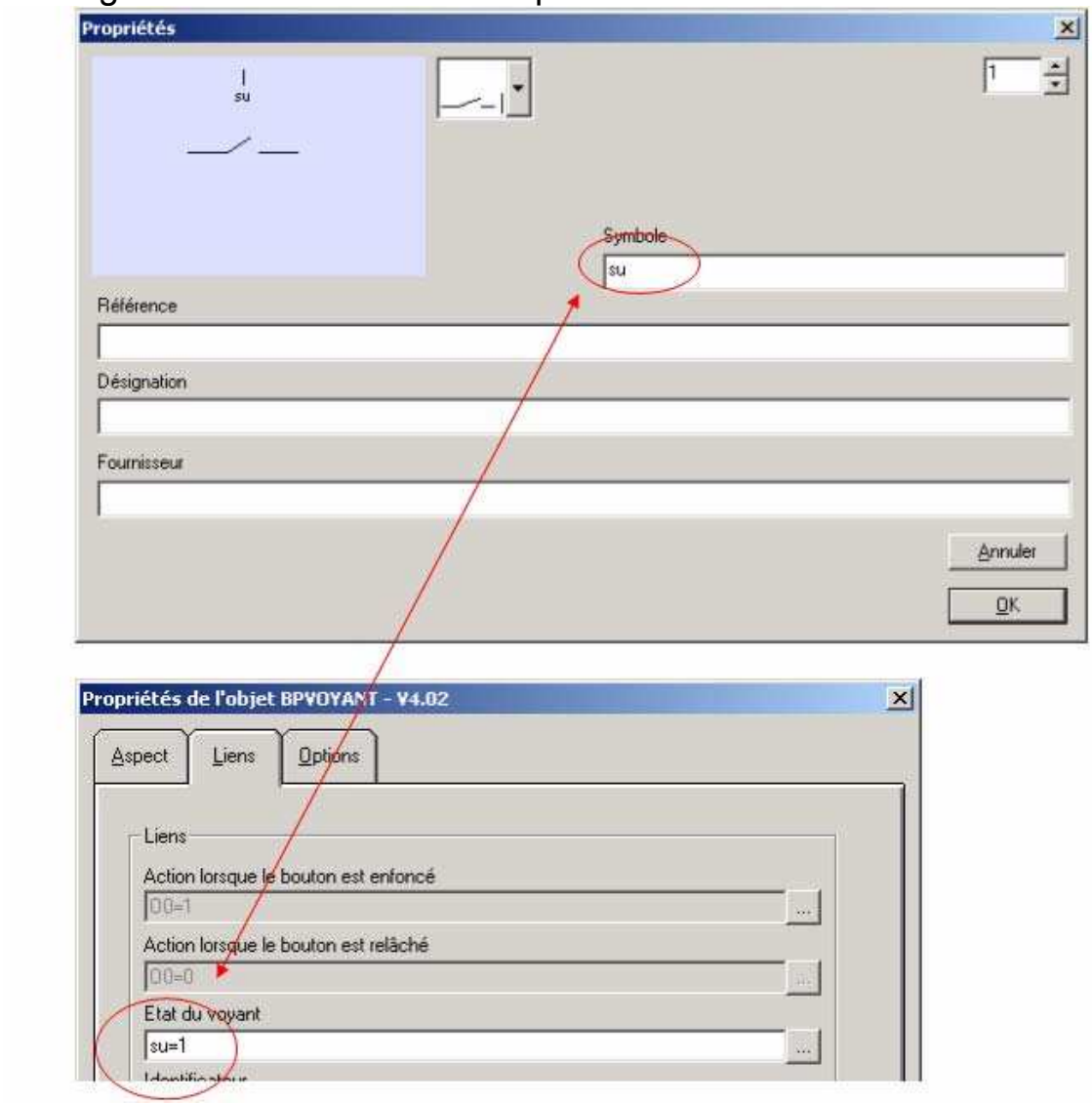
Comment réaliser le lien entre un bouton poussoir ou un interrupteur d'IRIS2D et un bouton poussoir ou un interrupteur d'AUTOMSIM ?



Comment réaliser un lien entre un objet d'AUTOMSIM et un voyant d'IRIS 2D ?

Remarque : notez que les variables d'AUTOMSIM sont considérées comme des variables numériques. Il est donc nécessaire d'écrire « su=1 ».

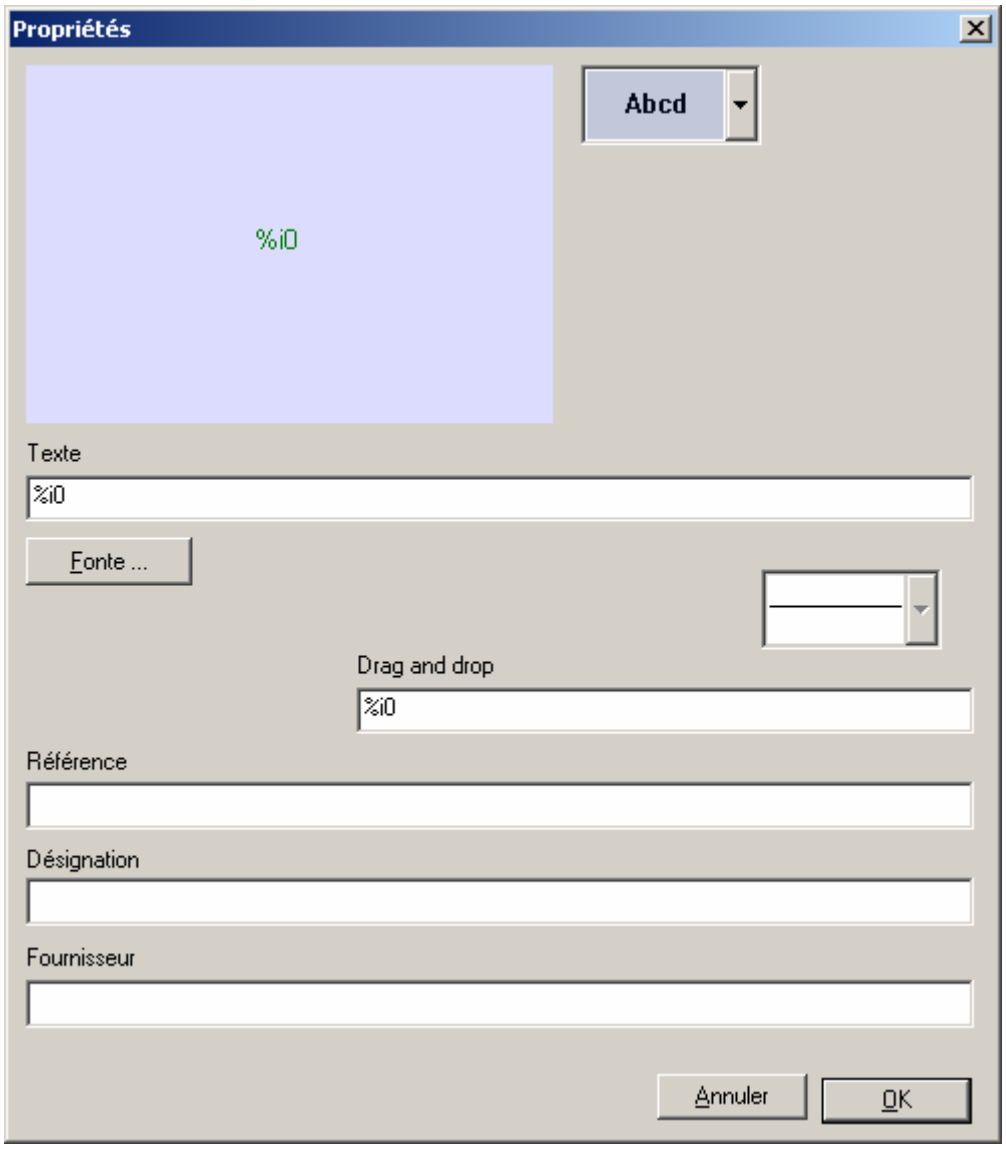
Voir également : IRIS3D dans la partie consacrée à l'environnement.



Drag and drop depuis un objet AUTOMSIM vers un folio AUTOMGEN

Cette fonctionnalité est par exemple utilisée dans le mode « Débutant » pour pouvoir « traîner » le nom des entrées ou des sorties depuis l'automate vers le folio AUTOMGEN.

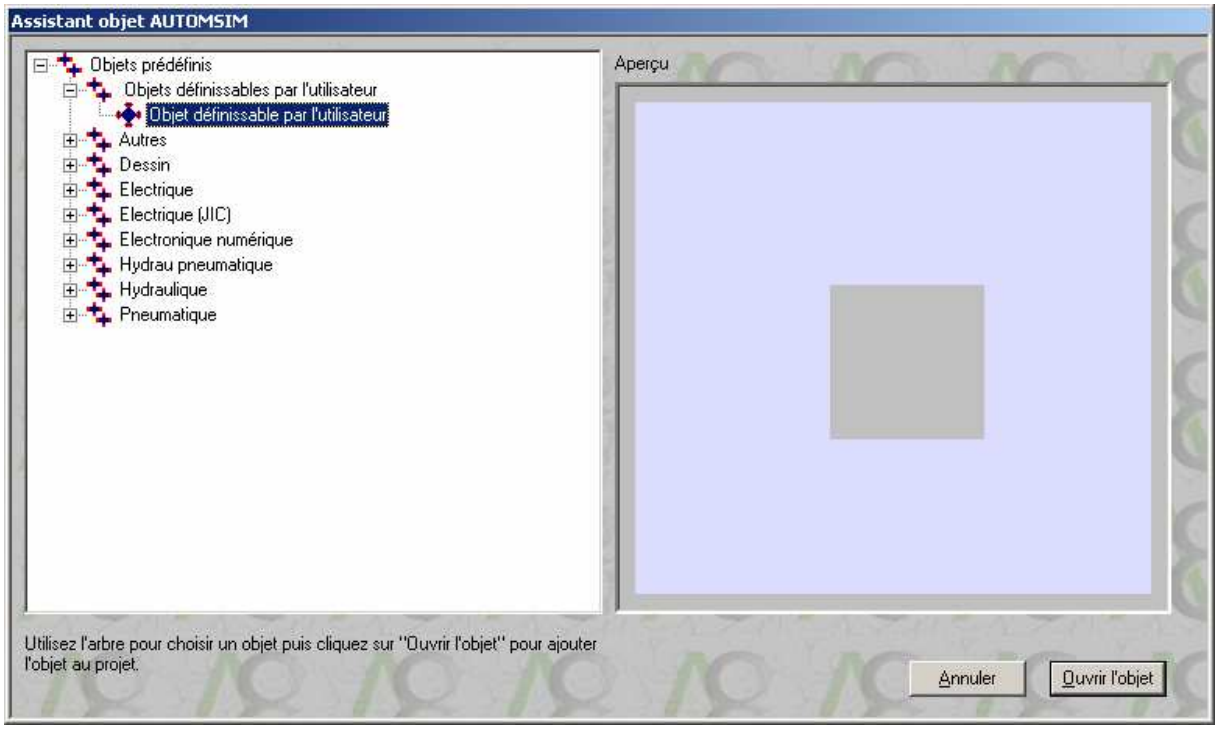
Pour utiliser cette fonctionnalité, utilisez un objet AUTOMSIM de type « Dessin » et documentez la rubrique « Drag and drop » avec le texte qui pourra être « traîné » du folio AUTOMSIM vers le folio AUTOMGEN.



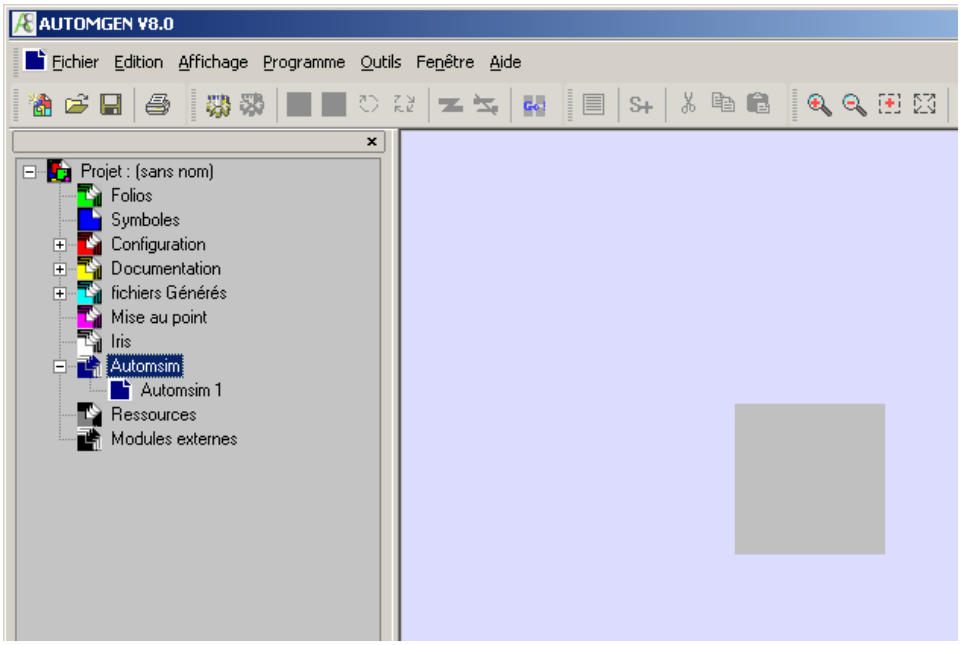
Objets définissables par l'utilisateur

L'objet définissable par l'utilisateur va vous permettre de créer vos propres objets de simulation.

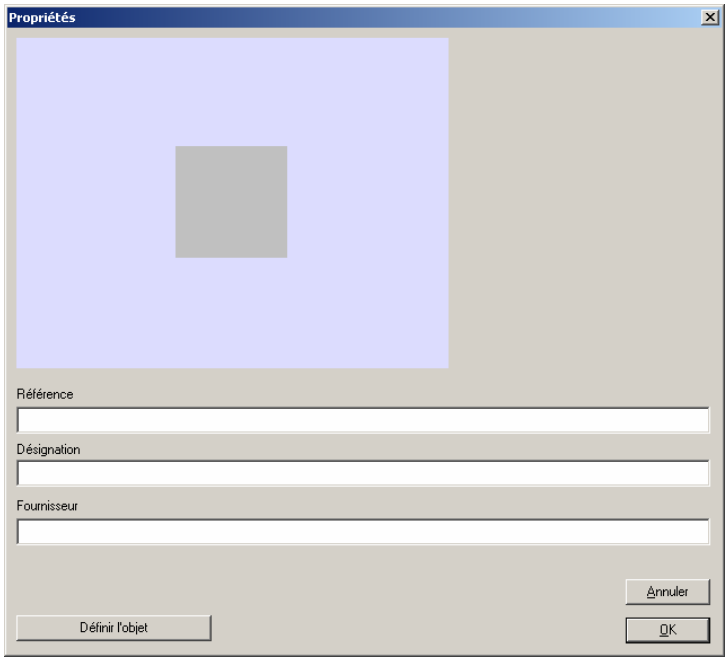
Pour créer un tel objet, ouvrez l'objet suivant :

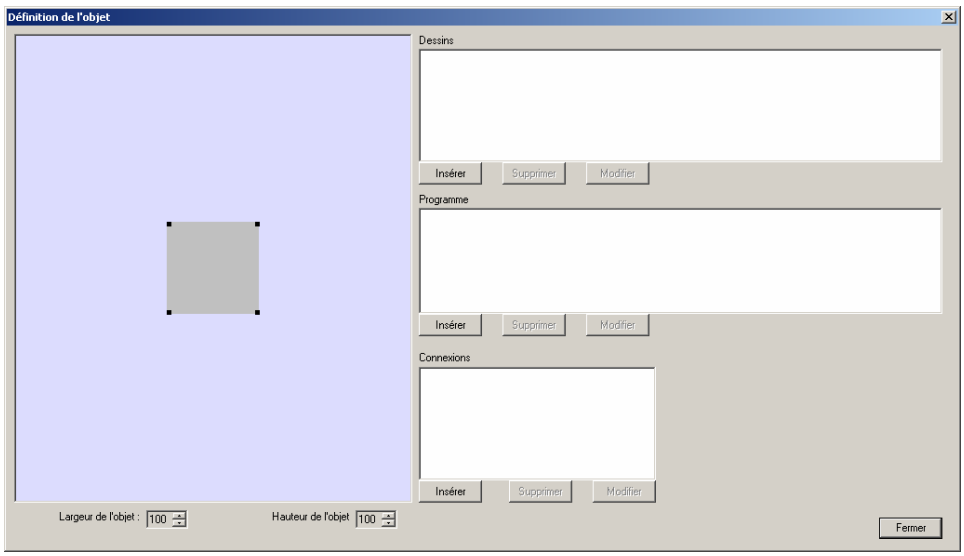


L'objet apparaît sous la forme d'un carré gris tant qu'il n'a pas été paramétré :



Pour accéder à la définition de l'objet, ouvrez les propriétés de l'objet (sélection de l'objet, clic droit dessus puis « Propriétés ») et cliquez sur « Définir l'objet ».



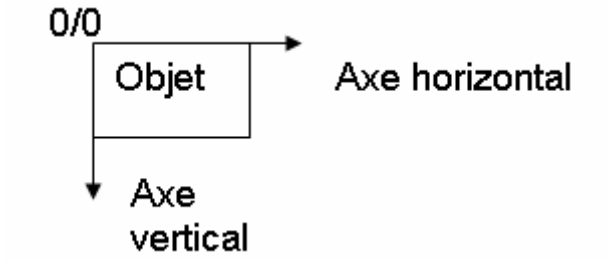


Les zones « Largeur de l'objet » et « Hauteur de l'objet » permettent de définir les dimensions de l'objet.

Les zones « Dessins », « Programme » et « Connexions » permettent respectivement de définir le dessin de l'objet (son apparence), son comportement ainsi que les connexions.

Dessins

Cette zone permet de définir le dessin de l'objet à l'aide de primitive de dessin. Les boutons « Insérer », « Supprimer » et « Modifier » permettent respectivement d'insérer, de supprimer une primitive ou de modifier les paramètres associés à une primitive. Les primitives de dessin utilisent ce système de coordonnées :



Chaque primitive peut recevoir un ou plusieurs paramètres.

A noter que les primitives de dessin ne définissent que l'objet n'ayant pas de rotation, le dessin avec rotation est automatiquement généré par AUTOMSIM. Il en est de même pour l'échelle, les primitives dessinent à l'échelle 1, AUTOMSIM gère la mise à l'échelle en fonction du zoom sélectionné par l'utilisateur.

En cliquant sur « Insérer », une boîte de dialogue vous permet de choisir une primitive de dessin.

Liste des primitives de dessin

Primitive de tracé

Ces primitives réalisent un tracé.

MOVE

Déplace le stylo (sans tracer).

Paramètres :
- position horizontale,
- position verticale.

LINE

Trace une ligne depuis la position actuelle du stylo jusqu'à la position indiquée.

Paramètres :
- position horizontale,
- position verticale.

RECT

Trace un rectangle.

Paramètres :
- position horizontale coin supérieur gauche,
- position verticale coin supérieur gauche,
- position horizontale coin inférieur droit,
- position verticale coin inférieur droit.

ELLI

Trace une ellipse.

Paramètres :
- position horizontale coin supérieur gauche du rectangle englobant l'ellipse,
- position verticale coin supérieur gauche du rectangle englobant l'ellipse,
- position horizontale coin inférieur droit du rectangle englobant l'ellipse,
- position verticale coin inférieur droit du rectangle englobant l'ellipse.

RREC

Trace un rectangle aux coins arrondis.

Paramètres :
- position horizontale coin supérieur gauche,
- position verticale coin supérieur gauche,
- position horizontale coin inférieur droit,
- position verticale coin inférieur droit,
- rayon arrondi horizontal,
- rayon arrondi vertical.

TRIA

Trace un triangle.

Paramètres :
- position horizontale point 1,
- position verticale point 1,
- position horizontale point 2,
- position verticale point 2,
- position horizontale point 3,
- position verticale point 3.

CHOR

Trace une corde (intersection d'une ellipse et d'une ligne droite).

Paramètres :
- position horizontale coin supérieur gauche du rectangle englobant l'ellipse,
- position verticale coin supérieur gauche du rectangle englobant l'ellipse,

- position horizontale coin inférieur droit du rectangle englobant l'ellipse,
- position verticale coin inférieur droit du rectangle englobant l'ellipse,
- position horizontale du départ de la ligne,
- position verticale du départ de la ligne,
- position horizontale de la fin de la ligne,
- position verticale de la fin de la ligne.

ARCE

Trace un arc d'ellipse (partie d'une ellipse coupée par une ligne droite).

Paramètres :
- position horizontale coin supérieur gauche du rectangle englobant l'ellipse,
- position verticale coin supérieur gauche du rectangle englobant l'ellipse,
- position horizontale coin inférieur droit du rectangle englobant l'ellipse,
- position verticale coin inférieur droit du rectangle englobant l'ellipse,
- position horizontale du départ de la ligne,
- position verticale du départ de la ligne,
- position horizontale de la fin de la ligne,
- position verticale de la fin de la ligne.

TEXT

Trace un texte.

Paramètres :
- position horizontale,
- position verticale,
- texte.

Primitives d'attribut

Ces primitives modifient le tracé des primitives de tracé (la couleur des lignes ou du remplissage par exemple).

BRUS

Modifie la couleur de remplissage des figures ou du fond pour les textes.

Paramètre :
- couleur.

PENC

Modifie la couleur des lignes ou du texte.

Paramètre :
- couleur.

FONT

Modifie la fonte du texte.

Autres primitives

JUMP

Saute de façon inconditionnelle.

Paramètre :
- label.

JPIF

Saute de façon conditionnelle.

Paramètres :
- label,
- élément 1,
- type de comparaison,
- élément 2.

(Voir plus loin les primitives de programmation pour plus d'informations).

DISP

Affiche l'état d'une variable. Peut être utilisé pour la mise au point d'un objet en affichant la valeur d'une variable associée à l'objet.

Paramètres :
- variable,
- position horizontale,
- position verticale.

Programme

Cette zone permet de définir le programme régissant le fonctionnement de l'objet. Chaque objet dispose de variables :

128 variables entières 32 bits,
128 variables flottantes 32 bits.

Ainsi que pour chaque connexion :
- une valeur flottante en entrée,
- une valeur flottante en sortie,
- un mode d'écriture associé qui peut prendre les valeurs suivantes :
  - 0 : aucune écriture n'est réalisée,
  - 1 : la valeur « valeur flottante en sortie » est écrite,
  - 2 : une connexion est réalisée avec la connexion dont le numéro se trouve dans « valeur flottante en sortie »,
  - 3 : blocage (bouchon pneumatique ou hydraulique).

Les variables entières internes suivantes sont spéciales :

125 : contient 0 si la visualisation dynamique est active, 1 autrement (utile pour avoir un dessin différent en visualisation dynamique et hors visualisation dynamique).

126 : contient une valeur représentant un évènement utilisateur : 0=pas d'évènement, 1=bouton gauche de la souris relâche, 2=bouton gauche de la souris enfoncé, 3=bouton droit de la souris relâché, 4=bouton droit de la souris enfoncé.

127 : contient le temps écoulé en ms entre 2 traitement du programme.

Liste des primitives de programme

MOVV

Recopie une constante ou une variable dans une variable.

Paramètres :
- variable de destination,
- variable ou constante source.

ADDV

Ajoute une constante ou une variable à une constante ou une variable et place le résultat dans une variable.

Paramètres :
- variable de destination,
- variable ou constante source 1,
- variable ou constante source 2.

SUBV

Soustrait une constante ou une variable à une constante ou une variable et place le résultat dans une variable.

Paramètres :
- variable de destination,
- variable ou constante source 1,
- variable ou constante source 2.

MULV

Multiplie une constante ou une variable à une constante ou une variable et place le résultat dans une variable.

Paramètres :
- variable de destination,
- variable ou constante source 1,
- variable ou constante source 2.

DIVV

Divise une constante ou une variable à une constante ou une variable et place le résultat dans une variable.

Paramètres :
- variable de destination,
- variable ou constante source 1,
- variable ou constante source 2.

ORRV

Effectue un OU bit à bit entre une constante ou une variable et une constante ou une variable et place le résultat dans une variable.

Paramètres :
- variable de destination,
- variable ou constante source 1,
- variable ou constante source 2.

ANDV

Effectue un ET bit à bit entre une constante ou une variable et une constante ou une variable et place le résultat dans une variable.

Paramètres :
- variable de destination,
- variable ou constante source 1,
- variable ou constante source 2.

XORV

Effectue un OU exclusif bit à bit entre une constante ou une variable et une constante ou une variable et place le résultat dans une variable.

Paramètres :
- variable de destination,
- variable ou constante source 1,
- variable ou constante source 2.

JUMP

Saute de façon inconditionnelle.

Paramètre :
- label.

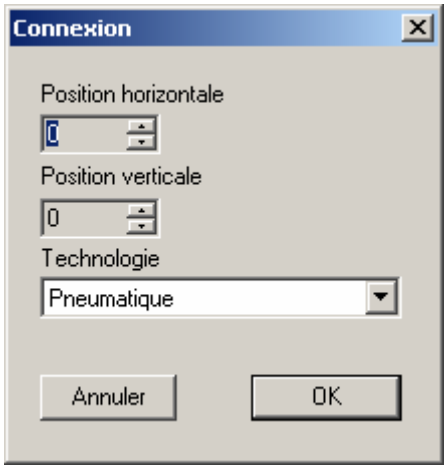
JPIF

Saute de façon conditionnelle.

Paramètres :
- label,
- élément 1,
- type de comparaison,
- élément 2.

Connexions

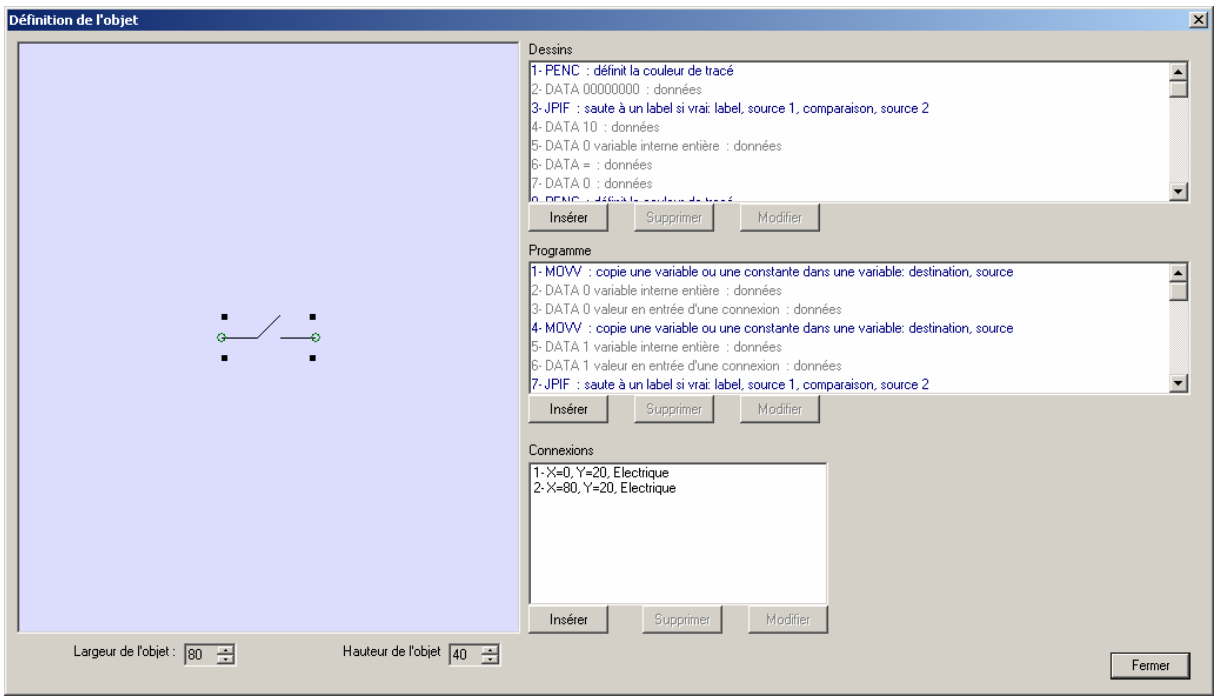
Permet de créer les points de connexion de l'objet. En cliquant sur « Insérer », la boîte de dialogue suivante s'ouvre :



Pour chaque connexion, définissez la position et la technologie. Le numéro affiché en face de chaque connexion doit être utilisé pour accéder à la valeur dans la programmation de l'objet.

Exemple

Le sous répertoire « Exemples\automsim » du répertoire d'installation d'AUTOMGEN contient un exemple illustrant l'utilisation de l'objet définissable par l'utilisateur : un contact :



POST-PROCESSEURS

Généralités

Les post-processeurs sont des modules logiciels permettant de traduire les fichiers de code pivot générés par le compilateur AUTOMGEN en fichiers exécutables sur une cible ainsi que d'assurer la connexion dynamique à la cible.

Le mot « Cible » désigne de façon générique un système programmable capable d'exécuter une application.

Un post-processeur d'AUTOMGEN permet de programmer un type ou un ensemble de type de cibles (généralement une famille d'automates partageant le même langage de programmation et programmable avec le même post-processeur dans AUTOMGEN).

Ce manuel contient en première partie des notions fondamentales communes à tous les post-processeurs. Viennent ensuite des informations spécifiques aux implémentations faites par chaque post-processeur.

Configuration

Nous vous invitons à apporter la plus grande attention aux explications de ce chapitre.

Les fichiers de configuration

Quatre éléments de configuration sont utilisés par chaque post-processeur. Chacun d'eux est utilisé de façon spécifique.

Système

Contient la configuration matérielle de la cible, la configuration logicielle, des options permettant de modifier la façon dont le post-processeur génère le code ainsi que des déclarations de variables réservées (pour l'usage interne du post-processeur). Généralement, vous serez amenés, suivant la cible, à modifier la configuration matérielle contenue dans cet élément (par exemple un type d'UC ou une configuration de type de cartes d'entrées / sorties).

Correspondances de variables

La maîtrise de la correspondance des variables est un des éléments essentiels à la maîtrise de l'utilisation des post-processeurs.

Lorsque le post-processeur traduit un fichier du langage pivot d'AUTOMGEN vers un langage cible spécifique, il doit attribuer les variables d'AUTOMGEN à des variables de la cible.

Cet élément contient la description précise de l'attribution des variables. En modifiant cet élément, vous avez le contrôle total sur l'utilisation de l'espace des variables de la cible.

Code constructeur démarrage

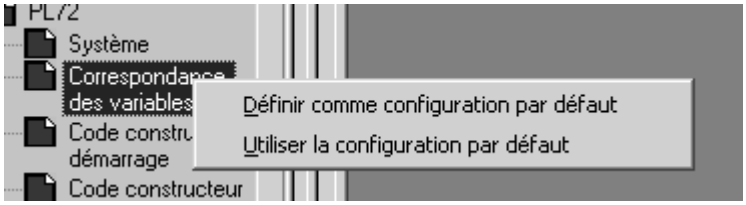
Cet élément contient du langage machine propre à la cible qui sera placé au début du code exécutable généré par le post-processeur (exécuté au début du cycle).

Code constructeur fin

Cet élément contient du langage machine propre à la cible qui sera placé à la fin du code exécutable généré par le post-processeur (exécuté à la fin du cycle).

Configuration par défaut

A la création du projet, des éléments de configuration par défaut sont dupliqués dans le projet. Les modifications apportées aux éléments de configuration du projet n'affecteront pas les déclarations par défaut.



Modifier les déclarations par défaut

Cliquez avec le bouton droit de la souris sur l'élément « Configuration / Post-processeurs / <nom de la cible> / … » et choisissez « Définir comme configuration par défaut ». L'élément de configuration du projet est alors défini comme configuration par défaut (écrasement de la configuration par défaut).

Attention, cette opération est irréversible. Seule la réinstallation du post-processeur permet de restaurer l'élément de configuration.

Utiliser les déclarations par défaut.

Cliquez avec le bouton droit de la souris sur l'élément « Configuration / Post-processeurs / <nom de la cible> / … » et choisissez « Utiliser la configuration par défaut ». L'élément de configuration du projet en cours est écrasé par la configuration par défaut.

Visualiser et modifier les éléments de configuration

Vous accédez à ces fichiers de configurations en double cliquant sur l'élément « Configuration / Post-processeurs / <nom de la cible> / … ». S'ouvre alors une fenêtre permettant de visualiser et de modifier l'élément de configuration.

Système

Cet élément de configuration est très spécifique à chaque post-processeur.

Configuration matérielle

Cette zone (optionnelle) doit être modifiée pour déterminer la configuration matérielle d'une cible (type de CPU, cartes d'entrées / sorties par exemple).

Configuration logicielle

Cette zone (optionnelle) doit être modifiée pour déterminer des caractéristiques propres à la configuration de l'application (la valeur du chien de garde par exemple).

Options de génération de code

Cette zone contient des réglages concernant la méthode de traduction que doit utiliser le post-processeur (réservés aux spécialistes). Le nombre d'options peut différer d'un post-processeur à l'autre. Ci dessous se trouve la liste des options communes à tous les post-processeurs :

« Optimiser le code généré »

Généralement réglé sur « Oui ». Le réglage sur « Non » peut permettre une analyse plus aisée du code généré.

« Ne pas générer le code d'évolution des étapes Grafcet »

Réglé par défaut sur « Non ». Si réglé sur « Oui », vous devez écrire dans l'élément « Code constructeur de fin » les instructions permettant la recopie des états immédiats de variables booléennes vers les états passés (voir le chapitre Gestion des variables booléennes d'AUTOMGEN)

« Ne pas générer le code d'évolution des bits utilisateurs »

Identique à l'option précédente mais appliquée aux bits utilisateurs (variables « U ») d'AUTOMGEN.

Déclarations de variables

Ce sont des déclarations de variables utilisées en interne par le post-processeur. Seuls des spécialistes peuvent avoir à modifier cet élément.

Autres éléments

D'autres éléments spécifiques à chaque post-processeur peuvent exister.

Voir l'élément « Système » sous forme de textes

En cliquant sur l'icône dans la barre d'outils, vous basculez du mode « arborescence » au mode « texte » (format des anciennes versions d'AUTOMGEN). Dans le format « Texte » vous pouvez copier et coller des informations entre les fichiers de configuration.

Les modifications en mode « texte » doivent être réservées aux spécialistes, toute modification intempestive peut entraîner des erreurs de compilation difficiles à localiser pour un néophyte.

Afficher les éléments système

En double cliquant sur « Configuration / Post-processeurs / <nom du post-processeur> / Système » vous ouvrez une fenêtre qui à l'aspect suivant :

Eléments	Valeurs	Commentaires
⊟ Configuration matérielle		
Type de l'automate	10	RPX 10
Module d'extension présent	NO	
Présence batterie		
⊟ Configuration logicielle		
Code compatible RPX C3	NO	
Chien de garde en ms		
⊟ Options de génération de code (attention, modifier avec précaution)		
Optimiser le code généré	Oui	
Ne pas générer le code d'évolution des étapes Grafcet	Non	
Ne pas générer le code d'évolution des bits utilisateur	Non	
Utiliser un seul bit automate pour chaque bit utilisateur AUTOMGEN	Non	
⊟ Déclaration de variables		
⊞ Affectation unitaire (une variable AUTOMGEN à une variable automate)		
Affectation linéaire (une table de variables AUTOMGEN à une table de variables automate)		
Affectation automatique (un ou plusieurs types de variables AUTOMGEN à une table de variables automate)		

Exemple de configuration système

Correspondances de variables

La maîtrise de la correspondance des variables est un des éléments essentiel à la maîtrise de l'utilisation des post-processeurs.

Lorsque le post-processeur traduit un fichier du langage pivot d'AUTOMGEN vers un langage cible spécifique, il doit attribuer les variables d'AUTOMGEN à des variables de la cible.

Par exemple, si vous avez utilisé le mot 200 d'AUTOMGEN dans votre application (il s'appelle M200 ou %MW200 dans AUTOMGEN) ce mot doit avoir une existence dans la mémoire de la cible et donc être repéré par un nom propre à cette cible.

AUTOMGEN propose trois types de déclaration de correspondance de variables :

- affectation unitaire,
- affectation linéaire,
- affectation automatique.

La correspondance des variables pour un projet sera constituée de « n » affectations utilisant chacune un de ces trois types.

L'affectation unitaire

Elle permet d'associer une variable AUTOMGEN à une variable de la cible. Elle est la plus simple des déclarations.

Elle doit être utilisée uniquement si une seule déclaration est nécessaire.

Cette déclaration utilise deux informations : le nom de la variable AUTOMGEN et le nom de la variable de la cible.

« Associe cette variable d'AUTOMGEN à cette variable de la cible », ainsi peut être résumée l'affectation unitaire.

L'affectation linéaire

C'est une forme plus évoluée de l'affectation unitaire.

Elle permet d'associer une série de variables consécutives (plusieurs variables de même type dont les numéros se suivent) d'AUTOMGEN à une série de variables consécutives de la cible.

Cette affectation est typiquement utilisée pour :

- la déclaration des variables d'entrées / sorties,
- la déclaration de tables de bits ou de mots devant avoir une adresse fixe (pour un lien avec un pupitre opérateur par exemple).

Cette déclaration nécessite trois informations : le nom de la première variable AUTOMGEN, le nom de la première variable de la cible et la dimension de la table en nombre de variables.

« Associe dans l'ordre cette table de variables AUTOMGEN à cette table de variables de la cible », ainsi peut être résumé l'affectation linéaire.

L'affectation automatique

C'est le type de déclaration le plus complexe et le plus puissant. Il permet d'associer un ou plusieurs types de variables AUTOMGEN à une plage de variables de la cible.

Cette affectation laisse le soin au compilateur de trouver une affectation à chaque variable présente dans le code généré (sous réserve quelle corresponde à l'un des types) de la déclaration.

Ce type de déclaration est typiquement utilisé pour toutes les variables de l'application AUTOMGEN dont l'adresse de la variable associée dans la cible n'a pas besoin d'être précisément fixée.

Cette déclaration nécessite trois informations :

- les types de variables AUTOMGEN (voir le chapitre Types de variables AUTOMGEN),
- le nom de la première variable de la plage de la cible,
- le numéro de la dernière variable (incluse) de la plage de la cible.

L'affectation automatique n'est utilisée par un post-processeur que si aucune autre déclaration n'a été trouvée pour une variable. Si par exemple une directive d'affectation linéaire défini l'attribution pour les mots 200 à 210 d'AUTOMGEN, alors le post-processeur n'utilisera pas l'affectation automatique pour essayer d'allouer ces mots.

Si plusieurs affectations automatiques existent pour un même type de variable AUTOMGEN, alors le post-processeur utilise la première plage

de variables de la cible jusqu'à saturation puis la seconde jusqu'à saturation, puis la troisième, etc…

Si au terme de l'utilisation de toutes les affectations automatiques une variable ne peut être allouée, alors un message d'erreur est généré par le compilateur indiquant que la variable n'est pas définie.

« Lorsque tu rencontres un de ces types de variables, utilises une variable de la cible de cette zone », ainsi peut être résumé l'affectation automatique.

Types de variables AUTOMGEN

Utilisés pour déclarer les correspondances de variables, ils sont un sur ensemble (car plus d'une variable de la cible peut être nécessaire pour loger une variable d'AUTOMGEN) des types de variables AUTOMGEN.

Gestion des variables booléennes d'AUTOMGEN

Un des principes de base de la traduction des langages booléens par le compilateur AUTOMGEN est de pouvoir accéder à deux états pour une même variable booléenne.

Ce concept fait référence à la notion de « cycle d'exécution » : entité représentant l'action réalisée par la cible consistant à lire les instructions de l'application de façon linéaire (du début jusqu'à la fin) et à accomplir les traitements qui leurs correspondent.

Ces deux états sont définis comme suit :

1- L'état immédiat de la variable : l'état écrit par la dernière instruction exécutée par la cible se reportant à cette variable, ou, à défaut celui qu'avait la variable à la fin du dernier cycle d'exécution, ou à défaut, si c'est le premier cycle d'exécution l'état d'initialisation de la variable.

2- L'état passé de la variable : l'état qu'avait la variable à la fin du dernier cycle d'exécution.

Remarques : ces deux états n'ont de validité que pour la tâche principale de l'application. Seul l'état immédiat a un sens pour les tâches asynchrones.

Le code généré par le compilateur AUTOMGEN assume ce qui suit :

- une affectation de variable booléenne se fait sur son état immédiat,
- un test de variable booléenne se fait sur son état passé.

Ces deux règles permettent de garantir une cohérence d'évolution des applications booléennes et notamment le respect des règles d'évolution des programmes générés par une description en langage Grafcet.

Le code généré par le post-processeur gère la recopie des états immédiats de variables vers les états passés en fin de cycle.

Lorsqu'une variable booléenne est utilisée dans AUTOMGEN deux variables booléennes sont utilisées sur la cible.
Trois exceptions existent :

1- pour une entrée tout ou rien, si aucun test de front n'est utilisé, seul l'état passé (« bi ») est utilisé (économie d'une variable booléenne),
2- pour une sortie tout ou rien, si aucun test de front n'est utilisé seul l'état immédiat (« o ») est utilisé.

(ceci explique pourquoi seules les variables « bi » et « o » se trouvent dans les directives d'attribution de variables).

3- pour le post-processeur ZELIO, compte tenu de la gestion temporelle des variables (quasi identique à celle d'AUTOMGEN) seuls les états immédiats sont utilisés dans le programme en langage ZELIO.

Syntaxe des éléments standards

« <nom de variable AUTOMGEN> » fait référence à l'état immédiat d'une variable booléenne ou à une variable numérique.

« b<nom de variable AUTOMGEN> » fait référence à l'état passé d'une variable booléenne.

Syntaxes spéciales pour les fronts

« u<nom de variable AUTOMGEN> » fait référence à l'état « front montant » d'une variable booléenne.

« d<nom de variable AUTOMGEN> » fait référence à l'état « front descendant » d'une variable booléenne.

Syntaxes spéciales pour les temporisations

« tempo <numéro> » fait référence au numéro d'une temporisation.

« tconsi<numéro> » fait référence à la consigne d'une temporisation.

« tcompt<numéro> » fait référence au compteur de temps d'une temporisation.

Autres syntaxes spéciales

(réservées aux spécialistes)

« ac » fait référence à l'accumulateur 16 bits.

« al » fait référence à l'accumulateur 32 bits.

« af » fait référence à l'accumulateur flottant.

« cf » fait référence au drapeau de retenue.

« zf » fait référence au drapeau de résultat nul.

« sf » fait référence au drapeau de résultat négatif.

« of » fait référence au drapeau de débordement.

Afficher les éléments de correspondances de variables

En double cliquant sur « Configuration / Post-processeurs / <nom du post-processeur> / Correspondance des variables » vous ouvrez une fenêtre qui a l'aspect suivant :

Eléments	Valeurs	Commentaires
Déclaration de variables		
Affectation unitaire (une variable AUTOMGEN à une variable automate)		
Affectation linéaire (une table de variables AUTOMGEN à une table de variables automate)		
<-32-> bi0	i0,0	
<-16-> o0	o0,0	
<-32-> tempo	0	
<-15-> c0	c0	
Affectation automatique (un ou plusieurs types de variables AUTOMGEN à une table de variables automate)		
i&bo&x&bx&bb&bu&b&u&t&bt&uxi	b2:255	
i&bo&x&bx&bb&bu&b&u&t&bt&uxi	x1:62	

Exemple de correspondances de variables

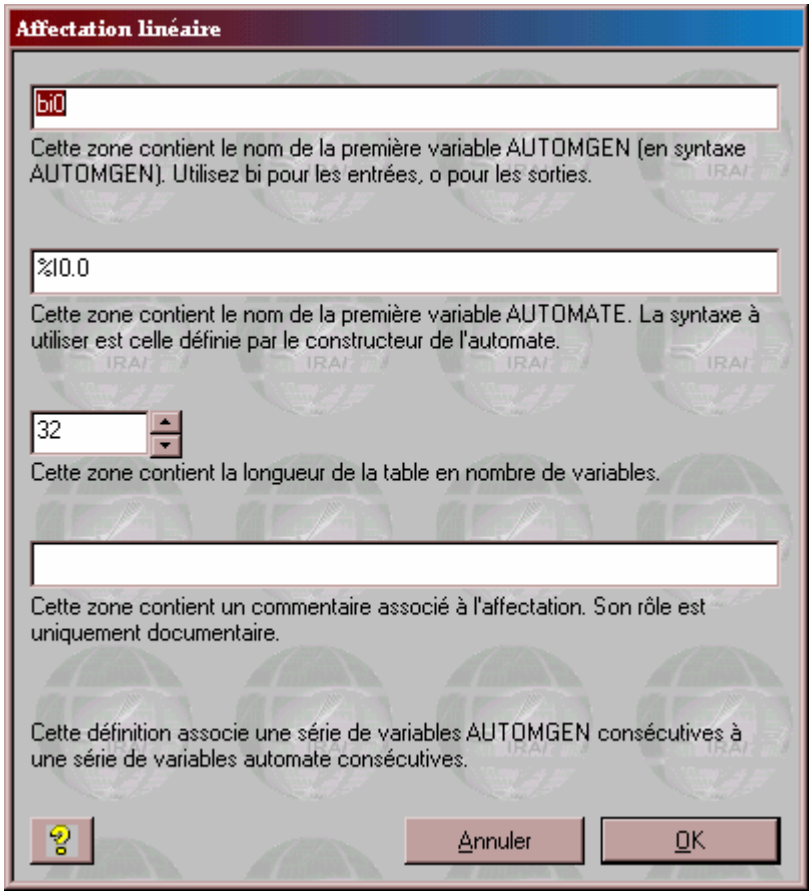
Remarque : dans le cas où un même post-processeur peut générer du code pour plusieurs types de cibles (plusieurs types de CPUs d'automate par exemple) les différents éléments peuvent être conditionnés pour l'ensemble des types de cibles ou pour un type de cible en particulier. Si les éléments sont conditionnés, ils sont rattachés à des lignes « Seulement pour xxx ». Voir exemple ci après.

Eléments	Valeurs	Commentaires
Déclaration de variables		
Affectation unitaire (une variable AUTOMGEN à une variable automate)		
Affectation linéaire (une table de variables AUTOMGEN à une table de variables automate)		
Seulement pour 1720		
<-32-> bi0	i0,0	
<-16-> o0	o0,0	
Seulement pour 47		
Seulement pour 4720		
Seulement pour 27		
<-32-> tempo	0	
Affectation automatique (un ou plusieurs types de variables AUTOMGEN à une table de variables automate)		
i&bo&x&bx&bb&bu&b&u&t&bt&uxi	b0:255	
i&bo&x&bx&bb&bu&b&u&t&bt&uxi	x0:95	
m&c	w1:1023	

En cliquant sur les éléments « + » dans l'arborescence, vous développez les branches, « - » les referme.

Modifier un élément de correspondance de variables

En double cliquant sur chacun des éléments, vous pouvez les modifier.

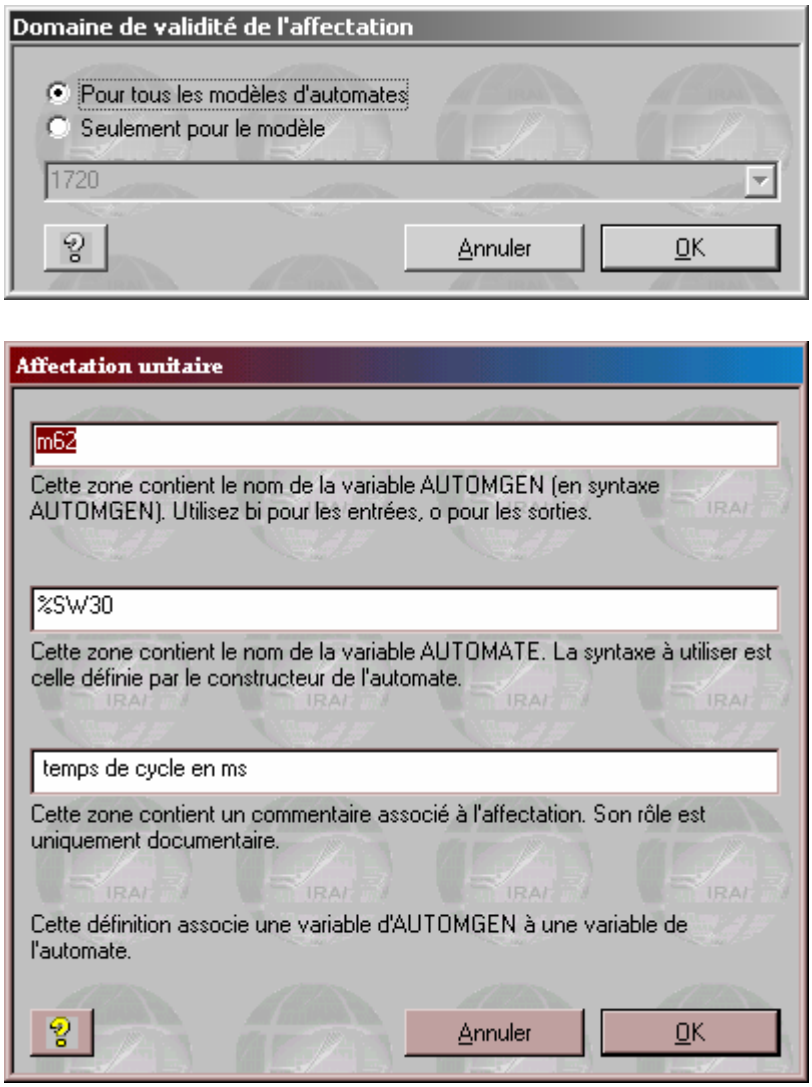


Exemple de boîte de dialogue de configuration d'une affectation linéaire.

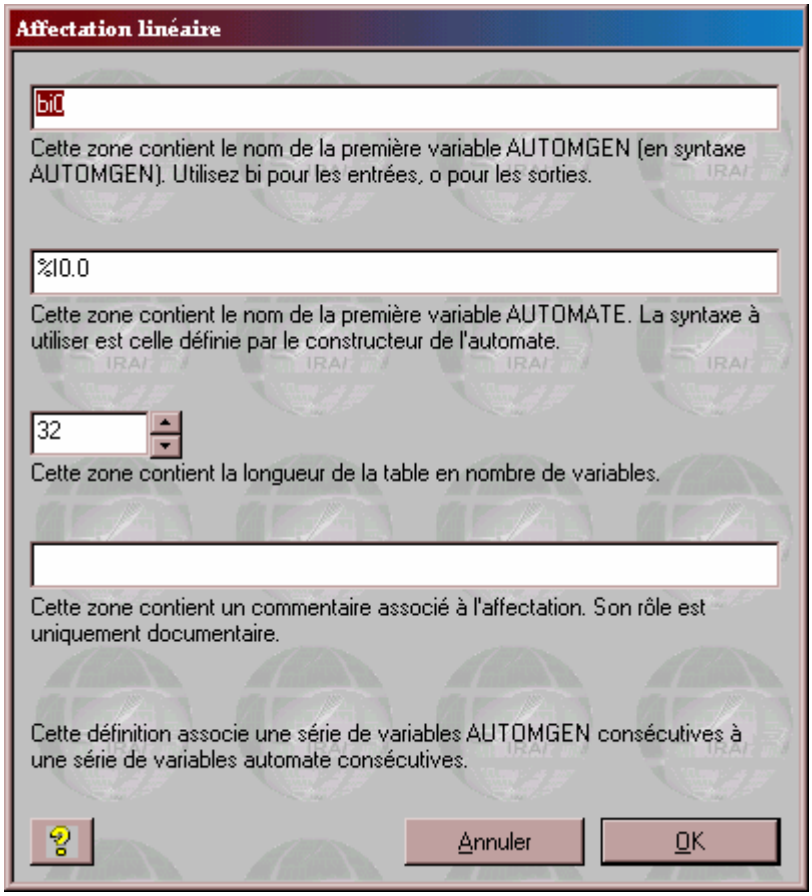
Ajouter un élément de correspondance de variables

En cliquant avec le bouton droit de la souris sur les éléments « Affectation … » de l'arborescence et en choisissant « Ajouter » dans le menu , vous pouvez ajouter une nouvelle affectation.

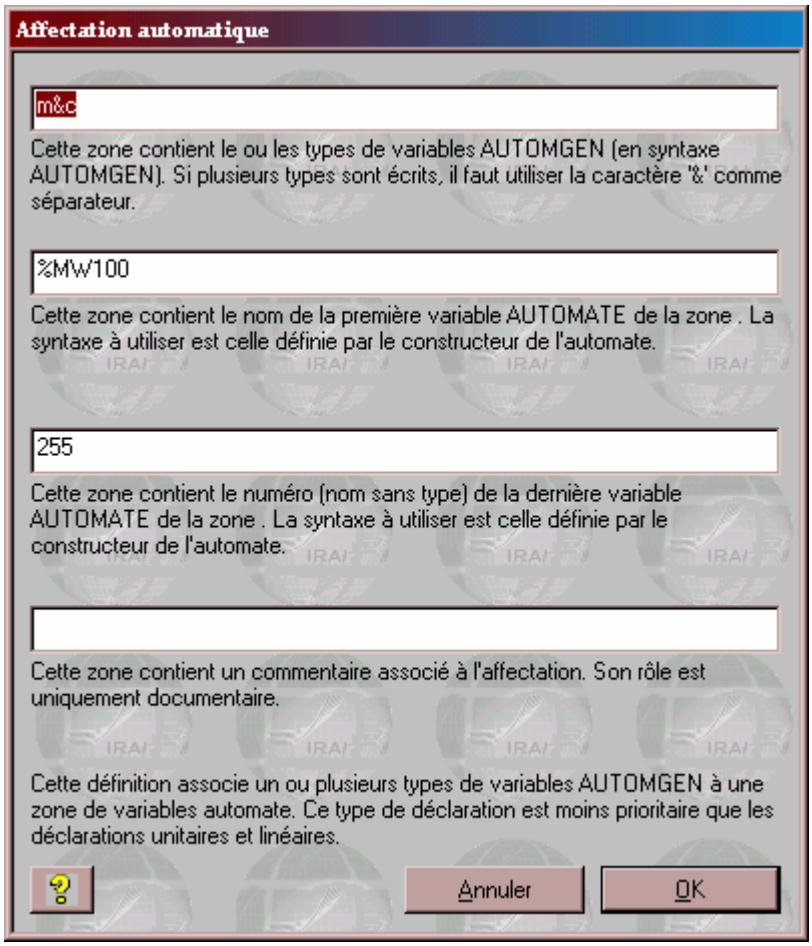
Si plusieurs types de cibles sont gérés par le post-processeur, la boîte de dialogue suivante permet de déterminer si la nouvelle affectation est seulement pour un type en particulier ou pour tous les types.



Affectation unitaire



Affectation linéaire



Affectation automatique

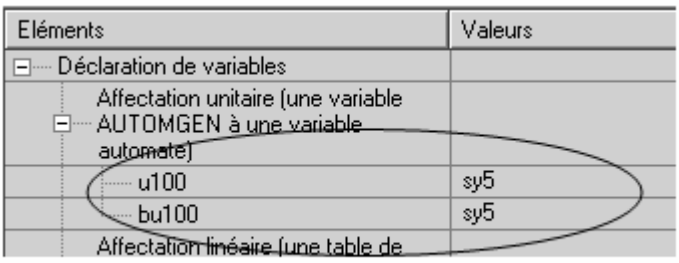
Notez que les types de variables AUTOMGEN doivent être séparés en utilisant le caractère « & ».

Supprimer un élément de correspondances de variables

Cliquez avec le bouton droit de la souris sur l'élément de correspondance de variables et choisissez « Supprimer » dans le menu.

Associer un bit d'AUTOMGEN à un bit système d'une cible

Deux déclarations sont nécessaires, il faut associer les deux variables d'état d'un bit « U » (« u » et « bu ») au bit système de la cible. Vous devez créer deux affectations unitaires, par exemple :



Attention, en affectant la variable u et bu d'AUTOMGEN au même bit système de la cible, vous annulez la possibilité de réaliser un test de front montant ou descendant dans l'application. Vous pouvez contourner ce problème en utilisant la syntaxe « ↑(u<n>) » ou « ↓(u<n>) » (avec « <n> » représentant le numéro du bit) dans l'application (cette syntaxe génère un bit intermédiaire sur lequel le front sera évalué correctement).

Associer une table de mots d'AUTOMGEN à une table de mots fixes de la cible

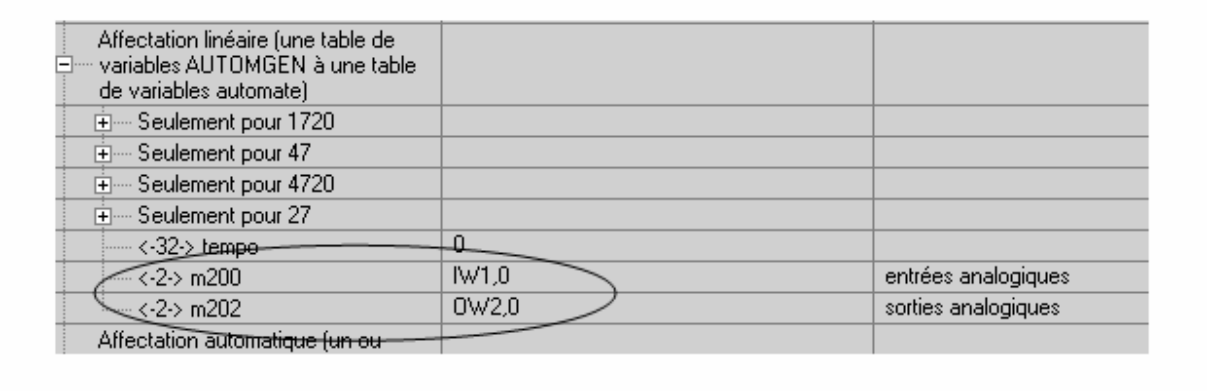
Pour ceci, une seule déclaration linaire suffit, par exemple :

Affectation linéaire (une table de variables AUTOMGEN à une table de variables automate)	
<-64-> bi0	e0.0
<-64-> o0	a0.0
Seulement pour 212	
Seulement pour 214	
Seulement pour 215	
Seulement pour 216	
Seulement pour 221	
Seulement pour 222	
Seulement pour 224	
<-10-> m200	VW128

Les mots de la cible ainsi alloués doivent être enlevés d'éventuelles autres affectations sous peine d'affecter deux fois de mêmes variables de la cible à plusieurs variables AUTOMGEN différentes.

Associer des mots d'AUTOMGEN à des entrées ou des sorties analogiques d'une cible

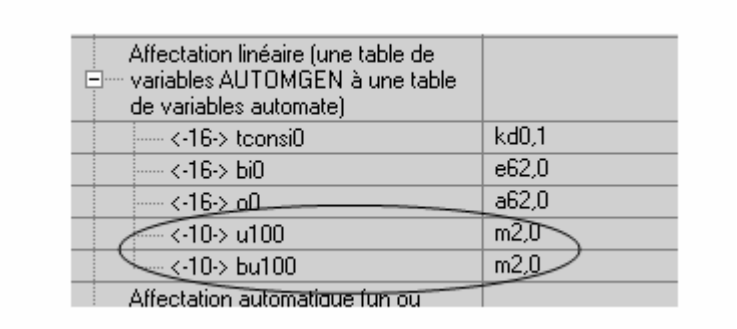
Utilisez des déclarations linéaires, par exemple :



Affectation linéaire (une table de variables AUTOMGEN à une table de variables automate)		
Seulement pour 1720		
Seulement pour 47		
Seulement pour 4720		
Seulement pour 27		
<-32-> tempo	0	
<-2-> m200	IW1,0	entrées analogiques
<-2-> m202	OW2,0	sorties analogiques
Affectation automatique (un ou		

Associer une table de bits d'AUTOMGEN à une table de bits d'une cible

Deux affectations linéaires sont nécessaires. Par exemple :



Affectation linéaire (une table de variables AUTOMGEN à une table de variables automate)	
<-16-> tconsi0	kd0,1
<-16-> bi0	e62,0
<-16-> o0	a62,0
<-10-> u100	m2,0
<-10-> bu100	m2,0
Affectation automatique (un ou	

Cet exemple (les états immédiats et les états passés sont associés aux mêmes variables de la cible) interdit l'utilisation de tests de front sur les bits AUTOMGEN u100 à u109.

Pour contourner le problème deux solutions sont possibles :

- utiliser la syntaxe « ↑(u<n>) » ou « ↓(u<n>) » (avec « <n> » représentant le numéro du bit) dans l'application,
- associer les bits états immédiats et passés à deux tables de bits différents dans la cible. Dans ce cas, les accès extérieurs à l'application qui pourraient être réalisés sur ces bits (par un terminal de dialogue ou un logiciel de supervision par exemple) doivent respecter la philosophie d'AUTOMGEN : accès en lecture sur les états passés, accès en écriture sur les états immédiats (un

accès en lecture sur les états immédiats est dans la pratique possible).

Les bits de la cible ainsi alloués doivent être enlevés d'éventuelles autres affectations sous peine d'affecter deux fois de mêmes variables de la cible à plusieurs variables AUTOMGEN différentes.

Voir les correspondances de variables sous forme de textes

En cliquant sur l'icône dans la barre d'outils, vous basculez du mode « arborescence » au mode « texte » (format des anciennes versions d'AUTOMGEN). Dans le format « Texte » vous pouvez copier et coller des informations entre les fichiers de configuration.

Les modifications en mode « texte » doivent être réservées aux spécialistes, toute modification intempestive peut entraîner des erreurs de compilation difficiles à localiser pour un néophyte.

Code constructeur démarrage, code constructeur de fin

Ces éléments de configuration contiennent du code machine propre à chaque cible sous forme textuel.

La syntaxe à utiliser dans ces sections est proche des langages de bas niveau utilisables sur chaque cibles. L'observation du code généré en passe 1 par chaque post-processeur vous permet de visualiser la syntaxe à utiliser.

Référence à une variable AUTOMGEN

Utilisez la syntaxe « _<nom de variable AUTOMGEN>_ » pour faire référence à une variable AUTOMGEN (pensez à ajouter le caractère « b » en tête de variable pour accéder à l'état passé d'une variable booléenne. Par exemple « _bu100_ »).

Référence à un symbole de l'application AUTOMGEN

Syntaxe :
|nom du symbole|
Le caractère « | » est généralement associé à la touche 6 du clavier.

Définition et référence à un label

« @<nom de label> » marque une destination de saut,
« _*<nom de label>*_ » fait référence au label.

Insérer du code constructeur dans une application

Les mots clés « #BEGIN_MACHINE_CODE » et « #END_MACHINE_CODE » permettent d'insérer du code constructeur dans une boîte de code AUTOMGEN.

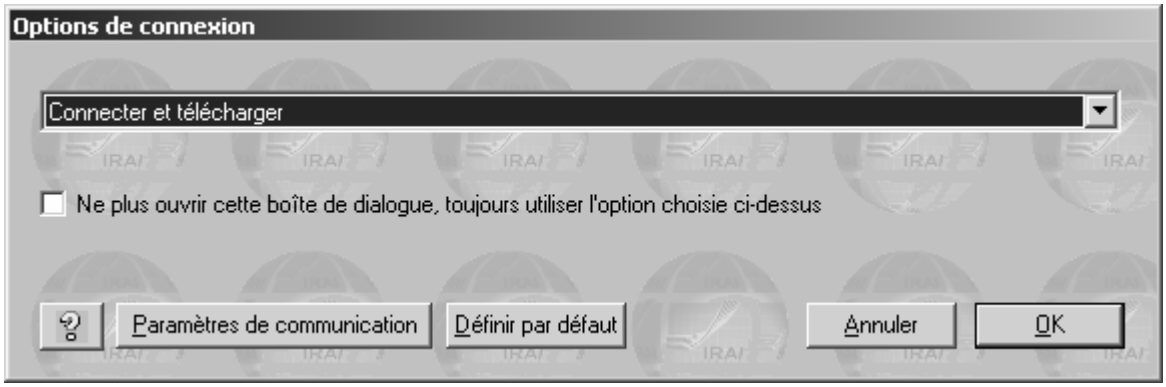
Ces deux directives doivent se trouver en début d'une ligne, aucun autre caractère ne doit se trouver sur la même ligne.

Les lignes se trouvant entre ces deux directives définissent une zone nommée « Section de langage constructeur ».

La syntaxe à utiliser dans une section de langage constructeur est la même que celle utilisée dans les éléments « Code de démarrage » et « Code de fin ».

Choix des options de connexion

Double cliquez sur l'élément « Configuration / Post-processeur / <nom du post-processeur> / Options de connexion ».



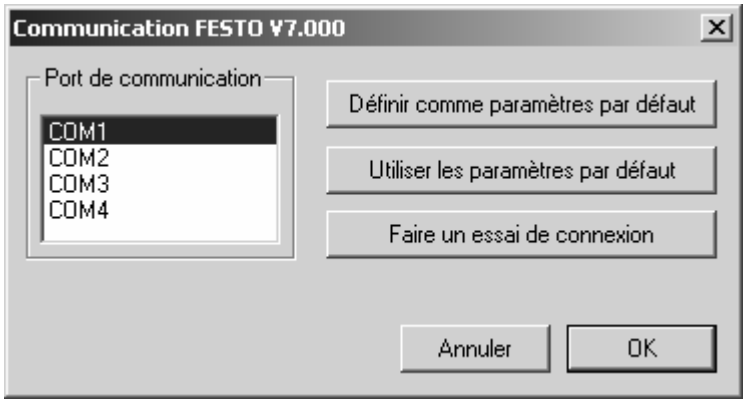
Choix d'un mode de connexion

Les modes de connexions possibles sont plus ou moins nombreux en fonction du post-processeur. Le mode « Seulement connecté » est typiquement utilisé pour créer une application de supervision.

Cette boîte de dialogue s'ouvre automatiquement lorsqu'une connexion à une cible est demandée. En cochant la case « Ne plus ouvrir … », cette ouverture n'est plus automatique. Pour l'ouvrir de nouveau, laissez enfoncée la touche [Shift] du clavier en lançant la commande de connexion ou la commande « Go ».

Paramétrage du module de communication

Double cliquez sur l'élément « Configuration / Post-processeur / <nom du post-processeur> / Module de communication ».



Exemple de paramétrage d'un module de communication.

La configuration en cours peut être définie comme configuration par défaut (pour les nouveaux projets) ou rétablie par défaut.

Un essai de connexion peut être réalisé.

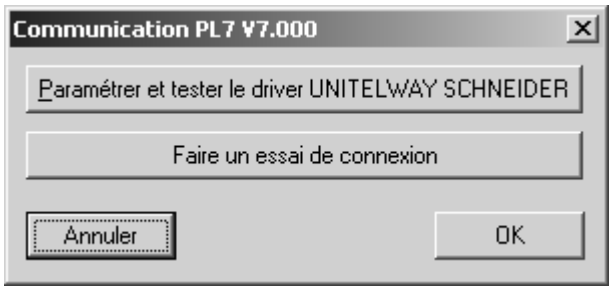
Post-processeur PL7

Ce post-processeur permet de programmer les automates MODICON TELEMECANIQUE SCHNEIDER TSX 37, (TSX MICRO) et TSX 57 (TSX PREMIUM).

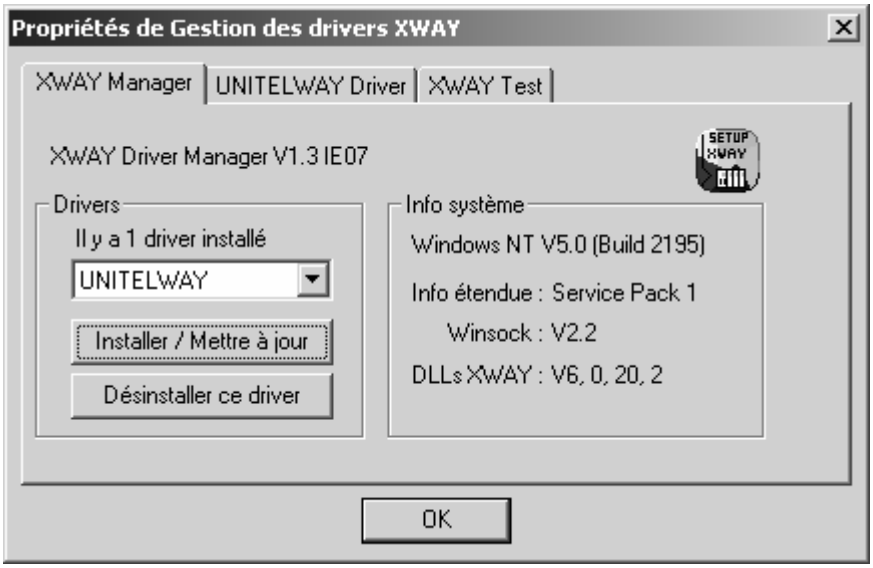
Module de communication

Le driver UNITELWAY SCHNEIDER doit impérativement être installé sur l'ordinateur (en local) pour pouvoir communiquer avec les automates SCHNEIDER TSX 37 et TSX 57. Des drivers adaptés à une ou plusieurs versions de WINDOWS se trouvent sur le CD-ROM et peuvent être téléchargés sur le site d'IRAI : www.irai.com.

Le module de communication utilise le driver de communication conçu par SCHNEIDER AUTOMATION. Cliquer sur « Paramétrer et tester … » vous permet d'accéder directement au menu du driver de communication SCHNEIDER.



Paramétrage du module de communication



Propriétés du module de communication UNITELWAY

Mode de génération d'un fichier exécutable

Le post-processeur peut générer soit un fichier binaire directement téléchargeable dans l'automate (disponible uniquement sur TSX 37, non disponible sur TSX 57), soit un fichier importable dans les outils SCHNEIDER (disponible pour TSX 37 et TSX 57). La première solution est nettement préférable (gain de temps, simplicité d'utilisation).

Le choix du mode s'effectue dans la configuration logicielle du post-processeur.

Mode de génération directe du fichier binaire

Ce mode est fortement recommandé pour TSX 37.

Il engendre les restrictions suivantes :

- pas d'utilisation d'instruction de sous-programmes,
- pas de support des cartouches mémoire,
- pas de support d'instructions spécifiques (communication ou PID par exemple).

Si votre application doit utiliser des éléments très spécifiques, utilisez une des méthodes d'importation décrite ci-après.

Eléments	Valeurs
⊞ Configuration matérielle	
⊟ Configuration logicielle	
Générer directement le fichier binaire sans passer par le logiciel SCHNEIDER	YES

Sélection de la génération automatique d'un fichier binaire.

Fichier de configuration de l'automate

Pour les automates TSX 37-05 et TSX 37-08 en version de base (avec une seule carte d'entrées / sorties), des fichiers 3705.stx et 3708.stx sont fournis en standard dans le répertoire d'installation d'AUTOMGEN.

Si le fichier de configuration n'est pas créé, alors les sorties TOR de l'automate ne seront pas activées.

Une fois le fichier créé ou téléchargé (voir ci-après), donnez le chemin d'accès au fichier dans l'élément de configuration suivant :

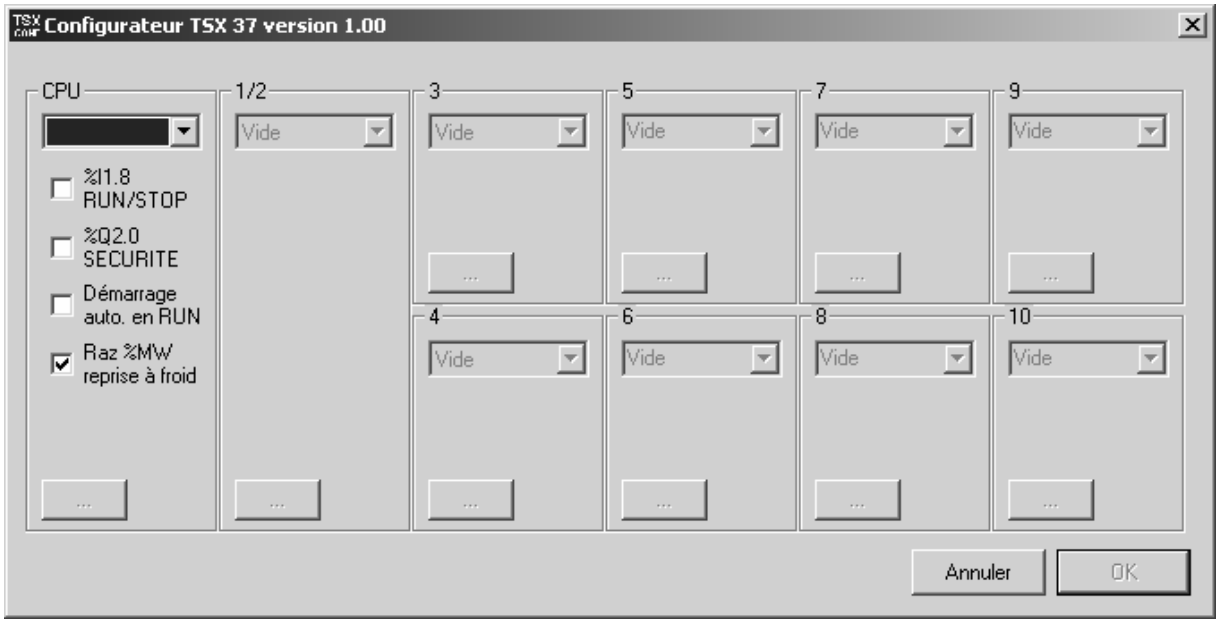
Configuration matérielle	
Fichier contenant la configuration matérielle de l'automate	<AUTOM7DIR>\3708.stx

Le nom du fichier contenant la configuration.

Quatre méthodes sont possibles pour obtenir un fichier de configuration :

Utiliser le configurateur fourni par IRAI

Double cliquez sur la ligne « Fichier contenant la configuration matérielle de l'automate » et répondez « Oui » à la question « Créer un nouveau fichier de configuration ». Le configurateur intégré à AUTOMGEN est alors automatiquement lancé :



Ce configurateur comporte certaines restrictions : pas de support de certaines cartes, pas de support des cartes de communications, pas de relecture de fichiers .STX existants.

Téléchargement du fichier de configuration sur le site IRAI

1- téléchargez un fichier correspondant à la configuration de votre automate sur le site d'IRAI : www.irai.com, rubrique « Téléchargement / AUTOMGEN7 / fichiers de configuration pour automate TSX 37 » (recommandé si le fichier de configuration est présent sur le site),

2- recopiez le fichier ainsi téléchargé sans le décompresser (les fichiers « .STX » sont des fichiers compressés) dans le répertoire d'installation d'AUTOMGEN ou insérez-le dans les ressources du projet AUTOMGEN.

Décompresser le fichier « .STX » empêchera son utilisation par le post-processeur.

Création d'un fichier avec un des outils de programmation SCHNEIDER

Les outils logiciel SCHNEDIER (PL7MICRO V3.1, PL7JUNIOR V3.1 ou PL7PRO V3.4) sont utilisables. Les fichiers créés avec d'autres versions peuvent ne pas fonctionner, dans ce cas, l'automate passe en mode erreur à l'issue du téléchargement de l'application (Voyant « ERR » allumé sur l'automate).

Pour créer le fichier « .STX » :

1- lancez un des outils SCHNEIDER, créer une application en suivant les règles suivantes :
   - choisissez le type de la CPU de votre automate en sélectionnant toujours la version 1.0 de la CPU,
   - sélectionnez la ou les cartes d'entrées / sorties installées sur votre automate et au besoin paramétrez les,

2- sauvegardez le fichier ainsi créé dans le répertoire d'installation d'AUTOMGEN ou intégrez-le dans les ressources du projet AUTOMGEN.

Communiquez la communication par email à IRAI pour obtenir un fichier de configuration

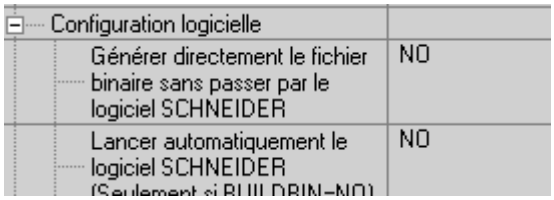
1- envoyez un email à IRAI en demandant un fichier de configuration en précisant :
   - le type de CPU TSX 37-05, 37-08, 37-10, 37-21 ou 3722,
   - la position et le type précis des cartes d'entrées / sorties (DMZ …).

2- à la réception du fichier par mail, recopiez le dans le répertoire d'installation d'AUTOMGEN (sans le décompresser) ou intégrez-le dans les ressources du projet AUTOMGEN.

Mode de génération d'un fichier « .FEF »

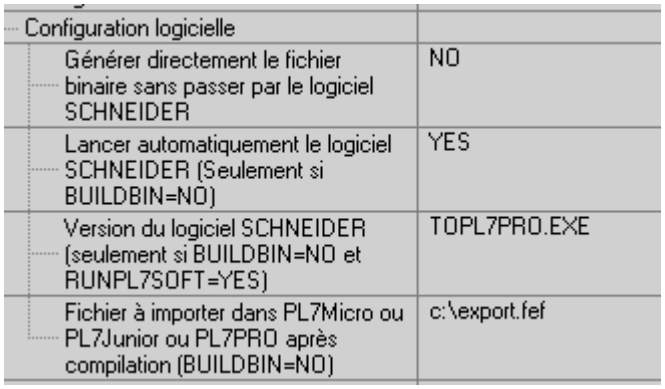
Dans ce mode, l'importation dans les outils de programmation SCHNEIDER (PL7 MICRO (TSX 37), PL7 JUNIOR (TSX 37 ou TSX 57) ou PL7 PRO (TSX 37 ou TSX 57) peut être automatique ou manuelle.

Importation manuelle



Sélection du mode d'import manuel

Vous devez choisir un nom de fichier qui sera exporté depuis AUTOMGEN :



Choix d'un fichier pour l'export vers l'atelier logiciel SCHNEIDER

Procédure :

1- Compilez l'application dans AUTOMGEN en utilisant la commande « Compile » du menu « Programme » ou en cliquant sur le bouton de la barre d'outils,

2- Lancez un atelier logiciel SCHNEIDER, créez un nouveau projet et utilisez la commande « Importer une application » du menu « Fichier »,

3- A la fin de l'importation, transférez l'application dans l'automate,

4- Pour obtenir la visualisation dynamique dans AUTOMGEN, cliquez sur le bouton « Go » de la barre d'outils et choisissez comme mode de connexion « Seulement connecter ».

Importation automatique

L'outil logiciel SCHNEIDER sera lancé automatiquement. Seul un nombre restreint de versions des logiciels SCHNEIDER sont utilisables. Le type et la version du logiciel SCHNEIDER doivent être réglé dans la configuration logicielle.

Configuration logicielle		
Générer directement le fichier binaire sans passer par le logiciel SCHNEIDER	NO	
Lancer automatiquement le logiciel SCHNEIDER (Seulement si BUILDBIN=NO)	YES	
Version du logiciel SCHNEIDER (seulement si BUILDBIN=NO et RUNPL7SOFT=YES)	TOPL7JU2.EXE	PL7 Junior version 3.1

Sélection du type et de la version du logiciel SCHNEIDER

Le fonctionnement de la procédure d'import automatique avec d'autres versions des logiciels SCHNEIDER n'est pas garanti.

Procédure à réaliser une seule fois :

1- Lancer un outil de programmation SCHNEIDER et créez une nouvelle application,

2- Configurez l'application : type d'automate, cartes d'entrées / sorties, etc…

3- Sauvegardez le fichier ainsi créé,

4- Donnez le chemin d'accès complet à ce fichier sous la rubrique « Configuration matérielle » de l'élément « Système», par exemple :

Configuration matérielle	
Fichier contenant la configuration matérielle de l'automate	c:\pl7user\automgen.stx

Procédure à réaliser à chaque exécution une application :

1- Lancez l'outil logiciel SCHNEIDER (s'il ne l'est pas déjà),

2- Cliquez sur le bouton « GO » dans la barre d'outils d'AUTOMGEN.

Utilisation de tâches d'interruptions

En définissant un folio de type tâche, vous pouvez insérer du langage bas niveau d'AUTOMGEN ou du langage constructeur à une tâche de l'automate. Le tableau ci-dessous donne la correspondance entre le numéro de tâche et le type de tâche d'interruption de l'automate.

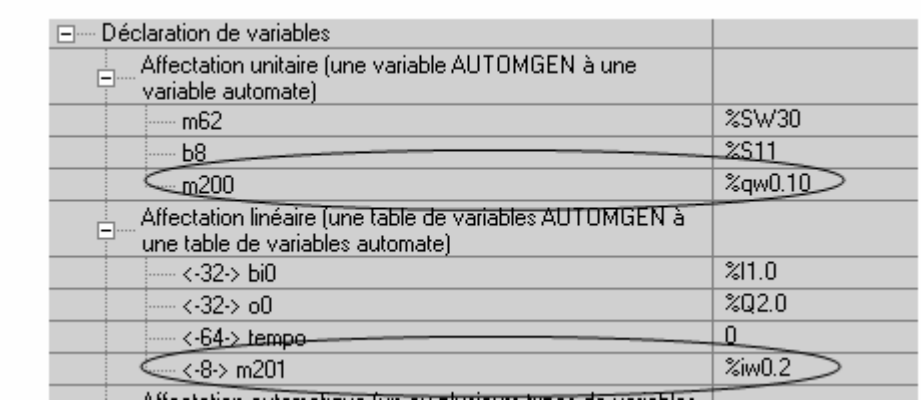
Numéro de tâche (folio AUTOMGEN)	Type de tâche automate TSX 37	Type de tâche automate TSX 57
0	Tâche rapide	Tâche rapide
1	EVT1	EVT0
2	EVT2	EVT1
3	EVT3	EVT2
etc…		

Exemples spécifiques

Ces exemples se trouvent dans le répertoire « <répertoire d'installation d'AUTOMGEN> /Exemples/Post-processeurs/PL7 ». Les fichiers portent le même nom que le titre des chapitres qui suivent.

Entrées / sorties analogiques

Cet exemple illustre l'utilisation des entrées / sorties analogiques.



Déclaration des entrées / sorties analogiques sur un automate TSX 37-22.

Compteur rapide TSX 37-10

Cet exemple illustre l'utilisation du compteur rapide sur un automate TSX 37-10.

Compteur rapide TSX 37-10 utilisé en décomptage

Cet exemple illustre l'utilisation du compteur rapide sur un automate TSX 37-10 en mode décomptage.

Compteur rapide TSX 37-22

Cet exemple illustre l'utilisation du compteur rapide sur un automate TSX 37-22.

ASI

Exemple d'utilisation d'entrées / sorties ASI

MAGELIS

Exemple d'utilisation d'un terminal MAGELIS

Post-processeur PL72

Ce post-processeur permet la programmation des automates TELEMECANIQUE SCHNEIDER TSX 17-20 (avec cartouche PL72), TSX 27, TSX 47 et TSX 47-20.

Choix du type de l'automate

L'élément « Configuration / Post-processeur / PL72 / Système / Configuration matérielle» du navigateur permet de choisir le type de l'automate.

Eléments syntaxiques spécifiques

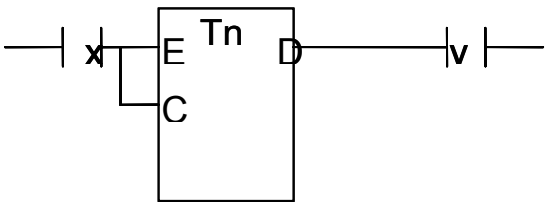
Appel des blocs fonction PL72

Les syntaxes suivantes permettent d'appeler les blocs temporisation, texte et compteur rapide (TSX 17-20) sous la forme textuelle utilisée dans les éléments « Code constructeur de démarrage », « Code constructeur de fin » et les sections en langage constructeur.

Bloc temporisation

x.Tn=y

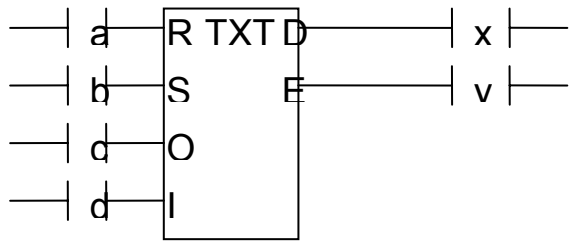
Equivalent PL72 :



Bloc texte

a+b+c+d.TXTn=x:y

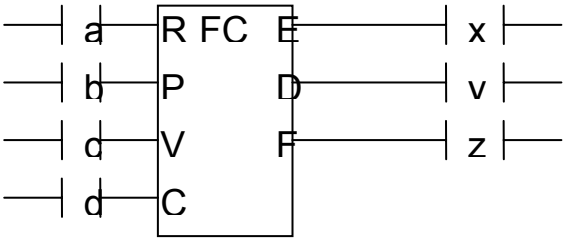
Equivalent PL72 :



Bloc compteur rapide

a+b+c+d.FC=x:y:z

Equivalent PL72 :



Bloc horodateur type WEEK (sur TSX 17-20 uniquement)

a.H,W,(jours),(heure de début),(heure de fin)=x :y :z

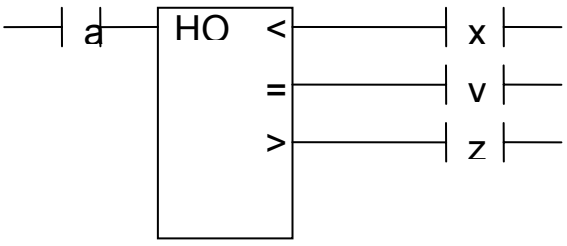
« jours » représente les jours de la semaine, c'est une valeur décimale codée sur 7 bits, chaque bit représente un jour de la semaine. Le jour est actif si le bit est à 1. b0 correspond à Dimanche et b6 à Samedi. Par exemple pour valider le lundi et le mercredi il faut écrire la valeur $2^1 + 2^3$: 2 + 8 = 10. Pour valider les 7 jours de la semaine : la valeur est 127.

« heure de début » et « heure de fin » sont exprimées sous la forme HH:MM : heures et minutes.

Bloc horodateur type YEAR (sur TSX 17-20 uniquement)
a.H,Y,(date de début),(date de fin)=x :y :z

« date de début » et « date de fin » sont exprimées sous la forme JJ/MM : jours et mois.

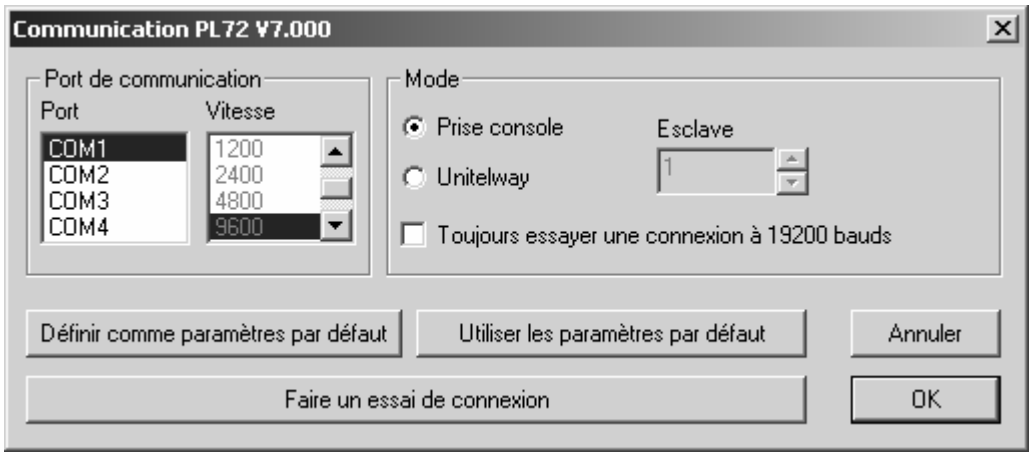
Equivalent PL72 :



Utilisation de la tâche rapide

Un folio de type « Tâche » portant le numéro « 1 » permet d'associer le code littéral ou le code PL72 écrit sur le folio à la tâche rapide. Le folio ne peut contenir que du code littéral bas niveau ou du code PL72 écrit dans un rectangle d'organigramme.

Module de communication



Paramétrage du module de communication

Si vous connectez le PC sur la prise console de l'automate sélectionnez impérativement « Prise console ».

Ne cochez « Toujours essayer une connexion à 19200 bauds » que si votre automate est un TSX 17-20 récent (cette option permet un dialogue plus rapide entre le PC et l'automate).

Le mode « UNITELWAY » permet de connecter le PC à un coupleur UNITELWAY. Dans ce cas, la vitesse doit être en accord avec la configuration du coupleur.

Si vous cochez « Toujours essayer une connexion à 19200 bauds » et que votre TSX 17-20 ne supporte pas la communication à 19200 bauds alors la connexion échouera.

Si le mode n'est pas en accord avec la connexion (mode UNITELWAY sélectionné et connexion sur ma prise console par exemple) alors la connexion échouera.

Exemples spécifiques

Ces exemples se trouvent dans le répertoire « <répertoire d'installation d'AUTOMGEN> / Exemples/ Post-processeurs / PL72 ». Les fichiers portent le même nom que le titre des chapitres qui suivent.

Entrées / sorties analogiques

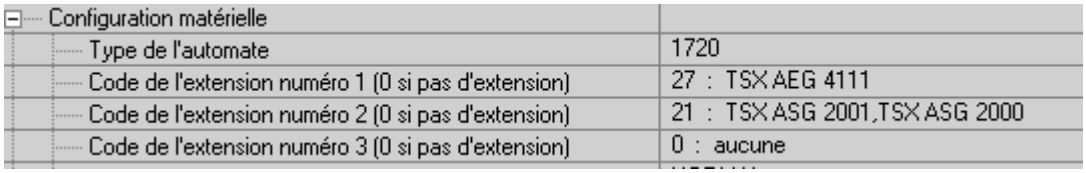
Pour pouvoir utiliser les sorties analogiques sur un automate TSX 17-20 il faut :

- déclarer le ou les blocs d'entrées / sorties analogiques dans l'élément « Système » de la configuration,
- associer une ou plusieurs variables AUTOMGEN aux mots d'entrées / sorties TELEMECANIQUE (IW et OW).

Exemple :

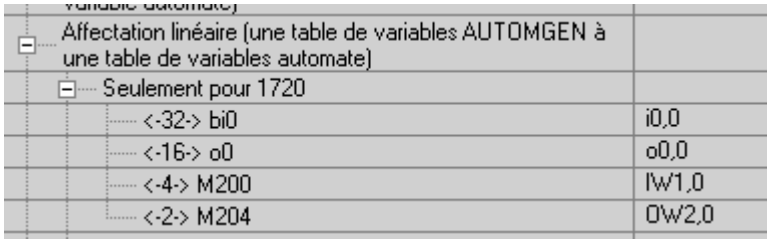
- automate TSX 17-20 utilisant un bloc de 4 entrées analogiques (code 27) en position 1 et un bloc de 2 sorties analogiques (code 21) en position 2.
- le programme recopiera simplement l'état de la première entrée analogique sur la première sortie analogique. Il comparera également la deuxième entrée analogique avec la valeur 500 (valeur arbitraire) et positionnera deux sorties booléennes : O0 si

l'entrée est inférieure à 500, O1 si l'entrée est supérieure ou égale à 500.



Configuration matérielle	
Type de l'automate	1720
Code de l'extension numéro 1 (0 si pas d'extension)	27 : TSX AEG 4111
Code de l'extension numéro 2 (0 si pas d'extension)	21 : TSX ASG 2001,TSX ASG 2000
Code de l'extension numéro 3 (0 si pas d'extension)	0 : aucune

La déclaration des deux modules d'extension



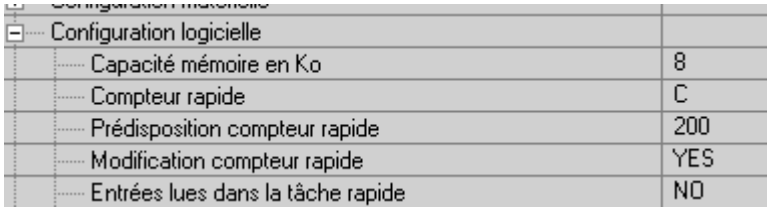
Affectation linéaire (une table de variables AUTOMGEN à une table de variables automate)	
Seulement pour 1720	
<-32-> bi0	i0,0
<-16-> o0	o0,0
<-4-> M200	IW1,0
<-2-> M204	OW2,0

L'affectation des variables

Ces deux déclarations associent les mots M200 à M203 d'AUTOMGEN aux variables IW1,0 à IW1,3 de l'automate ainsi que les variables M204 et M205 d'AUTOMGEN aux variables OW2,0 et OW2,1 de l'automate.

Compteur rapide

Le but est de compter 200 impulsions sur le compteur rapide. La valeur courante du compteur rapide sera recopiée dans le mot M200 d'AUTOMGEN. La sortie O5 sera activée par la tâche rapide en fin de comptage.



Configuration logicielle	
Capacité mémoire en Ko	8
Compteur rapide	C
Prédisposition compteur rapide	200
Modification compteur rapide	YES
Entrées lues dans la tâche rapide	NO

Le paramétrage du compteur rapide

Blocs texte et xbt

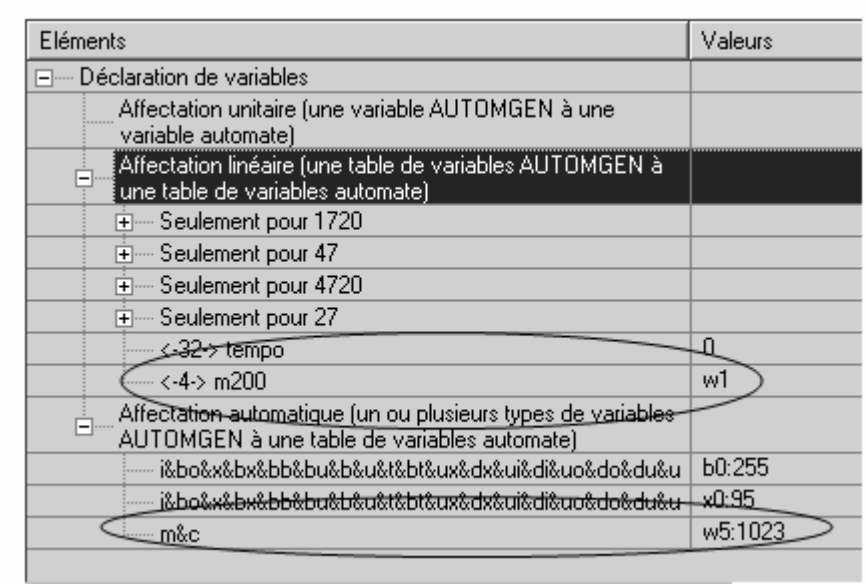
Le but est de dialoguer avec un XBT connecté sur le port console d'un automate TSX 17-20.

Les entrées I0 à I3 déclencheront l'affichage des messages numéro 0 à numéro 3 enregistrés dans l'XBT.

Le bloc texte TXT1 sera utilisé pour dialoguer sur le port console.

Le format du message à envoyer à l'XBT pour afficher un message est le suivant :
ESC V xxx LF CR

xxx représente le numéro de message codé en décimal sur trois caractères.



Allocation d'une table de mots pour les échanges

Blocs textes	
Type de bloc texte 0	
Type de bloc texte 1	TER
Type de bloc texte 2	
Type de bloc texte 3	
Type de bloc texte 4	
Type de bloc texte 5	
Type de bloc texte 6	
Type de bloc texte 7	
Adresse du buffer pour le bloc texte 0 (CW0 à CW127 ou W0 à W1023)	
Adresse du buffer pour le bloc texte 1 (CW0 à CW127 ou W0 à W1023)	W1[0]
Adresse du buffer pour le bloc texte 2 (CW0 à CW127 ou W0 à W1023)	
Adresse du buffer pour le bloc texte 3 (CW0 à CW127 ou W0 à W1023)	
Adresse du buffer pour le bloc texte 4 (CW0 à CW127 ou W0 à W1023)	
Adresse du buffer pour le bloc texte 5 (CW0 à CW127 ou W0 à W1023)	
Adresse du buffer pour le bloc texte 6 (CW0 à CW127 ou W0 à W1023)	
Adresse du buffer pour le bloc texte 7 (CW0 à CW127 ou W0 à W1023)	
TXT0,L	
TXT1,L	7
TXT2,L	
TXT3,L	
TXT4,L	
TXT5,L	
TXT6,L	
TXT7,L	
TXT0,C	
TXT1,C	$FD
TXT2,C	
TXT3,C	
TXT4,C	
TXT5,C	
TXT6,C	
TXT7,C	
TXT0,M	
TXT1,M	$00
TXT2,M	
TXT3,M	
TXT4,M	
TXT5,M	
TXT6,M	
TXT7,M	
TXT0,A	
TXT1,A	$FE
TXT2,A	
TXT3,A	
TXT4,A	
TXT5,A	
TXT6,A	
TXT7,A	
TXT0,T	
TXT1,T	0

Paramétrage du bloc texte

Blocs texte et UNITELWAY

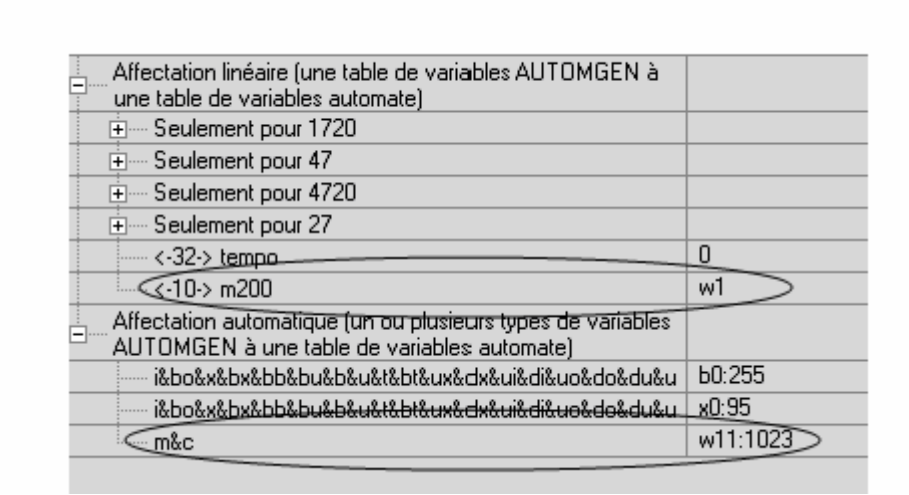
Le but est d'utiliser un coupleur UNITELWAY pour effectuer l'acquisition d'une table de 3 mots sur un l'automate cible. Le coupleur UNITELWAY est installé comme première extension, il sera configuré en maître pour utiliser deux esclaves. L'automate lu sera l'esclave numéro 1.

Eléments	Valeurs
Configuration matérielle	
Type de l'automate	1720
Code de l'extension numéro 1 (0 si pas d'extension)	63
Code de l'extension numéro 2 (0 si pas d'extension)	0
Code de l'extension numéro 3 (0 si pas d'extension)	0

Configuration du coupleur UNITELWAY

Blocs textes	
Type de bloc texte 0	
Type de bloc texte 1	CPL
Type de bloc texte 2	CPL
Type de bloc texte 3	
Type de bloc texte 4	
Type de bloc texte 5	
Type de bloc texte 6	
Type de bloc texte 7	
Adresse du buffer pour le bloc texte 0 (CW0 à CW127 ou W0 à W1023)	
Adresse du buffer pour le bloc texte 1 (CW0 à CW127 ou W0 à W1023)	CW0
Adresse du buffer pour le bloc texte 2 (CW0 à CW127 ou W0 à W1023)	W1[10]
Adresse du buffer pour le bloc texte 3 (CW0 à CW127 ou W0 à W1023)	
Adresse du buffer pour le bloc texte 4 (CW0 à CW127 ou W0 à W1023)	
Adresse du buffer pour le bloc texte 5 (CW0 à CW127 ou W0 à W1023)	
Adresse du buffer pour le bloc texte 6 (CW0 à CW127 ou W0 à W1023)	
Adresse du buffer pour le bloc texte 7 (CW0 à CW127 ou W0 à W1023)	
TXT0,L	
TXT1,L	10
TXT2,L	6
TXT3,L	
TXT4,L	
TXT5,L	
TXT6,L	
TXT7,L	
TXT0,C	
TXT1,C	$40
TXT2,C	$0736
TXT3,C	
TXT4,C	
TXT5,C	
TXT6,C	
TXT7,C	
TXT0,M	
TXT1,M	$100
TXT2,M	$165
TXT3,M	
TXT4,M	
TXT5,M	
TXT6,M	
TXT7,M	
TXT0,A	
TXT1,A	$FE
TXT2,A	$FE

Paramétrage des deux blocs texte



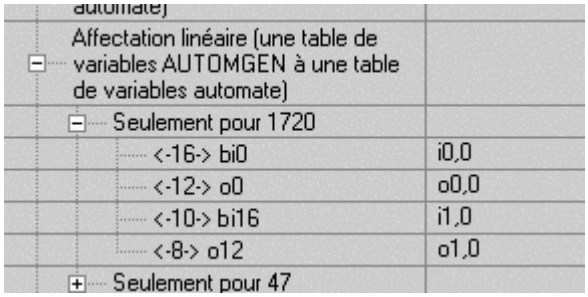
Attribution d'une table de mots pour les échanges

Module d'extension TOR

Cet exemple illustre la configuration d'un module d'extension TOR. Nous prenons comme hypothèses un module de base équipé de 16 entrées et de 12 sorties et un module d'extension équipé de 10 entrées et de 8 sorties.

Eléments	Valeurs	Commentaires
Configuration matérielle		
Type de l'automate	1720	automate TSX17-20 avec cartouche PL72
Code de l'extension numéro 1 (0 si pas d'extension)	15	TSX DMF 401,TSX DMF 400,TSX DMF 342A,TSX DMF 344A
Code de l'extension numéro 2 (0 si pas d'extension)	0	aucune
Code de l'extension numéro 3 (0 si pas d'extension)	0	aucune
Entrée I0.0	NORMAL	entrée normale

La définition du module d'extension



L'affectation des variables

Conversion

Montre comment appeler les fonctions de conversion du langage PL72.

Horodateur

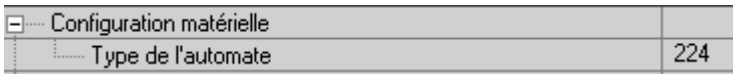
Exemple d'utilisation du bloc fonctionnel horodateur.

Post-processeur S7200

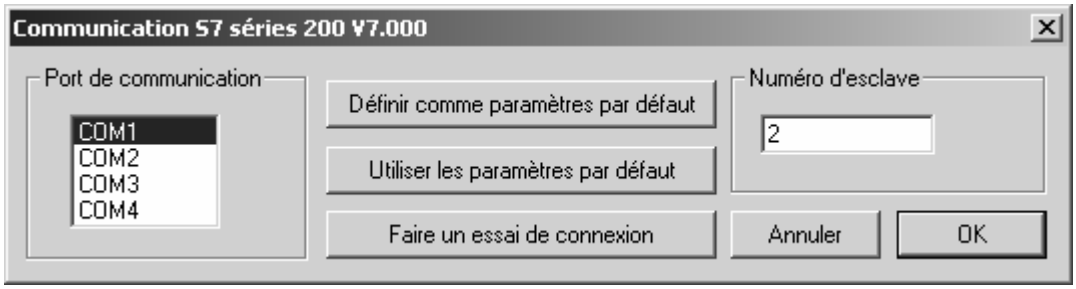
Ce post-processeur permet de programmer les automates SIEMENS S7200 (toutes les CPU 2xx).

Choix du type de CPU

L'élément « Configuration / Post-processeur / STEP7 (S7200) / Système / Configuration matérielle» du navigateur permet de choisir le type de la CPU.



Module de communication



Paramétrage du module de communication

Veillez à régler le numéro d'esclave en accord avec celui configuré sur l'automate.

Exemple spécifique

Cet exemple se trouve dans le répertoire « <répertoire d'installation d'AUTOMGEN> / Exemples / Post-processeurs / S7200 ». Le fichier porte le même nom que le titre du chapitre qui suit.

Tâche d'interruption

Exemple d'appel d'une tâche d'interruption.

Post-processeur ABB

Ce post-processeur permet de programmer les automates ABB CS31 et AC31.

Choix du type d'automate

L'élément « Configuration / Post-processeur / ABB / Système / Configuration matérielle » du navigateur permet de choisir le type de l'automate.

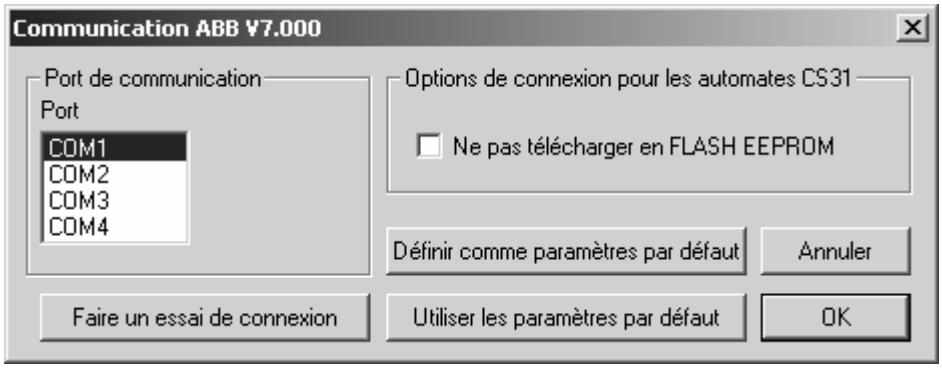
Automate AC31

Configuration matérielle	
Type de l'automate	AC
Code compatible AC31	Oui

Automate CS31

Configuration matérielle	
Type de l'automate	CS
Code compatible AC31	Non

Module de communication



Paramétrage du module de communication

Utilitaire

L'élément « Configuration / Post-processeur / ABB / Terminal emulator » du navigateur permet d'accéder à un émulateur de terminal utilisable pour dialoguer avec l'automate.

Exemples spécifiques

Ces exemples se trouvent dans le répertoire « <répertoire d'installation d'AUTOMGEN> / Exemples / Post-processeurs / ABB ». Les fichiers portent le même nom que les titres des chapitres qui suivent.

Entrées / sorties analogiques

Exemple illustrant l'utilisation d'un module d'extension analogique sur un automate AC31.

Interruptions

Exemple illustrant l'utilisation des tâches d'interruptions sur un automate AC31.

Post-processeur GE-FANUC / ALSPA

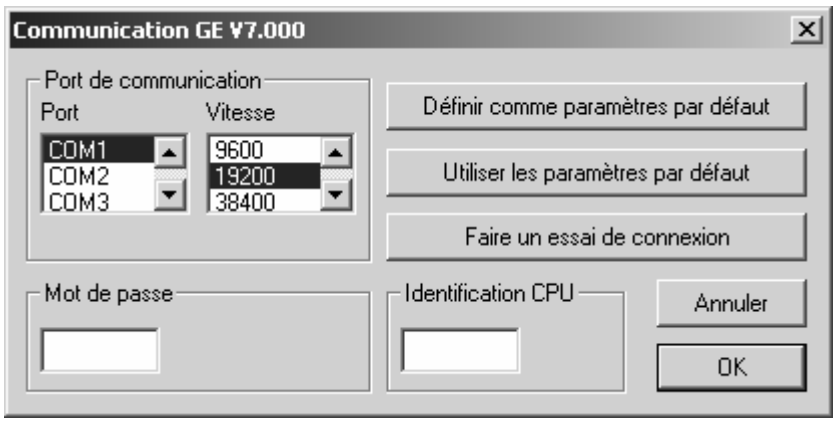
Ce post-processeur permet de programmer les automates GE-FANUC 90 MICRO et 9030 ou ALSPA 8005 et 8035.

Choix du type d'automate

L'élément « Configuration / Post-processeur / GE-FANUC / Système / Configuration matérielle » du navigateur permet de choisir le type de l'automate.

Choisir standard pour les CPUs autres que 350, 351 ou VERSAMAX.

Module de communication



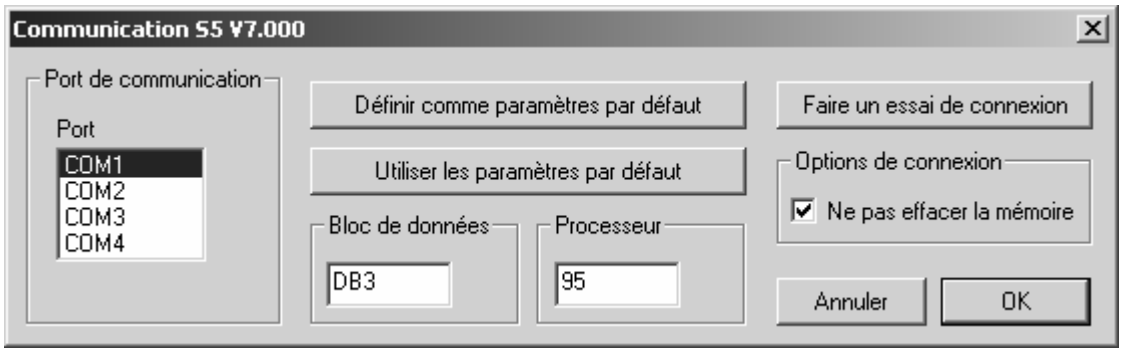
Paramétrage du module de communication

Utilitaire

L'élément « Configuration / Post-processeur / GE-FANUC / Hardware configuration & diagnostic » du navigateur permet d'accéder à un utilitaire de configuration et de diagnostique.

Post-processeur STEP5

Module de communication



Paramétrage du module de communication

Structure de l'application

Le langage STEP5 de SIEMENS est organisé en blocs de programmes et de données. Les applications AUTOMGEN traduites par le post-processeur STEP5 sont découpées en plusieurs blocs. Par défaut, le post-processeur utilise les blocs suivants :

- OB1 bloc d'organisation : ce bloc appelle tous les blocs qui doivent être traités de façon cyclique.
- OB20, OB21, OB22 : blocs exécutés au démarrage de l'automate. Ces blocs arment un bit pour activer les étapes initiales de Grafcet.

Des blocs PB sont utilisés pour le traitement des prédispositions, pour gérer l'évolution des variables booléennes et des temporisations.

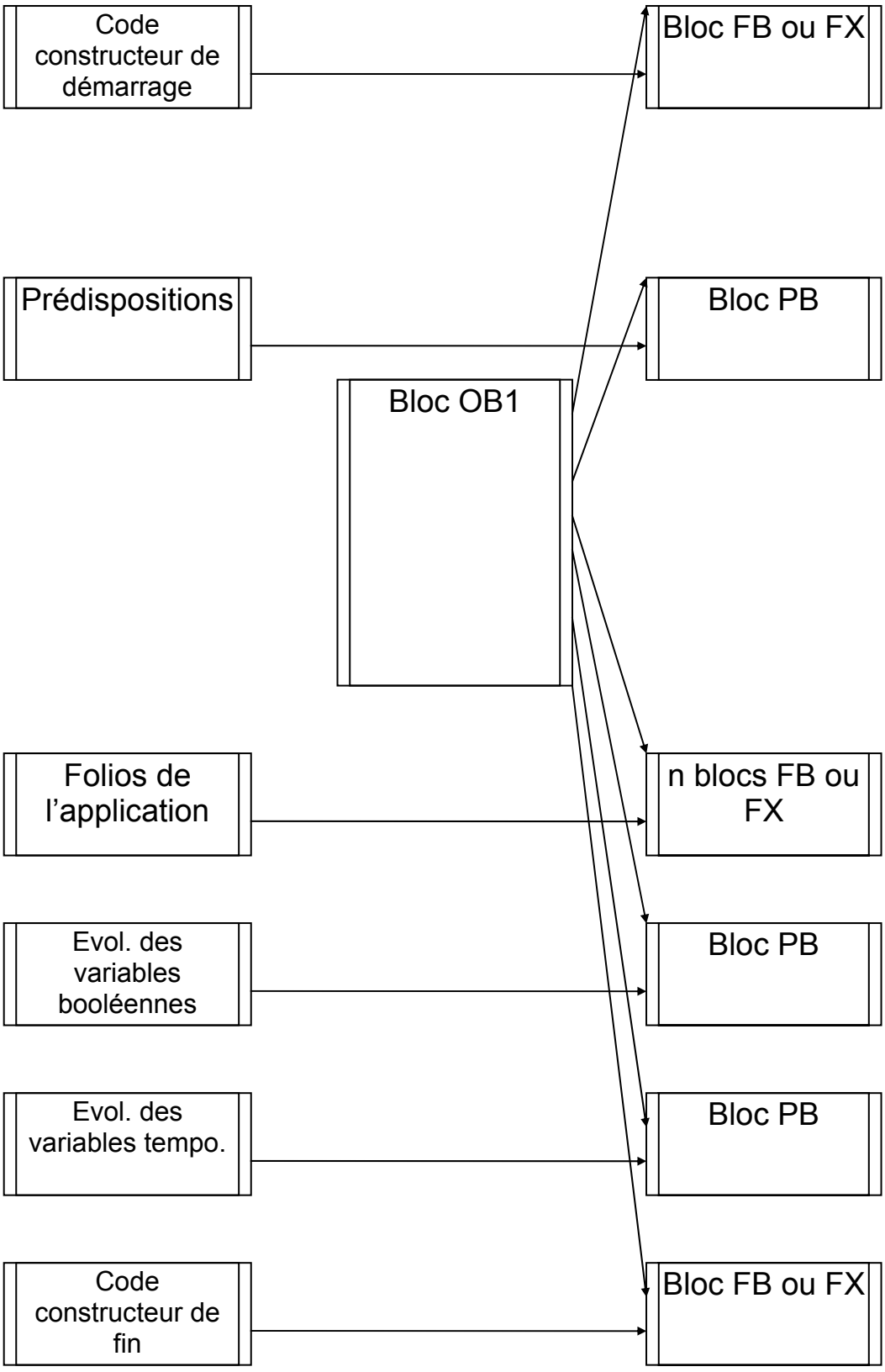
Des blocs FB ou FX sont utilisés pour le code issu de l'application et pour le code écrit dans les fichiers « .SRT » et « .END ». Un bloc FB ou FX est créé pour chaque folio de l'application.
De plus, des folios peuvent directement être associés à un bloc de codes ou de données.

Si un volume de code trop important est généré pour un bloc (code issu d'un folio contenant un programme volumineux par exemple), alors le post-processeur utilise automatiquement un nouveau bloc.

Par défaut, le post-processeur utilise selon ses besoins les blocs PB1 à PB255 et FB1 à FB239.

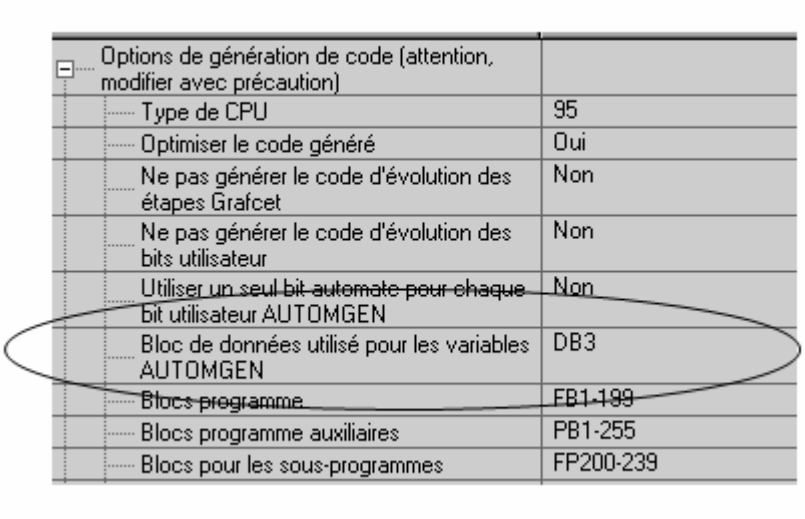
Ces valeurs peuvent être modifiées (voir chapitre Choix des blocs de programmes à utiliser).

La figure suivante illustre la structure du code généré par le post-processeur SIEMENS :



Choix des blocs de programmes à utiliser

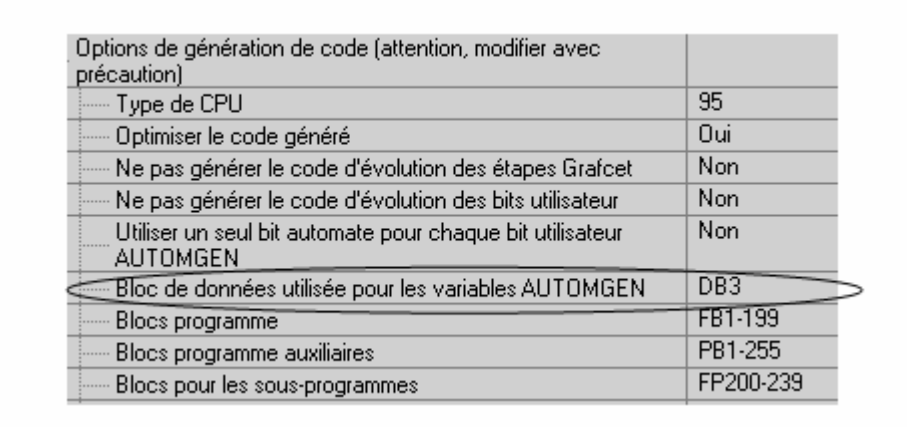
Par défaut, les blocs PB 1 à PB 255 et FB 1 à FB 239 sont utilisés. Trois éléments de configuration permettent de choisir d'autres blocs.



Options de génération de code (attention, modifier avec précaution)	
Type de CPU	95
Optimiser le code généré	Oui
Ne pas générer le code d'évolution des étapes Grafcet	Non
Ne pas générer le code d'évolution des bits utilisateur	Non
Utiliser un seul bit automate pour chaque bit utilisateur AUTOMGEN	Non
Bloc de données utilisé pour les variables AUTOMGEN	DB3
Blocs programme	FB1-199
Blocs programme auxiliaires	PB1-255
Blocs pour les sous-programmes	FP200-239

Choix du bloc de données

Par défaut le bloc DB 3 est utilisé pour les variables numériques. Cette directive permet d'utiliser un autre bloc.



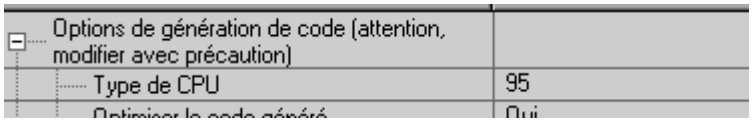
Options de génération de code (attention, modifier avec précaution)	
Type de CPU	95
Optimiser le code généré	Oui
Ne pas générer le code d'évolution des étapes Grafcet	Non
Ne pas générer le code d'évolution des bits utilisateur	Non
Utiliser un seul bit automate pour chaque bit utilisateur AUTOMGEN	Non
Bloc de données utilisée pour les variables AUTOMGEN	DB3
Blocs programme	FB1-199
Blocs programme auxiliaires	PB1-255
Blocs pour les sous-programmes	FP200-239

Le changement de bloc de données implique deux autres modifications :

- dans le code constructeur de démarrage, il faut créer le bloc de données correspondant,

- il faut sélectionner le bloc de données choisi dans le paramétrage du module de dialogue.

Choix du type de processeur



Association du code écrit sur un folio à un bloc programme

En écrivant du code littéral bas niveau ou du code constructeur écrit dans un organigramme sur un folio de type « Tâche », on associe ce code à un bloc STEP5.

Le numéro de tâche détermine le type et le numéro du bloc.

Le code généré par ce folio doit tenir compte du type du bloc et des instructions utilisables dans ce type de bloc (jeu d'instructions limité dans les blocs OB et PB).

La table ci-dessous donne la correspondance entre la valeur de « n » et le bloc.

Numéro de tâche	*Bloc STEP5*
0 à 255	OB 0 à OB 255
256 à 511	PB 0 à PB 255
512 à 767	FB 0 à FB 255
768 à 1023	FX 0 à FX 255
1024 à 1279	DB 0 à DB 255
1280 à 1535	DX 0 à DX 255

Syntaxes spécifiques

Définition de blocs

La directive « $BLOC <type de bloc> <numéro>» permet de définir le début d'un bloc de programme ou de données.

Le type de bloc peut être « OB », « FB », « FX », « PB » pour le code ou « DB », « DX » pour les données.

Le numéro du bloc est une valeur comprise entre 0 et 255. Les blocs « FX » et « DX » ne peuvent être utilisés que sur les automates 135U et 155U. La directive « BE » marque la fin d'un bloc.

Exemple :

$BLOC DB1
...
$BE
$BLOC OB1
...
$BE

les directives « KH= », « KY= », « KC= », « KM= » et « KF= » insèrent des constantes dans les blocs de données DB.

« KH= » insère une constante 16 bits exprimée en hexadécimal.
« KY= » insère une constante 16 bits exprimée sous la forme de deux valeurs
comprises entre 0 et 255 séparées par une virgule.
« KC= » insère une suite de caractères entourés par des caractères « ' » (apostrophe).
« KM= » insère une constante 16 bits exprimée en binaire.
« KF= » insère une constante 16 bits exprimée en décimal signé.

Exemple :
$BLOC DB 4
KH= 1234
KY=100,5
KC= 'Ceci est un texte'
KM=11111111 00000000
KF=-200
$BE

Blocs FB et FX

Les blocs FB et FX du langage SIEMENS peuvent recevoir des paramètres.

Appel

Si des paramètres doivent être passés à un bloc fonctionnel, alors il faut utiliser la syntaxe suivante :

- l'appel doit être suivi du caractère « * »,
- la ligne suivante doit contenir une instruction de saut « SPA » vers la ligne située au-delà des paramètres,
- les lignes suivantes doivent contenir les paramètres précédés d'un mnémonique « U » pour les bits ou « L » pour les mots. Les constantes doivent être écrites sous la forme « Kx=valeur » (x est le type de constante voir le chapitre Définition de blocs).

Exemple d'appel à un bloc fonctionnel sans paramètre :

```
SPA FB 5
```

Exemples d'appel de bloc fonctionnel avec paramètres :

```
SPA FB 242*
SPA=_label_
L MW10     ; premier paramètre
L MW12     ; deuxième paramètre
U M0.0     ; troisième paramètre
L MW14     ; quatrième paramètre
L MW16     ; cinquième paramètre
@label
SPA FB200*
SPA=_label2_
KY=0,4     ; Exemple de paramètre constant
@label2
```

Ecriture

Dans les blocs FB et FX, il faut utiliser les mnémoniques se terminant par le caractère '=' suivi d'un numéro de paramètre (1=premier paramètre).

Exemple de bloc fonctionnel à deux paramètres (recopie du premier paramètre dans le deuxième) :

```
$BLOC FB 100
L=1
T=2
$BE
```


Post-processeur TSX 07

Ce post-processeur permet de programmer les automates MODICON TELEMECANIQUE SCHNEIDER TSX 07.

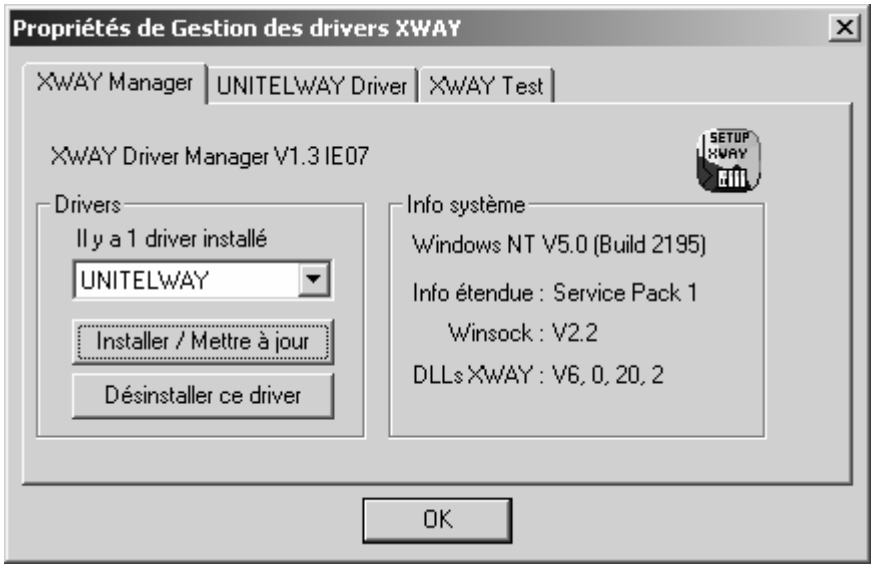
Module de communication

Le driver UNITELWAY SCHNEIDER doit impérativement être installé sur l'ordinateur (en local) pour pouvoir communiquer avec les automates SCHNEIDER TSX 07. Des drivers adaptés à une ou plusieurs versions de WINDOWS se trouvent sur le CD-ROM et peuvent être téléchargés sur le site d'IRAI : www.irai.com.

Attention, certaines versions de WINDOWS sont incompatibles avec certains type d'automates TSX 07 (TSX 0720… et TSX 0721… incompatibles avec WINDOWS ME, WINDOWS 2000, WINDOWS XP, WINDOWS 2003 ou WNDOWS VISTA).

Le module de communication utilise le driver de communication conçu par SCHNEIDER AUTOMATION. Cliquer sur « Paramétrer et tester … » vous permet d'accéder directement au menus du driver de communication SCHNEIDER.

Paramétrage du module de communication

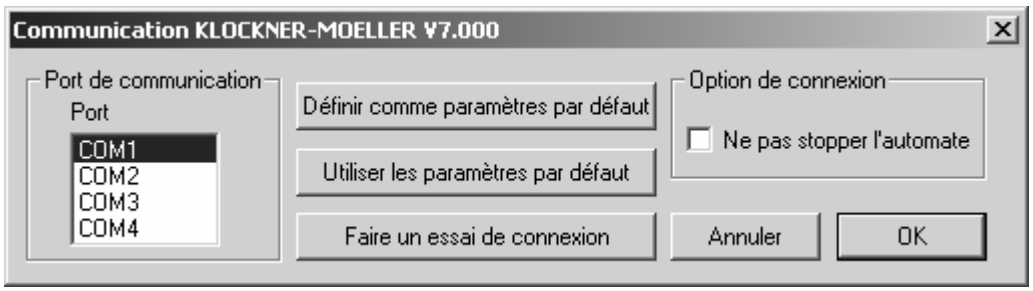


Propriétés du module de communication UNITELWAY

Post-processeur PS3-PS4

Ce post-processeur permet de programmer les automates KLOCKNER-MOELLER PS3 et PS4-100.

Module de communication

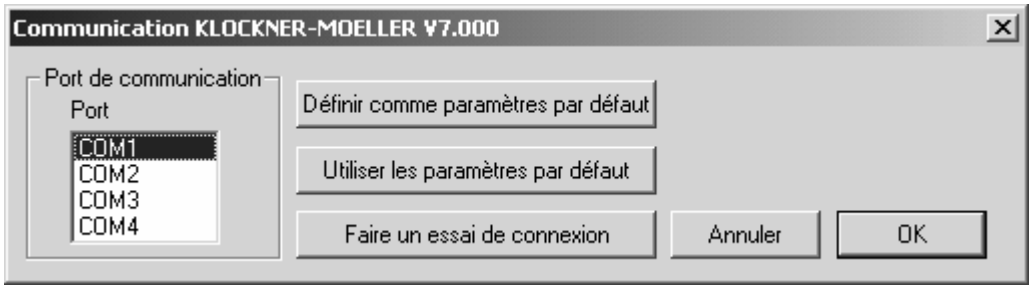


Paramétrage du module de communication

Post-processeur PS4

Ce post-processeur permet de programmer les automates MOELLER PS4-200, PS4-300 et PS416. Le logiciel SUCOSOFT S40 V5 ou supérieure de MOELLER doit être utilisé (la version de démonstration de ce logiciel peut être utilisée).

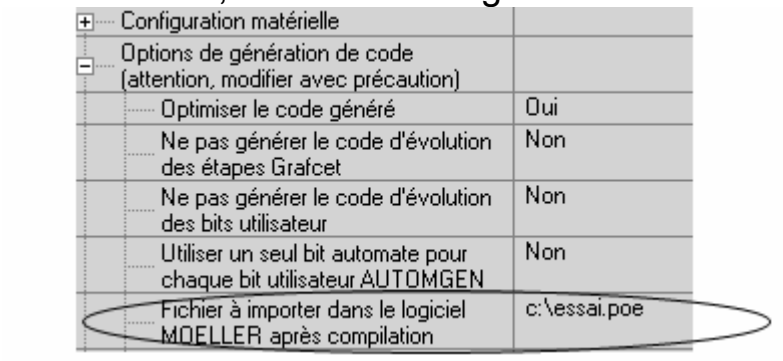
Module de communication



Paramétrage du module de communication

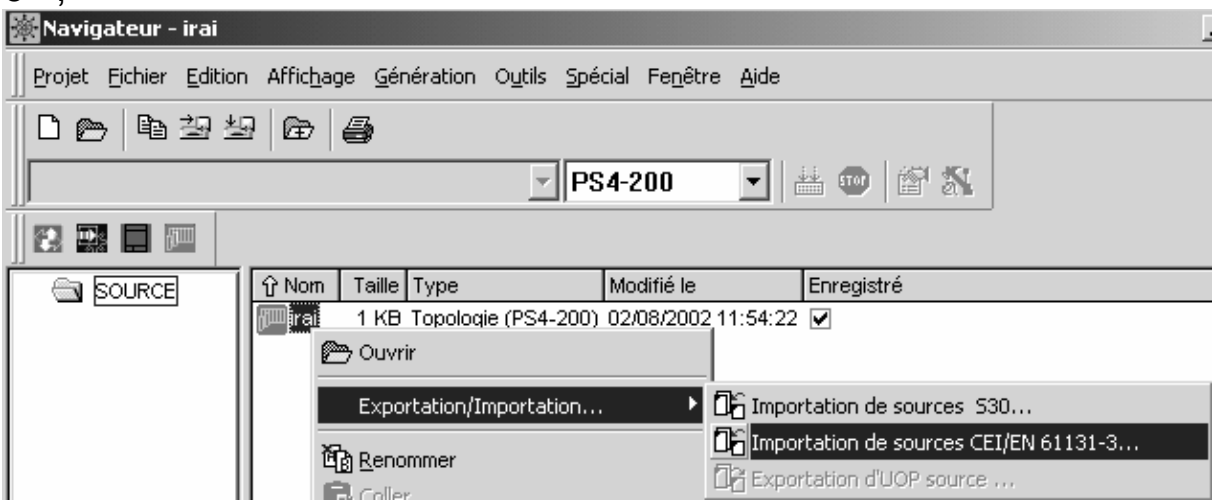
Transfert des programmes vers le logiciel SUCOSOFT S40 de MOELLER

Dans l'élément ci-dessous, désignez un fichier qui servira d'échange entre AUTOMGEN et SUCOSOFT. A la fin de la compilation dans AUTOMGEN, ce fichier sera généré.

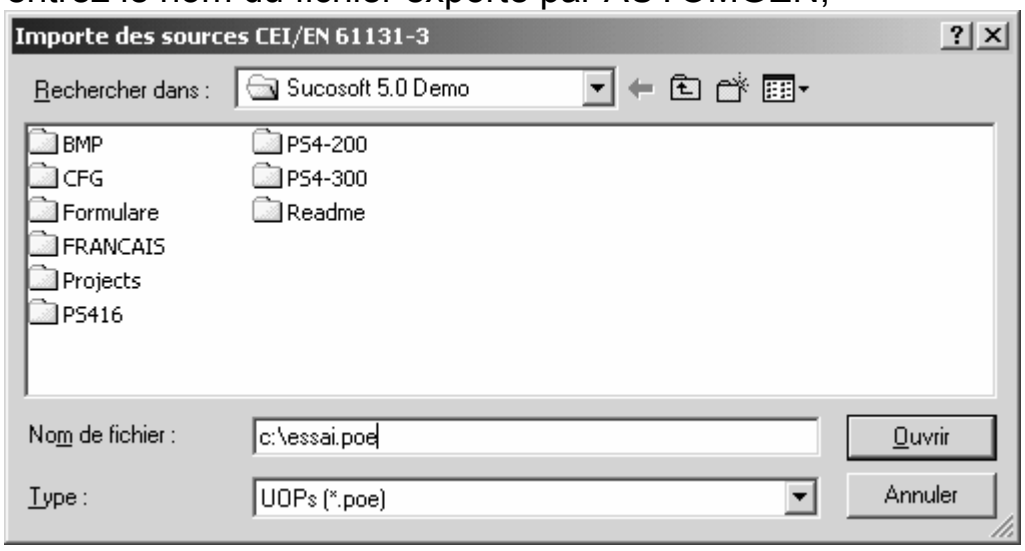


Démarche à suivre pour importer le fichier généré par AUTOMGEN dans le logiciel MOELLER puis l'injecter dans l'automate

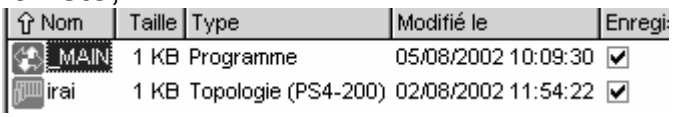
- lancez SUCOSOFT,
- créez un nouveau projet,
- cliquez avec le bouton droit de la souris sur la configuration topologique dans SUCOSOFT et choisissez l'option « Exportation/Importation / Importation de sources CEI/EN 61131-3 »,



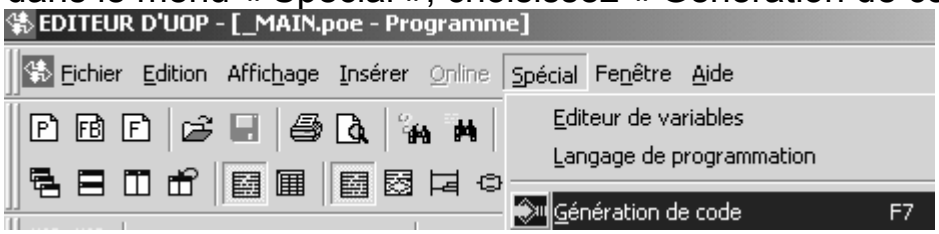
- entrez le nom du fichier exporté par AUTOMGEN,



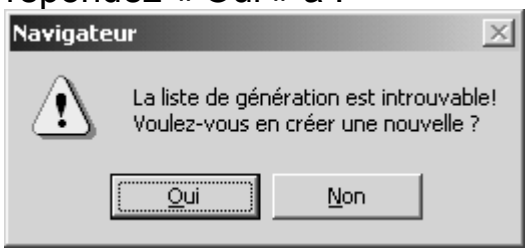
- double cliquez sur l'élément « _MAIN » qui vient d'apparaître dans la liste,



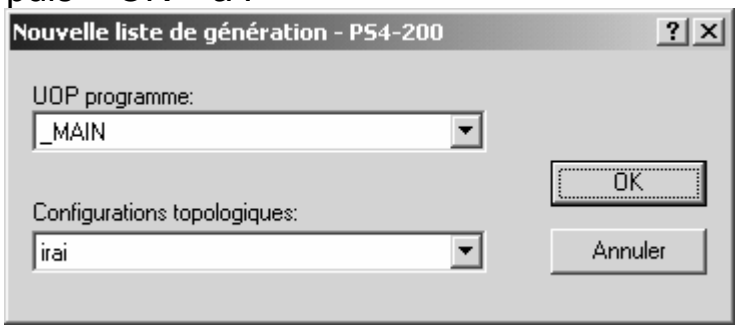
- dans le menu « Spécial », choisissez « Génération de code »,



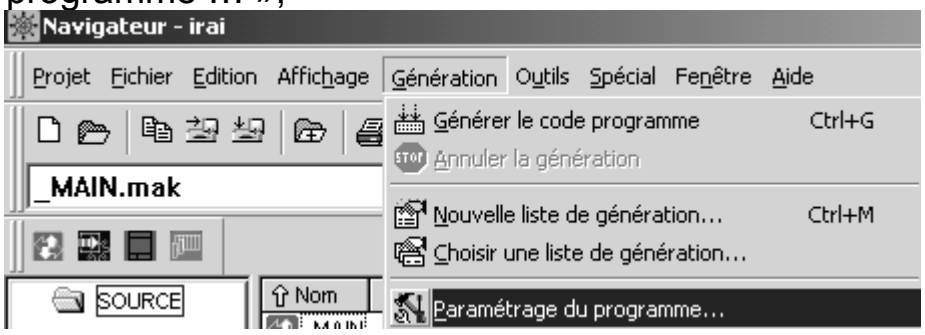
- répondez « Oui » à :



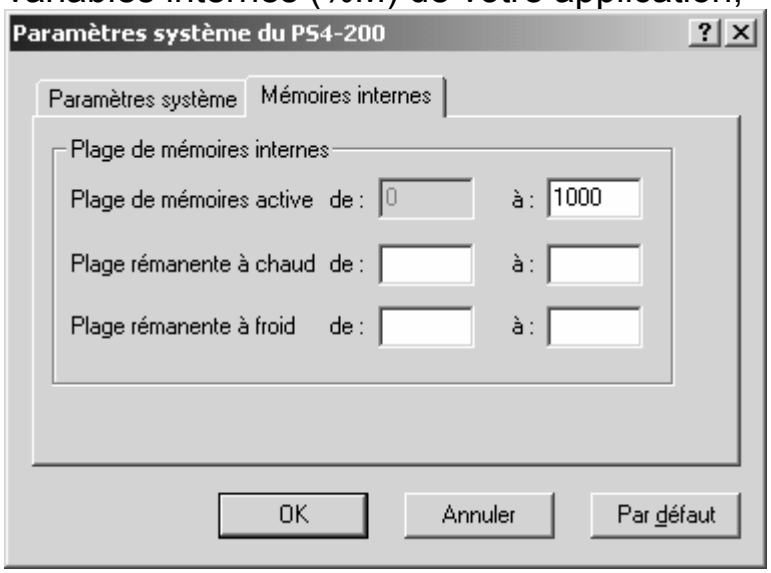
- puis « OK » à :



- dans le menu « Génération », choisissez « Paramétrage du programme … »,

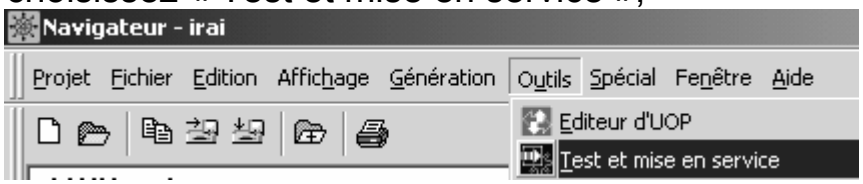


- Sélectionner une taille appropriée pour stocker l'ensemble des variables internes (%M) de votre application,

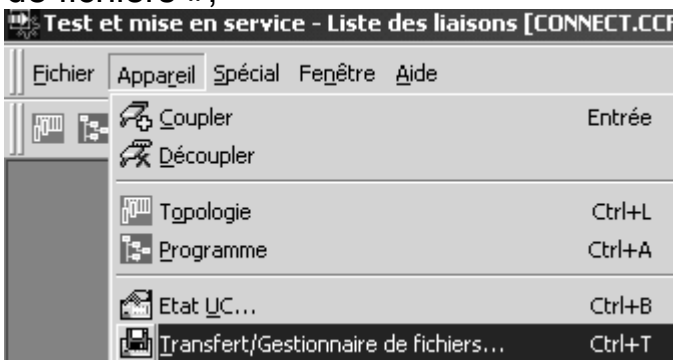


- Dans le menu « Génération », choisissez « Générer le code programme »,

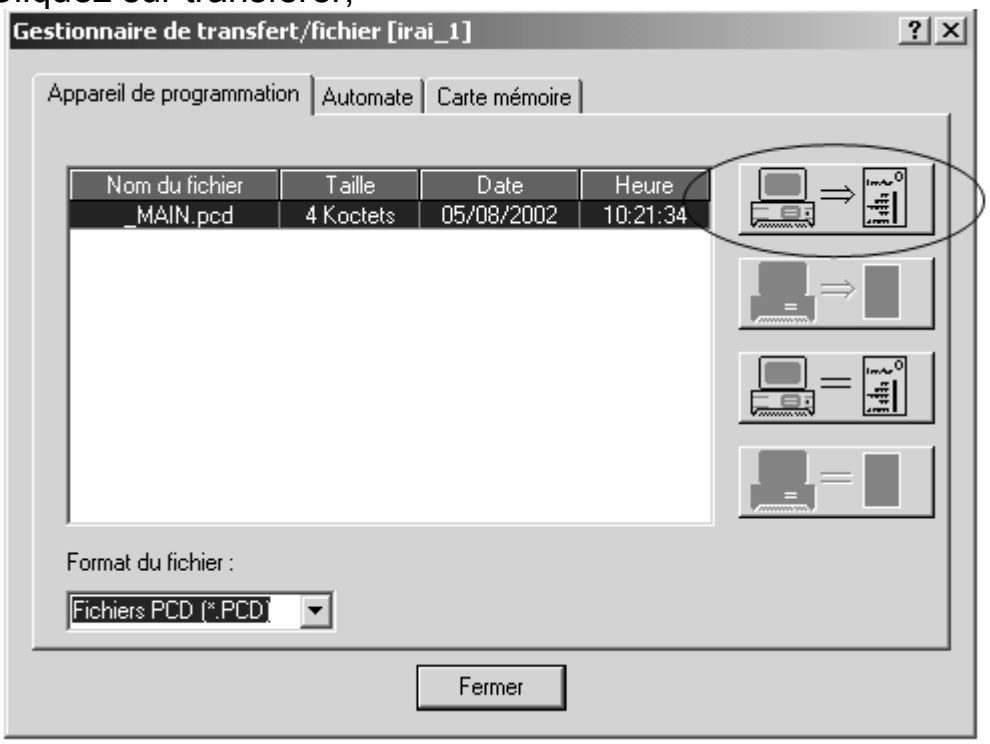
- Si il n'y a pas d'erreur de compilation, vous pouvez procéder au transfert de l'application vers l'automate. Dans le menu « Outils », choisissez « Test et mise en service »,



- Dans le menu « Appareil », choisissez « Transfert / Gestionnaire de fichiers »,



- Cliquez sur transférer,



- A la fin du transfert, déconnecter le logiciel SUCOSOFT pour pouvoir connecter AUTOMGEN à l'automate et activer le mode de mise au point dynamique.

Post-processeur RPX

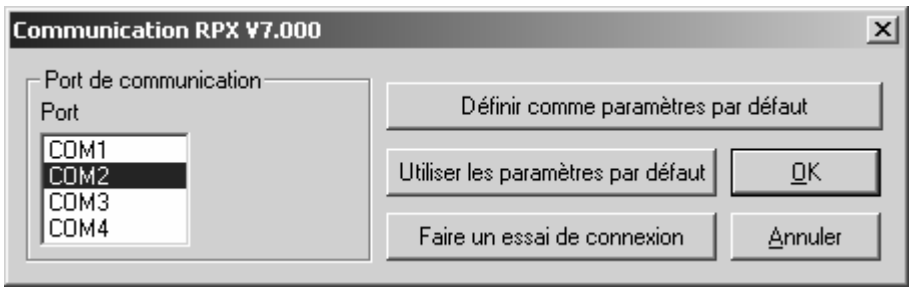
Ce post-processeur permet de programmer les automates CROUZET RPX.

Choix du type d'automate

L'élément « Configuration / Post-processeur / RPX / Système / Configuration matérielle » du navigateur permet de choisir le type de l'automate.

Configuration matérielle	
Type de l'automate	10
Module d'extension présent	NO

Module de communication



Paramétrage du module de communication

Utilitaire

L'élément « Configuration / Post-processeur / RPX / Terminal emulator » du navigateur permet d'accéder à un émulateur de terminal utilisable pour configurer les coupleurs de communication de l'automate.

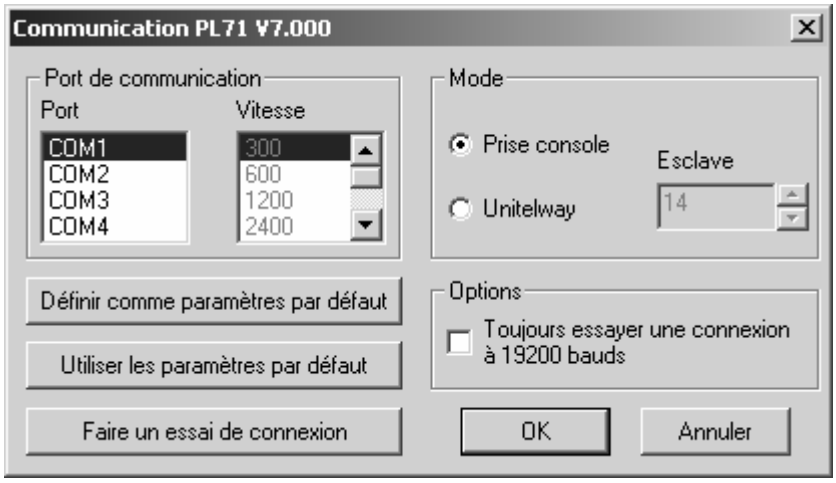
Post-processeur PL71

Ce post-processeur permet de programmer les automates SCHNEIDER TSX 17-10 et TSX 17-20 (sans cartouche PL72).

Choix du type d'automate

L'élément « Configuration / Post-processeur / PL71 / Système / Configuration matérielle » du navigateur permet de choisir le type de l'automate.

Module de communication



Paramétrage du module de communication

(Pour plus d'informations, reportez-vous à la configuration du Module de communication PL72).

Tâche compteur rapide

Un folio de type tâche portant le numéro 1 sera associé à la tâche compteur rapide de l'automate.

Exemples spécifiques

Ces exemples se trouvent dans le répertoire « <répertoire d'installation d'AUTOMGEN> / Exemples / Post-processeurs / PL71 ». Les fichiers portent le même nom que le titre des chapitres qui suivent.

Comptage

Les incrémentations et décrémentations de compteurs en PL71 étant limitées (sur front montant uniquement) par rapport aux possibilités d'AUTOMGEN et des automates TSX il est nécessaire d'utiliser du code en langage constructeur si l'on veut les utiliser (voir le contenu de l'exemple)

Compteur rapide

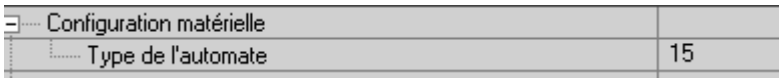
Le but est de compter 200 impulsions sur le compteur rapide. La sortie O5 sera activée par la tâche rapide en fin de comptage.

Post-processeur PB

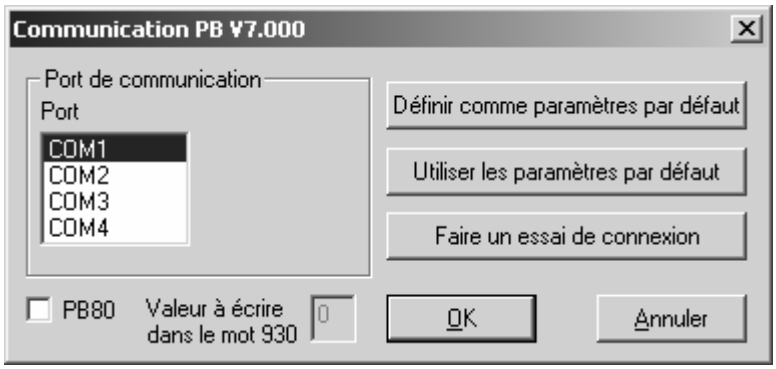
Ce post-processeur permet de programmer les automates SCHNEIDER APRIL PB.

Choix du type d'automate

L'élément « Configuration / Post-processeur / PB / Système / Configuration matérielle » du navigateur permet de choisir le type de l'automate.



Module de communication



Paramétrage du module de communication

Syntaxes spécifiques

La directive « $ORG=xxxx » permet de définir le début de l'adresse d'assemblage, au départ l'adresse d'assemblage est fixée à 0C30,

Exemple :
$ORG=1C30

La directive « $TOP=xxx » définit l'adresse maximale pour le saut de page. Elle précise les trois digits de poids faible des adresses au dessus desquelles un saut de page sera automatiquement généré par l'assembleur.

La directive « $CONST=xxxx,yyyy » définit l'adresse de départ et de fin pour le stockage des constantes. Les constantes sont logées dans une table en dehors du programme.

La directive « WORD xxxx » insère la valeur xxxx (quatre digits en hexadécimal) dans le programme.

La directive « ADR xxxx » insère l'adresse de la variable xxxx (quatre digits en hexadécimal) dans le programme.

La syntaxe #nnnn permet de faire référence à une valeur constante.

Par exemple :
apl #1234 ; place la constante 1234 (hexadécimal) dans l'accumulateur.

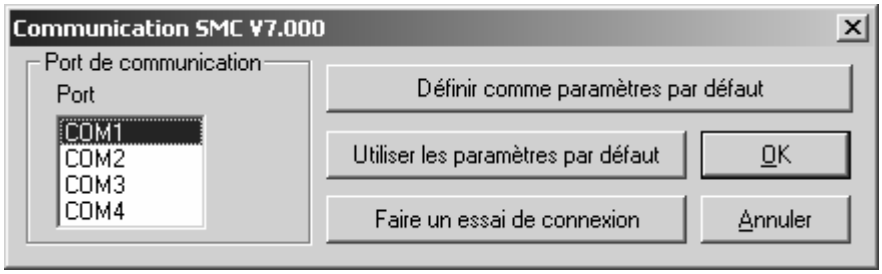
Post-processeur SMC

Ce post-processeur permet de programmer les automates SCHNEIDER APRIL SMC.

Choix du type d'automate

L'élément « Configuration / Post-processeur / SMC / Système / Configuration matérielle » du navigateur permet de choisir le type de l'automate.

Module de communication



Paramétrage du module de communication

Syntaxes spécifiques

La directive « $SEQ » marque le début d'une zone booléenne.

La directive « $CAL » débute une zone de calcul.

La directive « $PRD » débute une zone de prédisposition de variables.

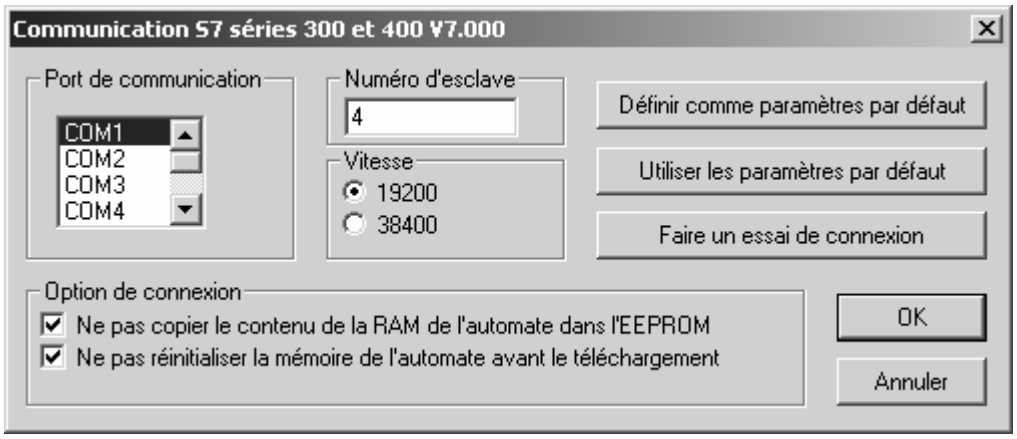
Les variables booléennes peuvent être utilisées en bistable ou en monostable indépendamment des conventions du langage SMC. Le caractère « ! » placé après le signe « = » force la variable à être bistable (mise à un ou mise à zéro), le caractère « ? » placé après le signe « = » force la variable à être monostable (affectation ou affectation complémentée).

La syntaxe « SS.cccccccc » permet d'écrire une séquence de sécurité (nécessaire sur les automates SMC 25 et 600), « cccccccc » représente un nom de programme sur 8 caractères maximum.

Post-processeur S7300

Ce post-processeur permet de programmer les automates SIEMENS S7300.

Module de communication



Paramétrage du module de communication

Le numéro d'esclave doit être en accord avec celui paramétré dans l'automate.

Syntaxes spécifiques

La directive « $BLOC <type de bloc> <numéro>» permet de définir le début d'un bloc de programme ou de données.

Le type de bloc peut être « OB », « FB », « FC », « SFC », « SFB », pour le code ou « DB » pour les données. Le numéro du bloc est une valeur comprise entre 0 et 65535. La directive «$ENDBLOC » marque la fin d'un bloc.

Exemple :
$BLOC DB1
...
$ENDBLOC
$BLOC OB1
...
$ENDBLOC

La syntaxe suivante permet de déclarer les variables pour un bloc :

Pour une variable d'entrée :

$VAR-nature type {:=initialisation}

ou

$VAR-nature symbole : {type :=initialisation}

« nature » peut être :

- « IN » pour une variable d'entrée,
- « OUT » pour une variable de sortie,
- « INOUT » pour une variable d'entrée / sortie,
- « TEMP » pour une variable temporaire,
- « STAT » pour une variable statique.

« type » peut être un des types de variable du langage STEP7 : BOOL, INT, WORD, STRUCT, ENDSTRUCT, etc …

« symbole » permet d'associer un mnémonique à une variable.

« initialisation » est optionnel et fixe la valeur par défaut d'une variable.

Les blocs DB n'autorisent que des variables de type statique.

Les blocs OB n'autorisent que des variables de type temporaire.

Les blocs FC et SFC n'autorisent pas des variables de type statique.

Comme dans le logiciel SIEMENS, les déclarations de variables doivent apparaître dans l'ordre suivant : entrée, sortie, entrée / sortie, statique et temporaire.

Définition des variables d'un bloc

La syntaxe « £D bloc déclaration » permet de définir une déclaration associée à un bloc particulier. Lorsque ce bloc est généré par le compilateur, alors la déclaration est utilisée.

Appel des blocs

La syntaxe « CALL nom de bloc {,DB d'instance} (liste des paramètres) » permet d'appeler un bloc FC, FB, SFC ou SFB.

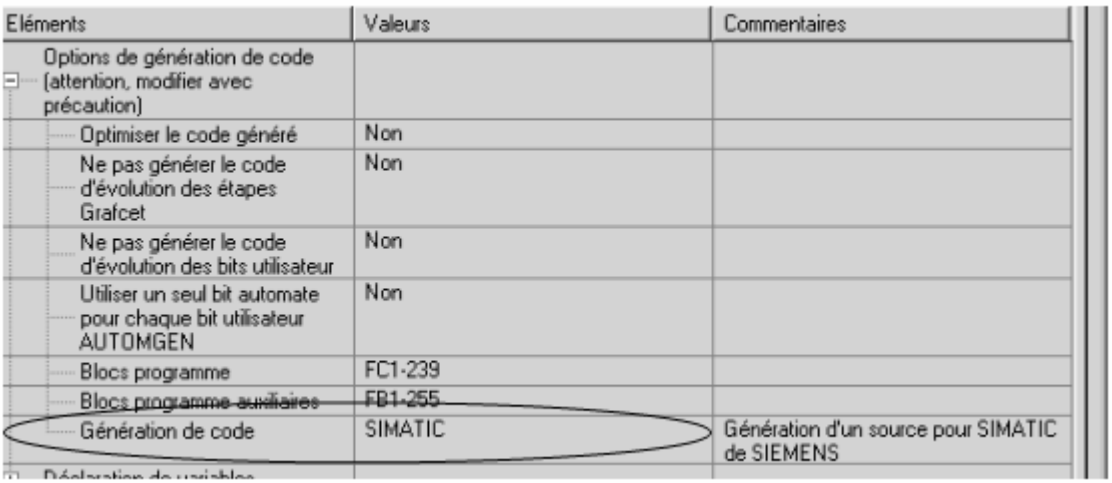
Exemple :

```
$BLOC FC1
$VAR-IN entree1 :BOOL :=FALSE ;
$VAR-IN entree2 :BOOL :=FALSE ;
$VAR-OUT sortie :BOOL ;
u_entree1_
u_entree2_
=_sortie_
$ENDBLOC

$BLOC OB1
CALL FC1(_entree1_ :=e0.0,_entree2_ :=e0.1,_sortie_ :=a0.0)
$ENDBLOC
```

Importation dans le logiciel SIMATIC de SIEMENS

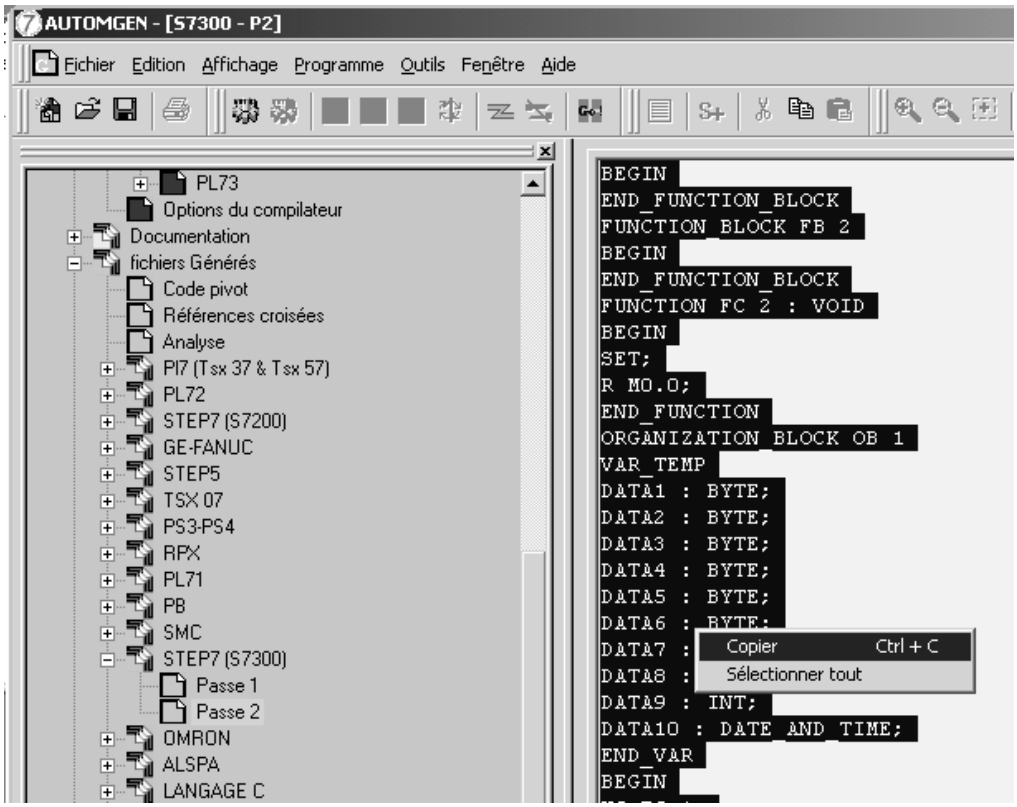
Pour importer le code généré par AUTOMGEN dans le logiciel SIMATIC de SIEMENS, suivre la procédure suivante :

1- Dans la partie « Système » de la configuration du post-processeur S7300, sélectionnez SIMATIC dans l'élément suivant

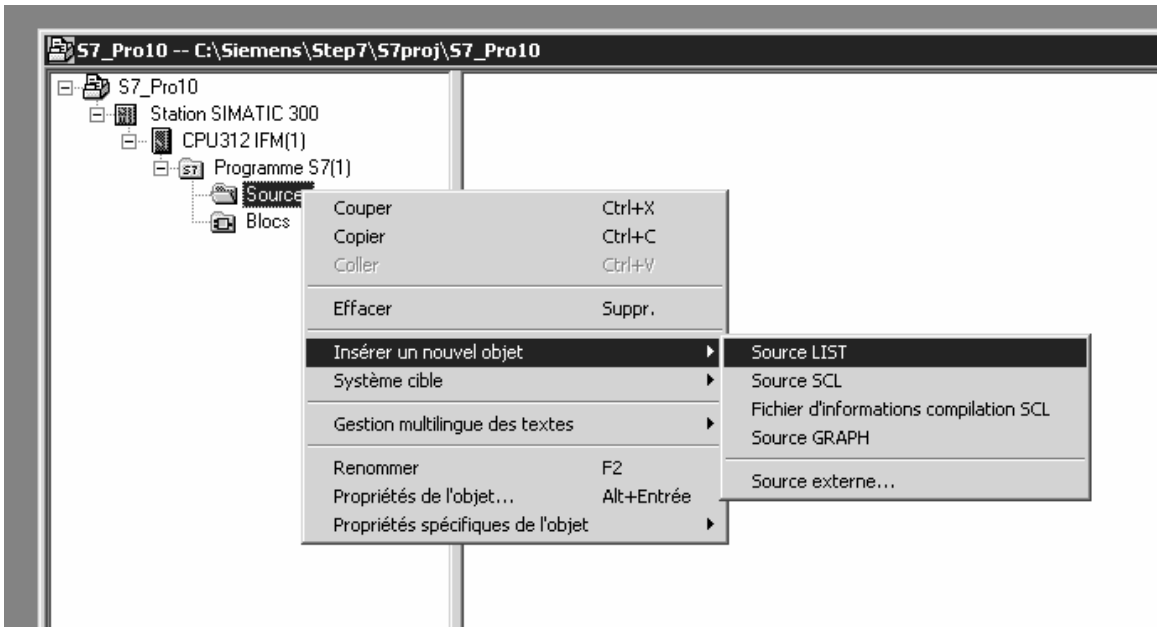


Eléments	Valeurs	Commentaires
Options de génération de code (attention, modifier avec précaution)		
Optimiser le code généré	Non	
Ne pas générer le code d'évolution des étapes Grafcet	Non	
Ne pas générer le code d'évolution des bits utilisateur	Non	
Utiliser un seul bit automate pour chaque bit utilisateur AUTOMGEN	Non	
Blocs programme	FC1-239	
Blocs programme auxiliaires	FB1-255	
Génération de code	SIMATIC	Génération d'un source pour SIMATIC de SIEMENS

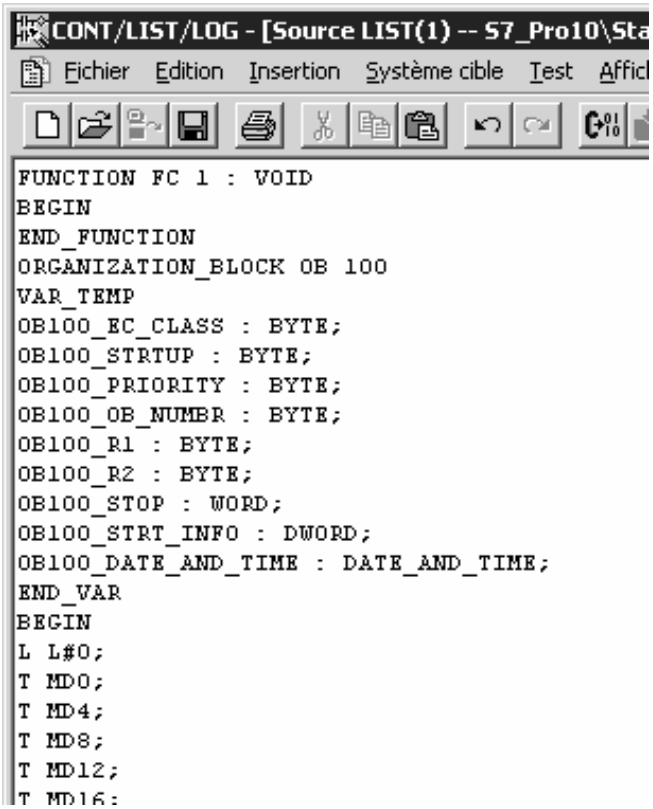
2- Compilez l'application,
3- Dans AUTOMGEN, ouvrez l'élément « Code généré / Post-processeur S7300 / Passe 2 », sélectionnez l'ensemble des lignes puis la commande « Copier » dans le menu « Edition ».



4- Dans le logiciel SIMATIC, créez un élément de type « Source LIST ».



5- Dans SIMATIC, collez le code dans la fenêtre contenant le source LIST avec la commande « Coller » du menu « Edition »,



6- Dans SIMATIC, compilez le source en cliquant sur  .

L'importation est alors terminée.

Structure du code généré

Le langage STEP7 de SIEMENS est organisé en blocs de programmes et de données. Les applications AUTOMGEN traduites par le post-processeur STEP7 sont découpées en plusieurs blocs. Par défaut, le post-processeur utilise les blocs suivants :

- OB1 bloc d'organisation : ce bloc appelle tous les blocs qui doivent être traités de façon cyclique,
- OB100 : blocs exécutés au démarrage de l'automate. Ce bloc arme un bit pour activer les étapes initiales de Grafcet.

Des blocs FB sont utilisés pour le traitement des prédispositions, pour gérer l'évolution des variables booléennes et des temporisations.

Des blocs FC sont utilisés pour le code issu de l'application et pour le code constructeur de démarrage et de fin.

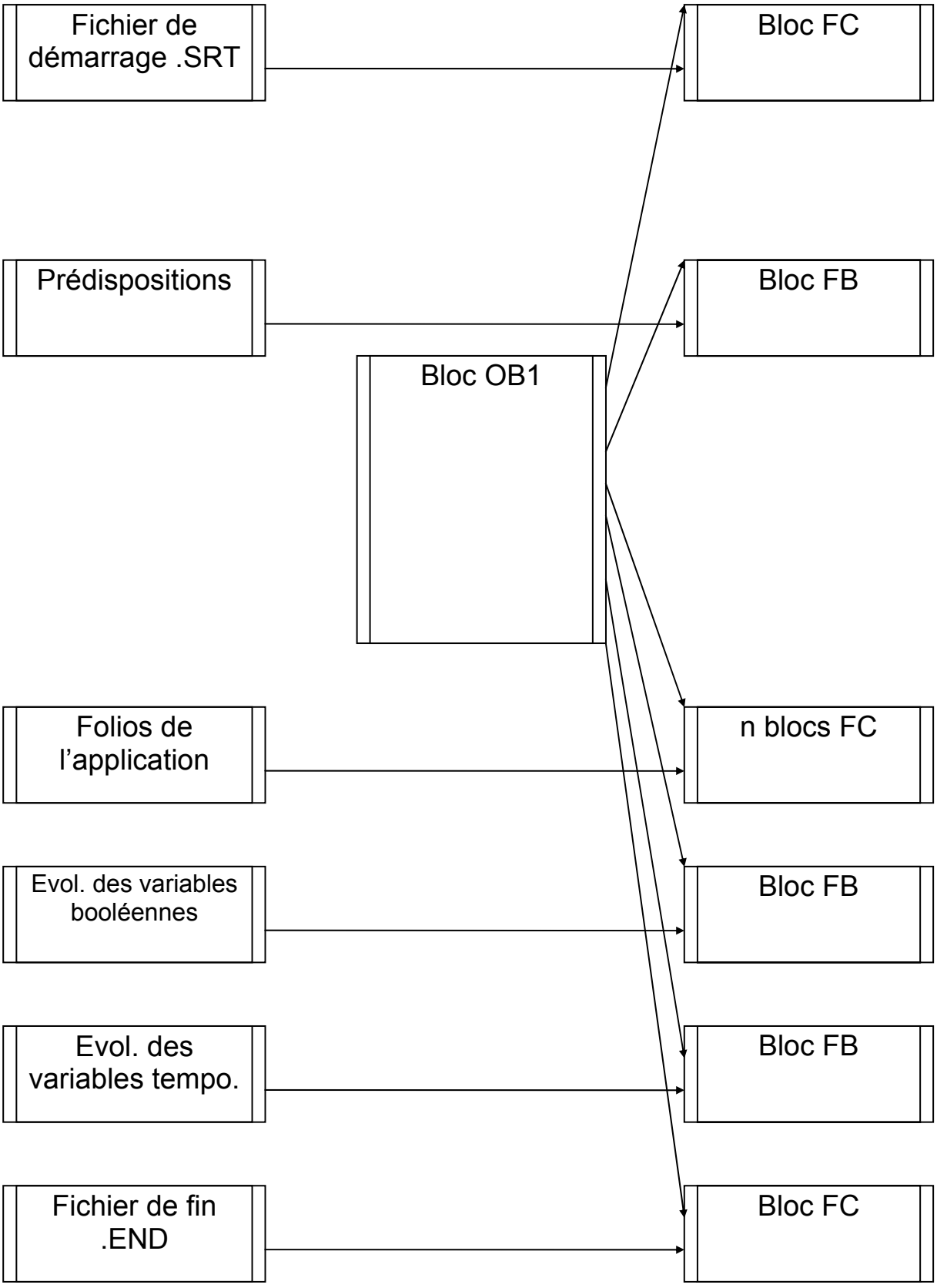
Un bloc FB pour chaque folio de l'application.

De plus, des folios peuvent directement être associés à un bloc (voir chapitre Association du code écrit sur un folio à un bloc programme).

Par défaut, le post-processeur utilise selon ses besoins les blocs FB1 à FB255 et FC1 à FC239.

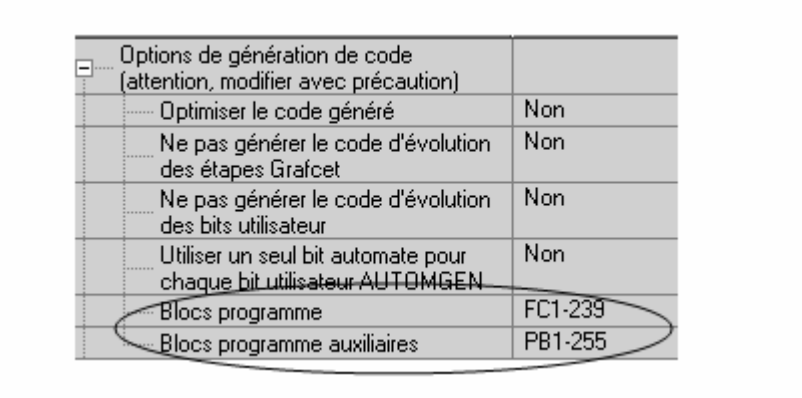
Ces valeurs peuvent être modifiées dans l'élément « Système » de la configuration.

La figure suivante illustre la structure du code généré par le post-processeur SIEMENS :



Choix des blocs de programmes à utiliser

Par défaut, les blocs FC 1 à FC 239 et FB 1 à FB 255 sont utilisés. Deux éléments de configuration permettent de choisir d'autres blocs.



Association du code écrit sur un folio à un bloc programme

En écrivant du code littéral bas niveau ou du code constructeur écrit dans un organigramme sur un folio de type « Tâche », on associe ce code à un bloc STEP7.

Le numéro de tâche détermine le type et le numéro du bloc.

Le code généré par ce folio doit tenir compte du type du bloc et des instructions utilisables dans ce type de bloc (jeu d'instructions limité dans les blocs OB et PB).

La table ci-dessous donne la correspondance entre le numéro de tâche et le bloc STEP7.

Numéro de tâche	*Bloc STEP7*
0 à 255	OB 0 à OB 255
256 à 511	FC 0 à FC 255
512 à 767	FB 0 à FB 255

Exemples spécifiques

Ces exemples se trouvent dans le répertoire « <répertoire d'installation d'AUTOMGEN> / Exemples / Post-processeurs / S7300 ». Les fichiers portent le même nom que les titres des chapitres qui suivent.

Appel d'un bloc STEP7

Exemple d'appel des blocs fonctions STEP7.

Utilisation d'un bloc OB

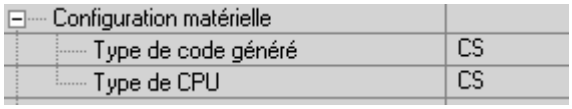
Exemple d'association du code écrit sur un folio à un bloc OB.

Post-processeur OMRON

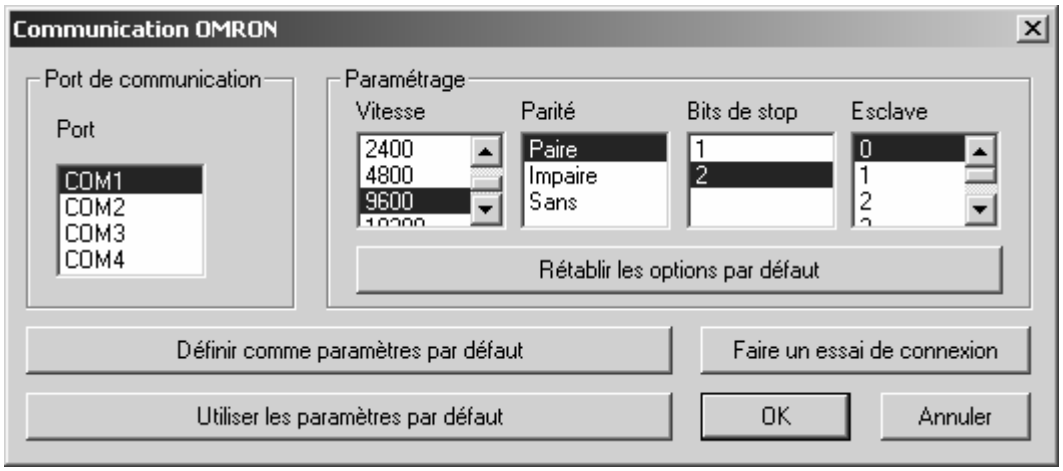
Ce post-processeur permet de programmer les automates OMRON série C, CS et CV. La génération directe des fichiers binaires et le transfert des applications dans l'automate sont supportés pour les automates série C. Pour les automates séries CS et CV, le logiciel CX-PROGRAMMER d'OMRON version 2.0 ou supérieur doit être utilisé.

Choix du type d'automate

L'élément « Configuration / Post-processeur / OMRON / Système / Configuration matérielle » du navigateur permet de choisir le type de l'automate.



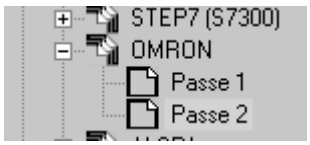
Module de communication



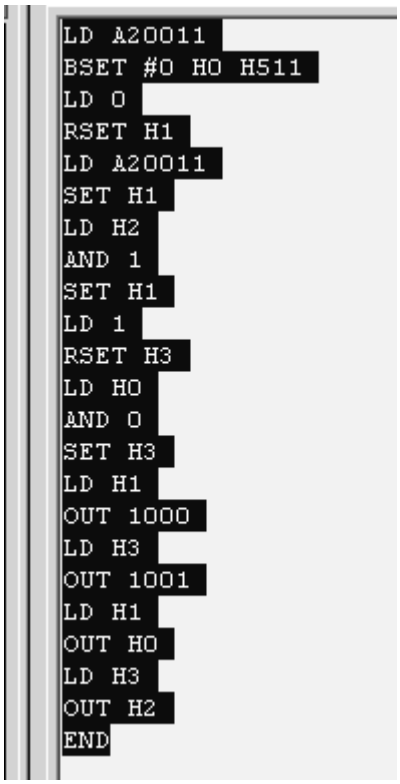
Paramétrage du module de communication

Transfert des applications dans le logiciel CX-PROGRAMMER

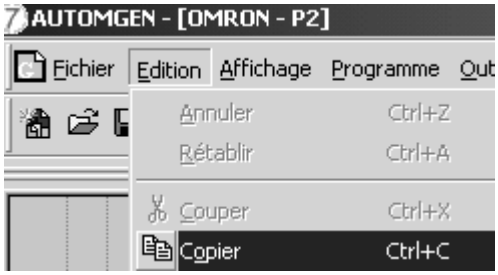
- dans AUTOMGEN, à la fin de la compilation, double cliquez sur l'élément « Fichiers générés / OMRON / passe 2 »,



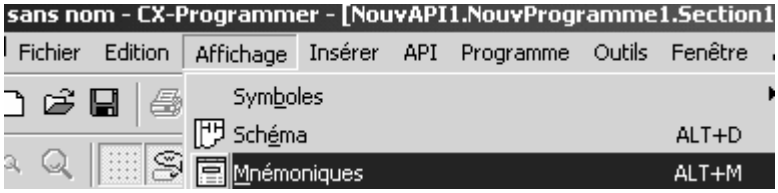
- sélectionnez l'ensemble des lignes,



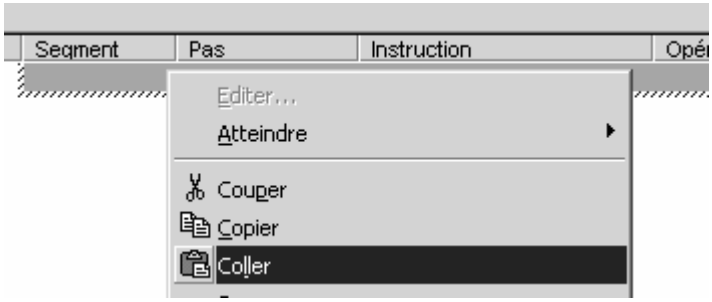
- sélectionnez la commande « Copier » du menu « Edition »,



- dans CX-PROGRAMMER, créer une application vierge, affichez la zone programme sous forme de mnémoniques,



- sélectionnez la zone programme puis collez les lignes,



Segment	Pas	Instruction	Opérande
0	0	LD	P_First_Cycle
	1	BSET(071)	#0
			H0
			H511
1	2	LD	0.00
	3	RSET	H0.01
2	4	LD	P_First_Cycle
	5	SET	H0.01
3	6	LD	H0.02
	7	AND	0.01
	8	SET	H0.01
4	9	LD	0.01
	10	RSET	H0.03
5	11	LD	H0.00
	12	AND	0.00
	13	SET	H0.03
6	14	LD	H0.01
	15	OUT	10.00
7	16	LD	H0.03
	17	OUT	10.01
8	18	LD	H0.01
	19	OUT	H0.00
9	20	LD	H0.03
	21	OUT	H0.02
10	22	END(001)	

Vous pouvez télécharger l'application dans l'automate à partir de CX-PROGRAMMER puis revenir dans AUTOMGEN pour réaliser la mise au point du programme en mode connecté (pensez à déconnecter CX-PROGRAMMER de l'automate pour pouvoir communiquer à partir d'AUTOMGEN).

Syntaxe spécifique

La syntaxe suivante permet de fixer la valeur d'un mot de données :

$DMn=valeur
« n » est le numéro du mot,

« valeur » est une valeur de 16 bits exprimée par défaut en décimal, ou en hexadécimal si elle est précédée du caractère 'H'.

Exemple :

$DM10=50
$DM100=HA000

Association du code écrit sur un folio à un bloc programme

En écrivant du code littéral bas niveau ou du code constructeur écrit dans un organigramme sur un folio de type « Tâche », on associe ce code à une tâche d'interruption. Le numéro de tâche est équivalent au numéro d'interruption.

Exemple spécifique

Cet exemple se trouve dans le répertoire « <répertoire d'installation d'AUTOMGEN> / Exemples / Post-processeurs / S7200 ». Le fichier porte le même nom que le titre du chapitre qui suit.

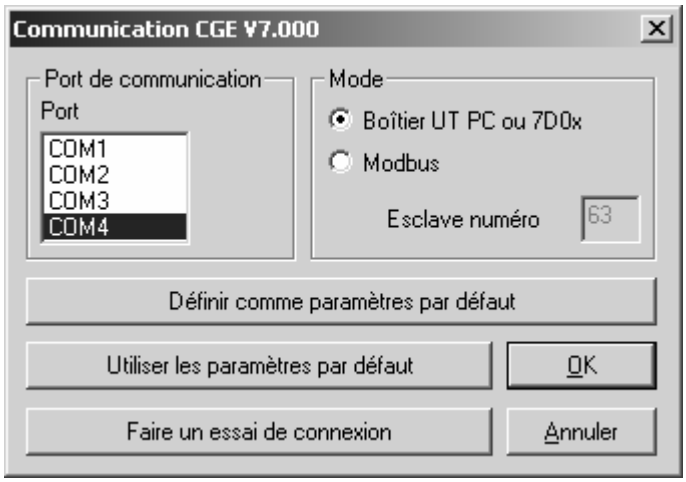
Tâche d'interruption

Exemple d'appel d'une tâche d'interruption.

Post-processeur ALSPA

Ce post-processeur permet de programmer les automates CEGELEC ALSPA C50 et C100.

Module de communication

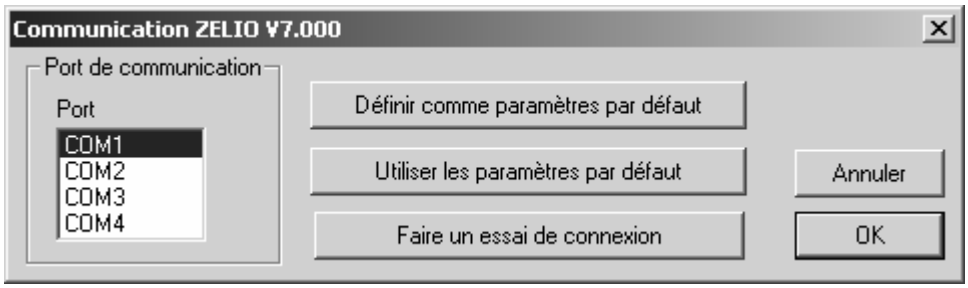


Paramétrage du module de communication

Post-processeur ZELIO

Ce post-processeur permet de programmer les modules SECHNEIDER ZELIO.

Module de communication

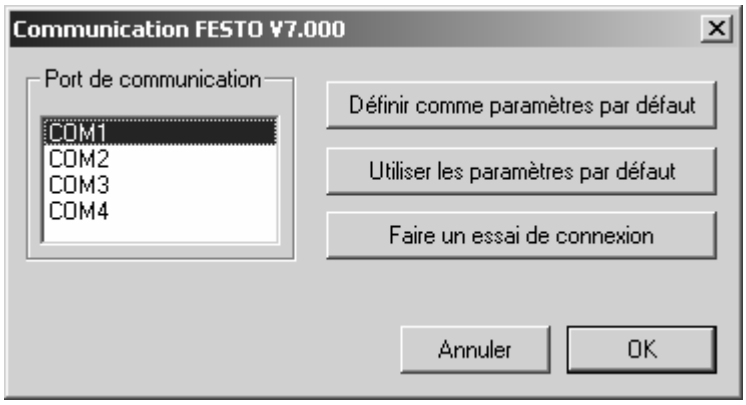


Paramétrage du module de communication

Post-processeur FESTO

Ce post-processeur permet de programmer les automates FPC 101, FPC 103 et FEC FESTO.

Module de communication



Paramétrage du module de communication

Génération de fichier binaire

Les fichiers binaires peuvent être directement générés et téléchargés dans les automates FPC 101 et FPC 103. L'atelier logiciel FST FESTO sous DOS ou sous WINDOWS est nécessaire pour les automates FEC.

Configuration matérielle		
Type de l'automate	FPC103	FPC 103

Choix d'un type de CPU (génération directe d'un fichier binaire)

Importation dans les ateliers logiciel FESTO

Configuration matérielle		
Type de l'automate	NO	Génère un source .AWL mais pas de fichier binaire
Configuration logicielle		
Fichier à importer dans le logiciel FESTO après compilation (si type de l'automate=NO)	C:\export.AWL	

Génération d'un fichier .AWL compatible avec les ateliers FESTO

Si vous utilisez l'atelier logiciel FST FESTO sous DOS, relisez le fichier .AWL à partir de ce logiciel.

Si vous utilisez le logiciel FST FESTO sous WINDOWS, ouvrez le fichier généré dans AUTOMGEN en double cliquant sur l'élément « Fichiers générés / FESTO / Passe 2 », sélectionnez l'ensemble du fichier, utilisez la commande « Copier » du menu « Edition » puis utiliser la commande « Coller » dans le logiciel FESTO pour récupérer le code généré.

Transférer le programme vers l'automate avec le logiciel FESTO.

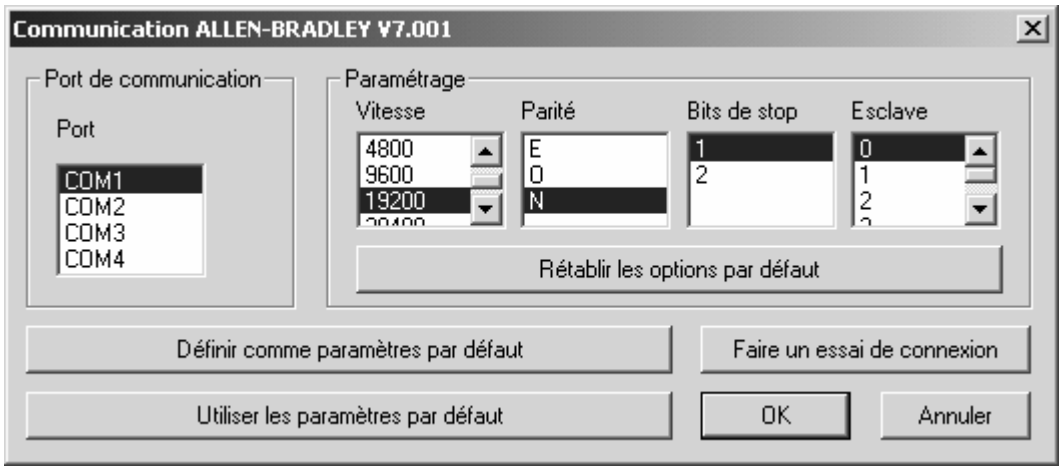
Vous pouvez ensuite vous connecter avec AUTOMGEN (après vous être déconnecté avec le logiciel FESTO) en utilisant comme mode de connexion « Seulement connecter ».

Post-processeur ALLEN-BRADLEY

Ce post-processeur permet de programmer les automates SLC de ROCKWELL. Le logiciel RSLogix 500 V5 ou supérieure de ROCKWELL est nécessaire.

La version STARTER de RSLogix 500 ne permet pas d'importer les fichiers générés par AUTOMGEN.

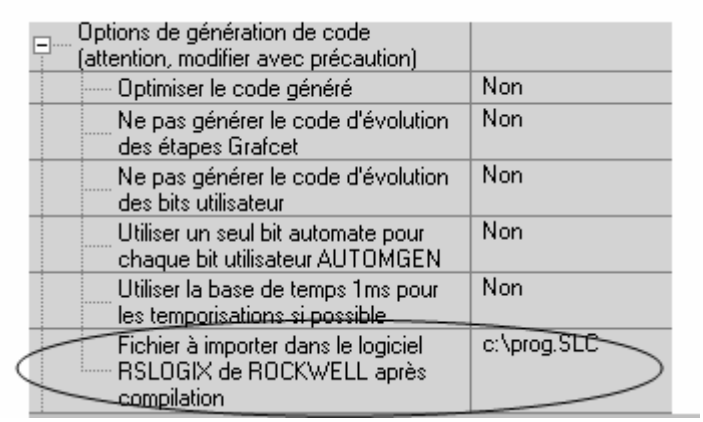
Module de communication



Paramétrage du module de communication

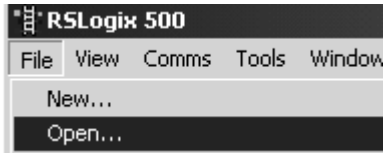
Transfert des programmes vers le logiciel RS-Logix 500 de ROCKWELL Software

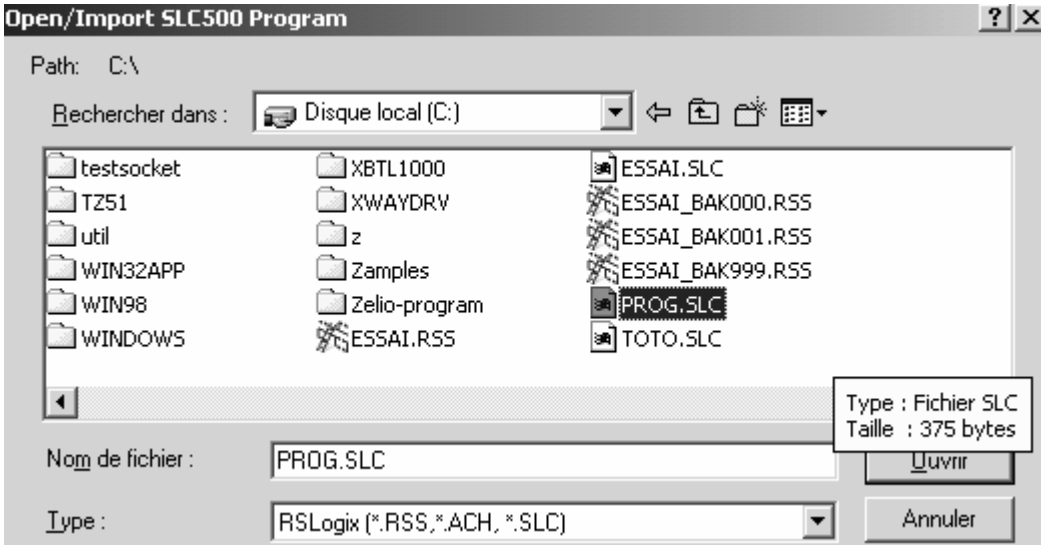
Dans l'élément ci-dessous, désignez un fichier qui servira d'échange entre AUTOMGEN et RSLogix 500. A la fin de la compilation dans AUTOMGEN, ce fichier sera généré.



Génération d'un fichier .SLC compatible avec RSLogix 500

Lancez RSLogix500, puis ouvrez le fichier .SLC généré par AUTOMGEN.





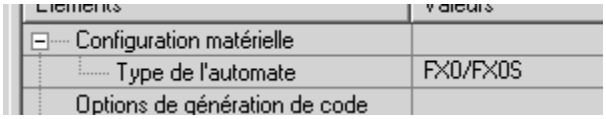
Transférez le programme vers l'automate avec le logiciel RSLogix 500. Après avoir déconnecté RSLogix 500 de l'automate, vous pouvez réaliser la mise au point en mode connecté à partir d'AUTOMGEN.

Post-processeur MITSUBISHI

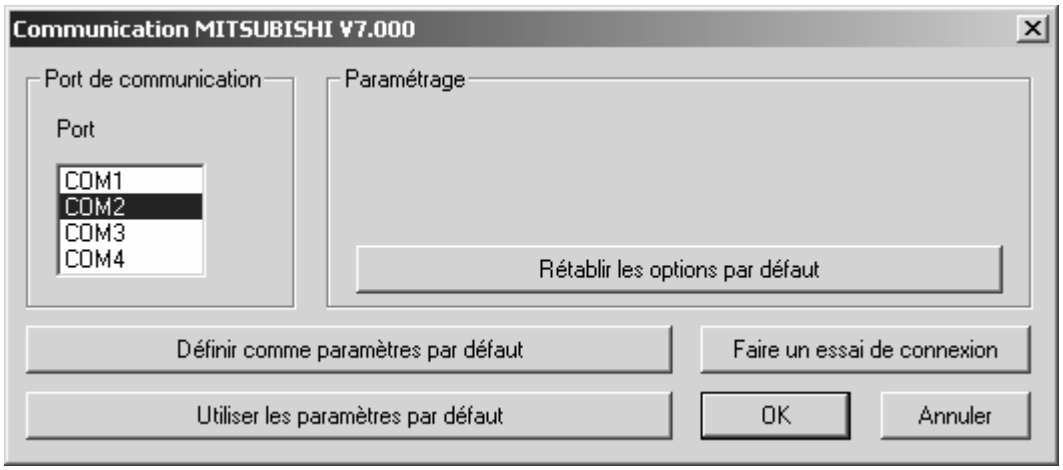
Ce post-processeur permet de programmer les automates MITSUBISHI de la gamme FX. Le code généré par AUTOMGEN peut être directement envoyé dans les automates MITSUBISHI de la gamme FX ou importer dans les logiciels MITSUBISHI FX-WIN ou GX-DEVELOPPER.

Choix du type d'automate

L'élément « Configuration / Post-processeur / MITSUBISHI / Système / Configuration matérielle » du navigateur permet de choisir le type de l'automate.



Module de communication



Paramétrage du module de communication

Transfert des programmes vers le logiciel FX-WIN de MITSUBISHI

Dans l'élément ci-dessous, choisissez FXWIN.

Eléments	Valeurs	Commentaires
Configuration matérielle		
Type de l'automate	FX0/FX0S	
Options de génération de code (attention, modifier avec précaution)		
Optimiser le code généré	Oui	
Ne pas générer le code d'évolution des étapes Grafcet	Non	
Ne pas générer le code d'évolution des bits utilisateur	Non	
Utiliser un seul bit automate pour chaque bit utilisateur AUTOMGEN	Non	
Option de génération de code	FXWIN	Génère un source pour FX WIN
Déclaration de variables		

La version Française de FXWIN doit impérativement être utilisée. L'importation a été validée avec la version 3.20 de FX-WIN.

Après compilation dans AUTOMGEN, créez un projet dans FX-WIN. Ouvrez le programme en édition en mode liste d'instructions et choisissez « Insertion » dans le menu « Edition ».

Transfert des programmes vers le logiciel GX-DEVELOPPER de MITSUBISHI

Dans l'élément ci-dessous, choisissez GXDEVELOPPER.

Configuration matérielle		
Options de génération de code (attention, modifier avec précaution)		
Optimiser le code généré	Oui	
Ne pas générer le code d'évolution des étapes Grafcet	Non	
Ne pas générer le code d'évolution des bits utilisateur	Non	
Utiliser un seul bit automate pour chaque bit utilisateur AUTOMGEN	Non	
Option de génération de code	GXDEVELOPPER	Génère un source pour GX DEVELOPPER
Déclaration de variables		

Lancez l'exécutable « A7TOGX.EXE » qui se trouve dans le répertoire d'installation d'AUTOMGEN. Après lancement, une icône « A7 → GX » apparaît dans la barre des icônes de WINDOWS.

Après compilation dans AUTOMGEN, créez un projet dans GX-DEVELOPPER. Ouvrez le programme en édition en mode liste d'instructions. Pressez simultanément les deux touches SHIFT du clavier pendant une seconde. Le programme est alors transféré dans GX-DEVELOPPER.

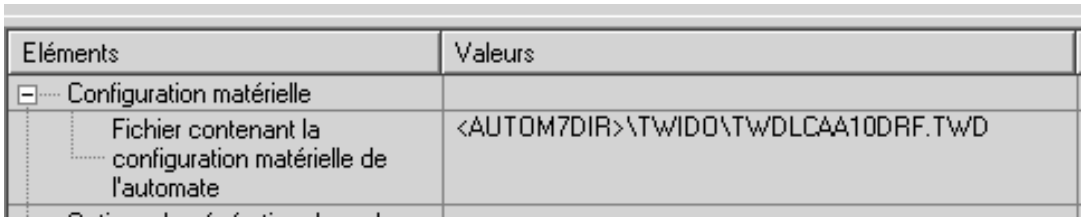
Pour désinstaller l'utilitaire « A7TOGX » cliquez sur l'icône avec le bouton droit de la souris. Vous pouvez laisser « A7TOGX » installé autant que nécessaire, nul n'est besoin de l'installer et de le désinstaller à chaque fois que vous désirez importer une application dans GX-DEVELOPPER.

Post-processeur TWIDO

Ce post-processeur permet de programmer les automates TWIDO de SCHNEIDER.

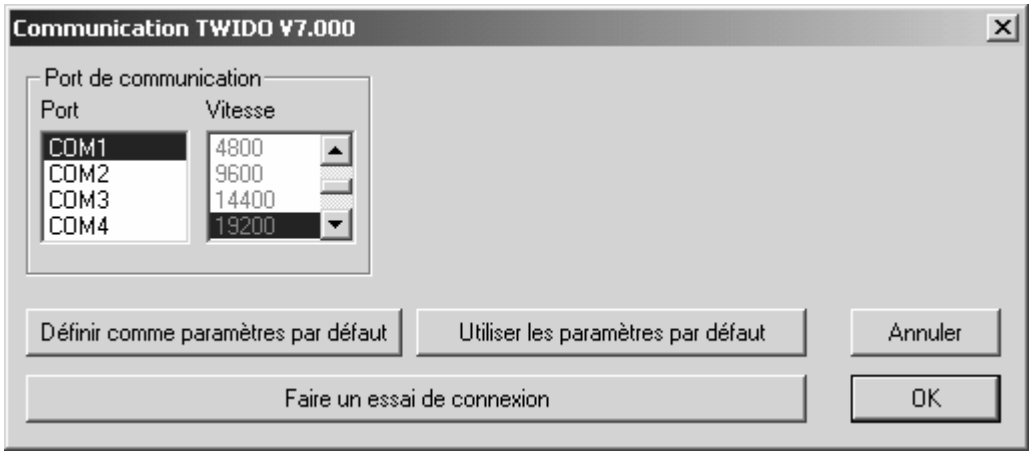
Choix de la configuration de l'automate

L'élément « Configuration / Post-processeur / TWIDO / Système / Configuration matérielle » du navigateur permet de choisir le type de l'automate.



Le fichier « .TWD » est un fichier de configuration qui doit être généré avec l'atelier logiciel TWIDOSOFT de SCHNEIDER. Le sous-répertoire « TWIDO » du répertoire d'installation d'AUTOMGEN contient les fichiers de configuration pour plusieurs types d'automates TWIDO.

Module de communication



Post-processeur ZELIO 2

Ce post-processeur permet de programmer les modules SECHNEIDER ZELIO 2.

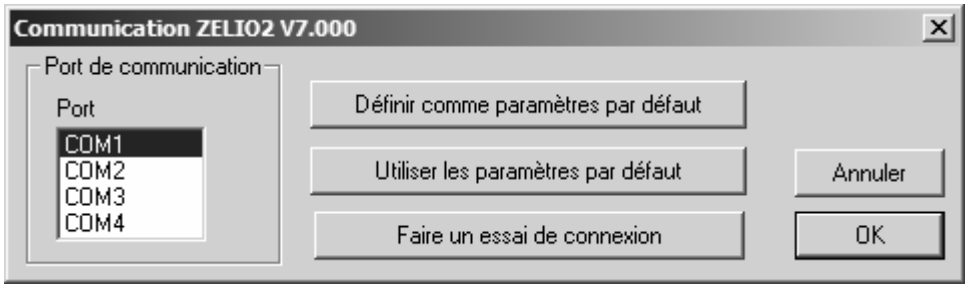
Initialisation de l'automate

Avant de pouvoir être utilisé avec AUTOMGEN, le firmware LD doit être téléchargé dans l'automate ZELIO 2 avec le logiciel ZELIOSOFT de SCHNEIDER.

Configuration de l'automate

La configuration de l'automate doit être réalisée avec le logiciel ZELIOSOFT de SCHNEIDER et téléchargée dans l'automate avec ce même logiciel avant de télécharger l'application avec AUTOMGEN.

Module de communication



Paramétrage du module de communication

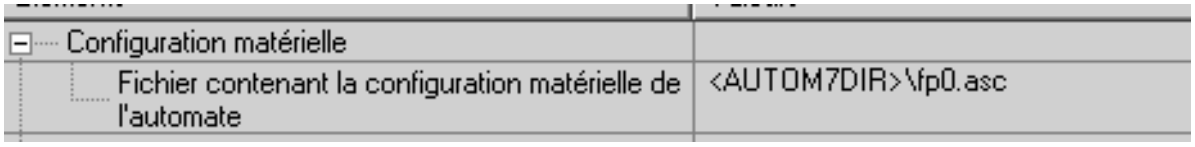
Post-processeur PANASONIC

Ce post-processeur permet de programmer les automates PANASONIC. Le logiciel FP WIN Pro 5 de PANASONIC doit être installé sur le PC pour pouvoir compiler et transférer le programme généré par AUTOMGEN vers les automates.

Le logiciel FP WIN PRO 5 doit être lancé avant de pouvoir lancer une compilation ou une exécution pour un automate PANASONIC avec AUTOMGEN.

Choix de la configuration de l'automate

L'élément « Configuration / Post-processeur / PANASONIC / Système / Configuration matérielle » du navigateur permet de choisir le type de l'automate.



Le fichier « .asc » est un fichier de configuration qui doit être généré avec l'atelier logiciel FP WIN Pro 5 de PANASONIC en utilisant la fonction « Exporter un projet » du menu « Projet » de ce logiciel.

Module de communication