

# Laboratoires du cours ELE4202

## Présentation du logiciel PL7-Pro

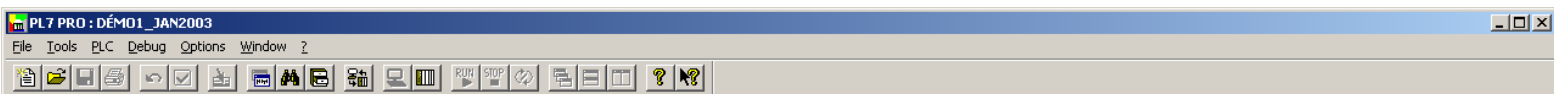
Par Cédric Demers-Roy  
Septembre 2003

Comme plusieurs automates disponibles sur le marché, l'automate TSX-Premium de la société Télémécanique (maintenant Schneider automation) possède son propre logiciel de programmation. Comparativement à la majorité des autres logiciels toutefois, PL7-Pro fournit une interface graphique pour la programmation des diagrammes à relais (Ladder) et pour les diagrammes Grafcet. Cette possibilité de programmation graphique offre plusieurs avantages, mais aussi quelques inconvénients, principalement au niveau de la façon d'introduire certains objets graphiques. Ce fascicule présente en bref l'utilisation de PL7-Pro pour remplir les objectifs du premier laboratoire du cours ELE4202.

La présentation sera divisée en cinq sections : l'interface générale du logiciel, la gestion des adresses et des variables, la programmation des diagrammes à relais, la programmation des Grafcets et le transfert des programmes vers l'automate.

### Interface générale du logiciel

Lors de son ouverture, le logiciel offre une interface semblable à celle de n'importe quel programme Windows. On retrouve une barre de menus, une barre d'outils et une fenêtre de travail. Pour atteindre nos objectifs, nous n'avons pas besoin de connaître tous les détails des menus et des outils. Seules quelques options nous seront utiles au cours du laboratoire. Cette section présente ces options et leur utilisation dans le cadre du laboratoire.

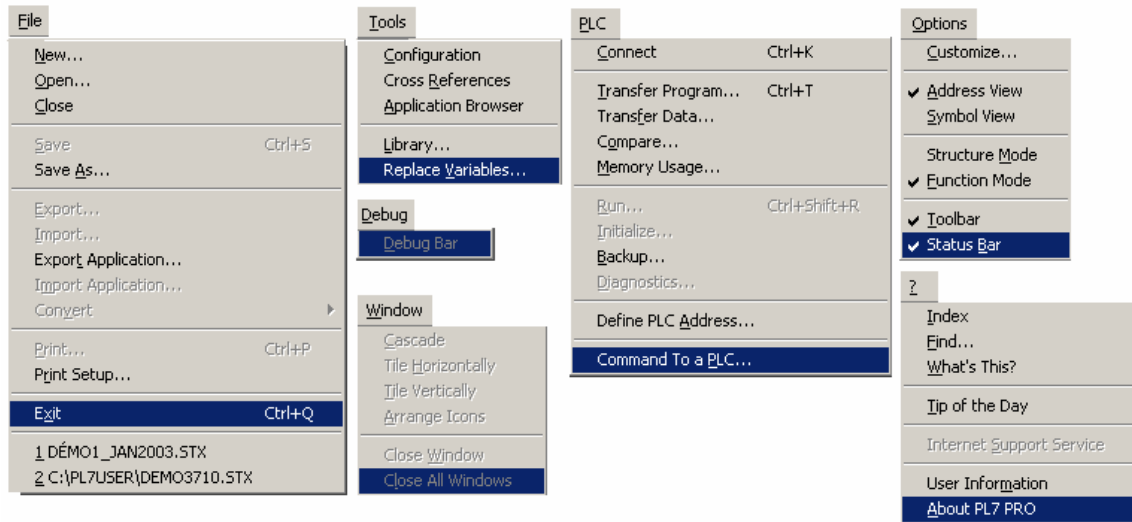


**Figure 1 : Interface générale du logiciel PL7\_pro**

Premièrement, la barre des menus. La figure 1 présente cette barre telle qu'elle se retrouve en haut de l'écran après l'ouverture du fichier « Démol\_jan2003.stx ». D'ailleurs, tous les fichiers réalisés à l'aide du logiciel PL7 portent l'extension STX. La figure montre que le logiciel comporte sept menus :

- File;
- Tools;
- PLC;
- Debug;
- Options;
- Window;
- Help (?).

Ces menus regroupent toutes les commandes du logiciel et reprennent donc, entre autres, toutes les fonctions associées aux boutons de la barre d'outils. La figure 2 illustre le contenu de chacun des menus, mais la plupart de ces options nous seront inutiles.



**Figure 2 : Contenu détaillé des menus de l'interface générale**

Tout d'abord, pour le menu « File », seules deux commandes seront utilisées : « Open » et « Save » (ou « Save As... »). En effet, la commande « New » ne devra jamais être utilisée pour une raison : la création d'un nouveau fichier débute par la configuration de l'automate, configuration qui demande des connaissances approfondies sur ses modules et sur ses caractéristiques. Or, dans le cours ELE4202, nous n'avons pas le temps de voir tout le détail de la configuration d'un automate. Cette tâche est relativement simple mais demande beaucoup de temps en recherche des informations nécessaires au sujet de l'automate. [Ainsi, pour le laboratoire, nous utiliserons toujours un fichier de base où la configuration est déjà effectuée.](#)

Le menu « Tools » ne sera à peu près pas utilisé, sauf peut être pour récupérer la fenêtre de navigation lorsque vous la fermez par erreur (choix « Application Browser » du menu). Nous verrons plus tard le détail de cette fenêtre de navigation. C'est à partir de cette fenêtre que vous pourrez accéder aux différentes sections de votre programme. Au début de la session, le choix « Configuration » peut être utilisé pour consulter la configuration utilisée pour l'automate et mieux comprendre l'usage des modules. [Toutefois, vous ne devez jamais modifier cette configuration au risque de ne pas pouvoir faire fonctionner votre programme par la suite...](#)

Le menu « Debug » ne sera jamais utilisé car il est souvent plus simple d'observer le comportement des différents diagrammes directement plutôt que de passer par les mémoires internes de l'automate. Cette option n'est vraiment utile que pour de très grosses applications où il est impossible de repérer à l'oeil toutes les liaisons logiques entre les diagrammes. Dans notre cas, il sera toujours possible de le faire.

Le menu PLC contient les fonctions d'accès à l'automate. Ces fonctions sont essentielles puisque ce sont elles qui font le lien entre le logiciel et l'automate. D'ailleurs, un aspect négatif du logiciel est qu'il n'offre pas la possibilité de simuler l'exécution des programmes avant de les transférer sur l'automate. La procédure de test de votre programme passe donc nécessairement par l'automate... Les choix que nous utiliserons dans le menu sont les suivants :

- **Transfer program** : permet le transfert du programme vers l'automate;
- **Connect** : permet de se connecter à l'automate pour voir l'évolution du programme;
- **Run** : lance l'exécution du programme sur l'automate (cette option devient « Stop » quand l'automate est déjà en mode d'exécution);
- **Initialize** : permet de réinitialiser les mémoires internes de l'automate sans avoir à l'éteindre ou à transférer le programme de nouveau;
- **Memory usage** : permet de connaître l'espace occupé par le programme et les données dans la mémoire interne de l'automate (utile dans le cas de gros programmes).

La plupart de ces options sont disponibles sur la barre d'outils dont nous verrons le détail bientôt.

Le menu « Options » contient les fonctions permettant d'ajuster la présentation de l'interface du logiciel. Normalement, nous n'avons pas à modifier ces paramètres. L'exception toutefois est la possibilité de choisir entre « Address View » et « Symbol View ». Ces options fixe le mode d'affichage des adresses dans les fenêtres de programmation. En effet, nous verrons plus loin qu'il est possible de « nommer » les adresses de l'automate pour mieux les identifier dans les programmes. Vous verrez que de retenir environ cinquante adresses numériques s'avère laborieux et qu'il est plus agréable de retenir le nom de chacune qui reflète son usage.

Pour les menus d'aide (« ? ») et « Window », il s'agit d'options standard à tout programme Windows que nous n'avons pas à présenter ici. Dernier point à noter à propos des menus : ils sont contextuels, c'est-à-dire qu'ils changent en fonction de la fenêtre active. Nous présenterons donc quelques choix particuliers disponibles seulement lors de la programmation lorsque nous en traiterons dans les prochaines sections

À présent, traitons brièvement de la barre d'outils. Les outils sont simplement l'attribution des fonctions les plus utilisées des menus à des boutons à accès rapide dont le simple clic effectue l'appel de la fonction (comme dans n'importe quel programme Windows). La figure 1 montre cette barre d'outils située dans l'interface, mais la figure 3 offre une meilleure vue de ces boutons.



Figure 3 : Détails de la barre générale d'outils du logiciel

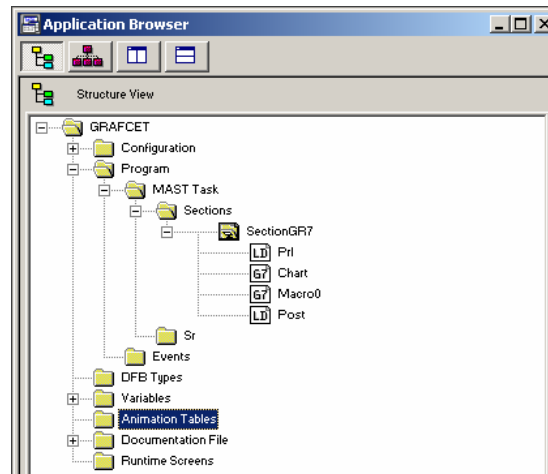
La liste qui suit présente chacun des outils avec sa fonction associée :

No	Fonction	Description
1	File->New	Créer un nouveau fichier (ne pas utiliser!)
2	File->Open	Ouvrir un fichier existant
3	File->Save	Sauver le programme en cours
4	File->Print	Imprimer le programme (vous ne pouvez imprimer que dans un fichier PDF au laboratoire car le logiciel génère une quantité phénoménale de pages...)
5	Edit->Undo	Annuler les modifications ( <a href="#">attention : cette commande annule TOUTES les modifications entre le moment d'entrée en mode d'édition et l'instant actuel...</a> )
6	Edit->Confirm	Confirmer les modifications (nécessaire pour pouvoir passer à une autre section de programme)
7	Go to	Aller rapidement d'une section à l'autre du programme (essayez, c'est pratique)
8	Tools->Application Browser	Afficher le navigateur d'application
9	Tools->Cross references	Chercher des références croisées entre les variables (permet de retrouver si une même variable est utilisée à plus d'un endroit dans votre programme)
10	Tools->Library	Fait appel à une librairie de fonctions (option inutilisée dans notre laboratoire)
11	PLC->Transfer	Effectue le transfert du programme vers l'automate
12	PLC->Disconnect	Déconnecte le PC de l'automate (nécessaire pour faire des modifications en mode local au programme; attention : les modifications réalisées en mode connecté peuvent ne pas être sauvegardées lors de la commande « save »... toujours travailler votre programme en mode local (déconnecté) )
13	PLC->Connect	Connecte le PC à l'automate (nécessaire pour voir l'évolution de votre programme et pour envoyer des commandes (« run », « stop », etc.) à l'automate)
14	PLC->Run	Commande à l'automate de lancer l'exécution
15	PLC->Stop	Commande à l'automate d'interrompre l'exécution
16	Utilities->Animate	Permet l'animation des données (commande inutilisée dont je ne connais pas la fonction...)
17	Window->Cascade	Contrôle l'affichage des fenêtres à l'écran.
18	Window->Tile Horizontally	
19	Window->Tile Vertically	
20	Help	Contrôle l'affichage des fenêtres d'aide
21	Help->What's this	

Avec ces outils, il est rare que vous aurez à aller dans les menus lors de votre programmation. Certaines options moins utilisées comme la configuration et la

consultation de l'utilisation de la mémoire de l'automate ne se retrouvent toutefois que dans les menus du logiciel.

Maintenant que l'interface générale est mieux connue, nous pouvons présenter la fenêtre de navigation qui servira à accéder aux différentes sections de votre programme. Cette fenêtre est représentée à la figure 4.



**Figure 4 : Fenêtre de navigation de l'application dans PL7-Pro v3.4**

Cette fenêtre ressemble à un explorateur de fichiers, mais elle sert plutôt à naviguer à travers les différents éléments d'un programme. Un programme STX standard se divise en sept répertoires principaux :

- Configuration
- Program
- DFB Types
- Variables
- Animation Tables
- Documentation File
- Runtime Screens

Le premier répertoire contient les paramètres de configuration de l'automate et du programme en cours. C'est là que vous trouverez les renseignements sur l'automate utilisé et sur la configuration du programme (nombre d'étapes permises, type de programme, etc.).

Le répertoire « Program » contient les fenêtres de programmation. C'est dans cette section que vous allez effectuer la quasi totalité de votre travail de conception logicielle. Ce répertoire contient lui-même un ou plusieurs sous répertoires selon le type de programme en cours. Sur la figure 4, il s'agit d'une application qui inclut du Grafcet, donc on retrouve une section gr7 qui contient plusieurs éléments : Prl, Chart, Macro0 et Post. Ces éléments sont les fenêtres qui contiennent le code du programme ouvert. L'icône à la droite de chaque élément indique s'il s'agit d'une programmation en

diagramme à relais (icône LD) ou en Grafcet (icône G7). Chaque élément a sa fonction comme nous le verrons au cours.

Le répertoire « DFB Types » n'est pas utilisé dans nos programmes. Normalement, il peut contenir le code de blocs fonctionnels élaborés par l'utilisateur et inclus dans l'application en cours. Cette inclusion fait partie de la programmation avancée et nous ne verrons pas ce cas dans le laboratoire.

Le répertoire « Animation Tables » permet l'ajout de tables servant à suivre l'évolution des données en cours d'exécution d'un programme sur l'automate. Par exemple, si j'inclus l'adresse %ID3.0, je pourrai voir l'évolution de son contenu lors d'un test du programme. Si je regarde la table d'attribution des adresses (nous verrons plus loin), je constate que cette adresse contient le décompte des impulsions du compteur rapide numéro 0. Sachant que ce compteur est relié à l'encodeur en X du montage de palan, le suivi du contenu de cette adresse m'indique le parcours du palan selon l'axe X (déplacement horizontal). Ainsi, une table d'animation est très pratique pour suivre les valeurs des mémoires lors des tests, particulièrement lorsque la valeur n'est pas binaire...

Le répertoire « Documentation File » permet de contrôler la documentation que PL7 génère lors de l'impression. C'est à cet endroit que vous pouvez inclure ou rejeter une section du programme pour son impression. Malheureusement, la gestion du papier n'est pas très efficace dans le logiciel. Pour vous en convaincre, faites générer le PDF pour le fichier « Démo3\_jan2003.stx ». Ce fichier pourtant très simple génère 48 pages si toutes les options de documentation sont sélectionnées... Typiquement, votre programme final pour le projet du laboratoire fera sans doute plus de 150 pages! C'est pour cette raison que nous permettons seulement la génération d'un fichier PDF au laboratoire. De cette façon, vous avez la documentation et vous pouvez choisir les pages à imprimer chez vous ou dans les laboratoires du service informatique.

Le dernier répertoire, « Runtime Screens », contient une interface graphique optionnelle qu'il est possible de générer pour chaque programme. Typiquement, cette interface permet d'afficher des informations sur l'exécution du programme qu'il est impossible ou difficile d'obtenir avec la console de contrôle. Pour le projet, vous devrez programmer une interface de base afin d'en voir les avantages. La programmation de cette interface sera couverte plus tard durant la session et un autre document vous sera remis à ce moment.

Avec les notions que nous venons de couvrir au sujet de l'interface du logiciel, vous pourrez évoluer plus efficacement lors de la programmation de votre projet. Évidemment, le logiciel offre beaucoup plus d'options que ce que nous venons de couvrir, mais l'essentiel des notions a été vu pour vous rendre autonome dans l'utilisation de son interface.

## Présentation de l'automate

L'automate est constitué d'un ensemble de différents modules que nous pouvons utiliser lors de la programmation. Cette section présente brièvement chacun des modules. La figure 5 montre la fenêtre de configuration matérielle de la démonstration 1 (identique à celle de tous les autres fichiers disponibles). Nous voyons que l'automate est constitué d'une suite de huit modules dont sept possèdent une adresse distincte. Ces adresses sont très importantes puisqu'elles fixent l'identification des modules dans le logiciel. Par exemple, une adresse débutant par « 1 » est nécessairement associée au module 1, donc au module des entrées logiques. Le reste de l'adresse indique le type d'adresse et l'élément exact à atteindre dans le module.

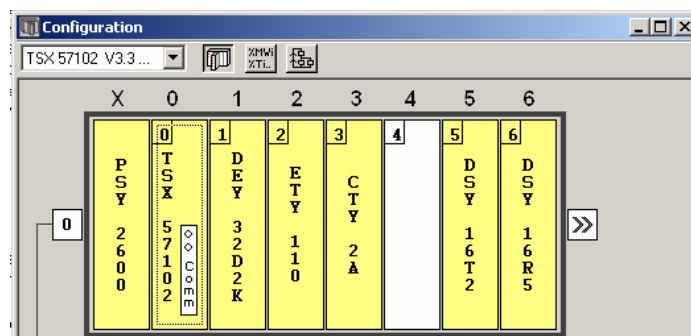


Figure 5 : Présentation des modules de l'automate et de leur adresse

Le tableau qui suit explique brièvement chacun des modules.

Adresse	Nom	Description
X	PSY2600	Module d'alimentation. Ce module est indépendant et ne peut pas être adressé de l'extérieur de l'automate. Il gère l'alimentation de tous les modules de l'automate.
0	TSX57102	Module processeur. Ce module contient le processeur de l'automate ainsi que les mémoires de programmes et de données. Tous les programmes seront transférés à ce module qui gère le fonctionnement de tout le reste de l'automate. Les adresses internes de ce module ne doivent pas être utilisées sans savoir vraiment ce que nous faisons car le moindre changement peut bloquer tout le programme.
1	DEY32D2K	Module d'entrées logiques. Ce module permet de brancher les capteurs binaires externes à l'automate. Puisqu'il est à l'adresse 1, toutes les entrées de nos programmes auront une adresse débutant par 1. En fait, l'identificateur « I » sera aussi ajouté pour signifier qu'il s'agit d'une entrée. L'adresse des entrées logiques sera donc %I1.# où # indique l'entrée en question.
2	ETY110	Module de communication Internet. Ce module permet d'accéder à l'automate via une interface Internet. Toutefois, un problème de configuration de ce module bloque l'accès à

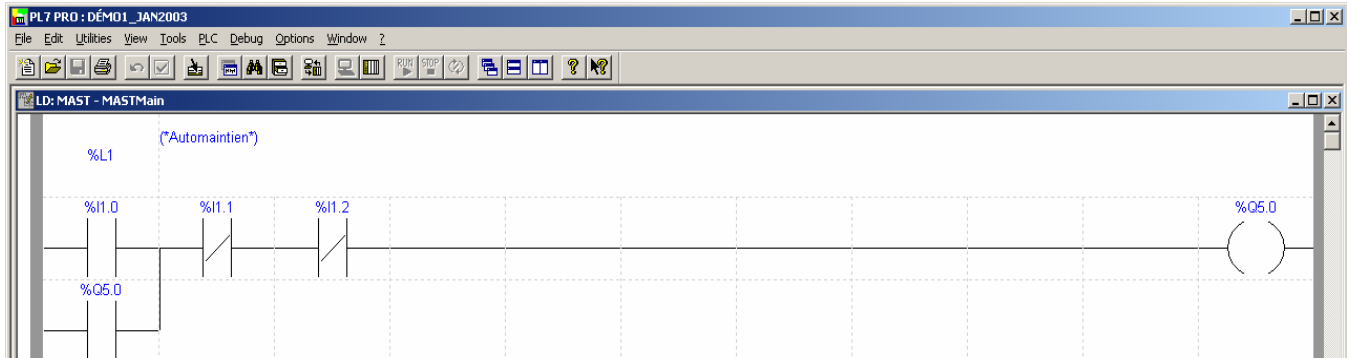
		l'automate par plus d'un usager, nous obligeant donc à ne pas l'utiliser. Vous ne verrez donc jamais d'adresses débutant par « 2 » dans nos programmes.
3	CTY2A	Module de compteurs rapides. Ce module permet d'acquérir des signaux oscillants rapidement entre la valeur « 0 » et « 1 ». Typiquement, ce module sert à lire les signaux provenant d'encodeurs optiques effectuant l'acquisition de la position en rotation d'un axe. Dans notre cas, les positions horizontales et verticales du palan seront lues par ce module. Puisqu'il est à l'adresse 3, nous y accéderons toujours via des adresses débutant par ce nombre. Puisqu'il s'agit encore d'une entrée, l'adresse comportera un « I », mais aussi un « D » pour indiquer qu'il s'agit d'une entrée sur 16 bits plutôt que d'une entrée binaire (1 bit). Le format sera donc %ID3.#. De plus des sorties %Q3.# permettront d'accéder à des bits de fonctionnement de ce module.
4	N/D	Aucun module à cette position.
5	DSY16T2	Module de sorties faible puissance. Ce module comporte 16 relais pouvant nous servir à alimenter des actionneurs à faible puissance comme des LEDs ou de très petits moteurs. Dans notre cas, ces 16 sorties sont reliées au 16 LEDs de la console de commande. L'adresse de ces sorties sera au format %Q5.#. L'identificateur « Q » indique qu'il s'agit d'une sortie.
6	DSY16R5	Module de sorties en puissance. Ce module comporte 16 relais pouvant nous servir à alimenter des actionneurs à moyenne puissance comme des moteurs DC moyens ou de petits électroaimants. Dans notre cas, ces 16 sorties sont reliées aux deux moteurs permettant les déplacements (une sortie pour chaque direction de déplacement) et à l'électroaimant permettant le transport des pièces. L'adresse de ces sorties sera au format %Q6.#. L'identificateur « Q » indique qu'il s'agit d'une sortie.

Physiquement, chacune des adresses sera reliée à des éléments internes ou externes de l'automate comme le laisse entendre le tableau. Lors de l'utilisation de l'automate dans un contexte industriel, un tableau d'assignation des adresses est toujours fourni avec ce dernier afin de connaître la liaison physique des adresses avec les éléments externes (capteurs et actionneurs). Cette assignation vous sera remise avec l'énoncé du projet. Si vous aviez à tout réaliser le travail, vous devriez vous-même relier chacun des capteurs et chacune des sorties à une entrée ou une sortie des modules de l'automate et construire du même coup le tableau des assignations. Dans notre cas, afin de faciliter et d'uniformiser le travail, cette assignation est déjà réalisée et sera la même pour tout le monde.



## Gestion des adresses et des variables

Le but de ce logiciel étant la programmation de l'automate, il doit fournir une méthode pour accéder à ce dernier dans nos programmes. Cette méthode, c'est l'utilisation des adresses internes de la mémoire de l'automate. Tout objet de programmation sera associé à une adresse, que ce soit un contact, une sortie ou une variable interne.



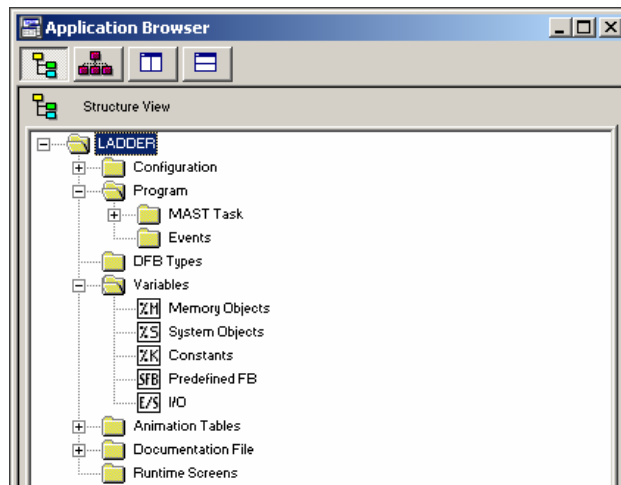
**Figure 6 : Exemple de programmation d'un diagramme à relais**

La figure 6 donne un exemple de code programmé en diagramme à relais. À la gauche, on voit une suite de contacts représentant la condition logique et à la droite une bobine représentant la sortie de la condition logique. Vous voyez au-dessus un identificateur qui commence par le symbole « % », ceci indique qu'il s'agit d'une adresse. En regardant ce petit exemple, nous voyons une condition qui se base sur quatre entrées associées aux adresses %I1.0, %I1.1, %I1.2 et %Q5.0 et une sortie associée à %Q5.0. La fonction exacte de ce diagramme a été vue au cours, il s'agit d'un automaintien de la sortie %Q5.0 basée sur les entrées %I1.0 à %I1.2. Tout ce qu'il faut retenir, c'est que le fait d'utiliser ces adresses indique à l'automate quelles entrées et quelle sortie nous voulons utiliser. Sachant que les entrées %I1.0 à %I1.15 sont branchées physiquement aux interrupteurs de la console de commande et que la sortie %Q5.0 est branchée physiquement à la LED 0 de la console, nous pouvons déterminer l'effet de ce petit exemple : allumer et maintenir la LED 0 dès que l'interrupteur 0 est enclenché et l'éteindre si l'interrupteur 1 ou 2 est enclenché.

L'automate contient beaucoup d'adresses différentes qu'il peut être difficile de retenir au début. Afin de faciliter la programmation, le logiciel fournit une manière optionnelle de faire appel aux adresses : la notation symbolique. En allant dans le répertoire « Variables » de la fenêtre de navigation de l'application, vous pouvez accéder aux tableaux d'assignation des adresses de l'automate. Ce répertoire contient cinq tableaux contenant chacun des adresses spécifiques (voir figure 7) :

- Memory Objects : adresses des variables disponibles pour vos programmes;
- System Objects : adresses des objets systèmes (ne pas modifier rien dans ce tableau!);
- Constants : adresses des constantes disponibles pour vos programmes;

- Predefined FB : adresses des blocs fonctionnels disponible pour les diagrammes à relais;
- I/O : adresses des entrées / sorties des différents modules de l'automate.



**Figure 7 : Répertoire des variables de la fenêtre de navigation de l'application**

Avec ces tableaux, il est possible de retrouver toutes les adresses internes de l'automate. Ceci peut nous permettre d'abord de connaître les adresses disponibles, mais surtout d'associer un symbole aux adresses couramment utilisées. Par exemple, si nous voulons nommer les adresses des interrupteurs, nous devons aller dans le tableau « I/O » et de choisir le module correspondant aux entrée (le module 1). Ceci affiche un tableau qui contient toutes les adresses disponibles dans ce module. Nous n'avons ensuite qu'à repérer les adresses %I1.0 à %I1.15 et d'entrer une valeur dans le champ « Symbol » pour chacune des adresses. Sur l'exemple de la figure 8, nous avons donné le nom « Inter1 » pour l'adresse %I1.0. Il en va de même pour toutes les autres adresses. Notez que le nom du symbole ne commence pas par le symbole « % » puisqu'il ne s'agit pas d'une adresse mais d'un symbole. Pour l'automate, le nom « Inter1 » correspondra désormais directement à l'adresse %I1.0.

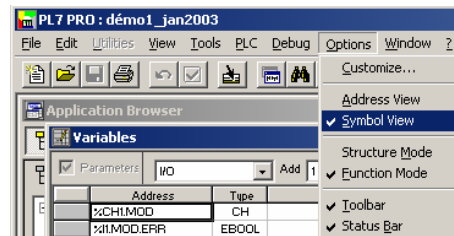
Address	Type	Symbol	Comment
%CH1MOD	CH		
%I1MODERR	EBOOL		
%M1MOD	WORD		
%M1MOD.1	WORD		
%M1MOD.2	WORD		
%CH10	CH		
%I10.ERR	BOOL		
%I1.0	EBOOL	Inter1	
%M1.0	WORD		
%M1.0.1	WORD		
%M1.0.2	WORD		
%K1.0	WORD		
%K1.0.1	WORD		

**Figure 8 : Exemple d'attribution d'un symbole à une adresse d'entrée**

Une fois les symboles attribués, vous aurez toujours la possibilité d'utiliser le symbole plutôt que l'adresse lors de la programmation. Pour ce faire, simplement retirer le symbole « % » au début de l'identificateur d'un objet et entrer le nom du symbole désiré. Lorsque vous validez votre choix, PL7 retrouvera l'adresse correspondante et l'affichera à la place du symbole. Toutefois, si vous préférez voir les symboles plutôt que les

adresses (plus facile à suivre), simplement activer l'option « Symbol View » dans le menu « Options » (voir figure 9). Avec cette option, toute adresse à laquelle est associée un symbole sera affichée à l'aide de son symbole. Un défaut de l'interface fait que cette option est souvent désactivée lorsque vous changez de fenêtre dans l'application. Il est donc bon de connaître le raccourci de cette option :

- Pour la vue des symboles : Ctrl-F ;
- Pour la vue des adresses : Ctrl-E .



**Figure 9 : Menu de sélection entre la vue des adresses et la vue symbolique**

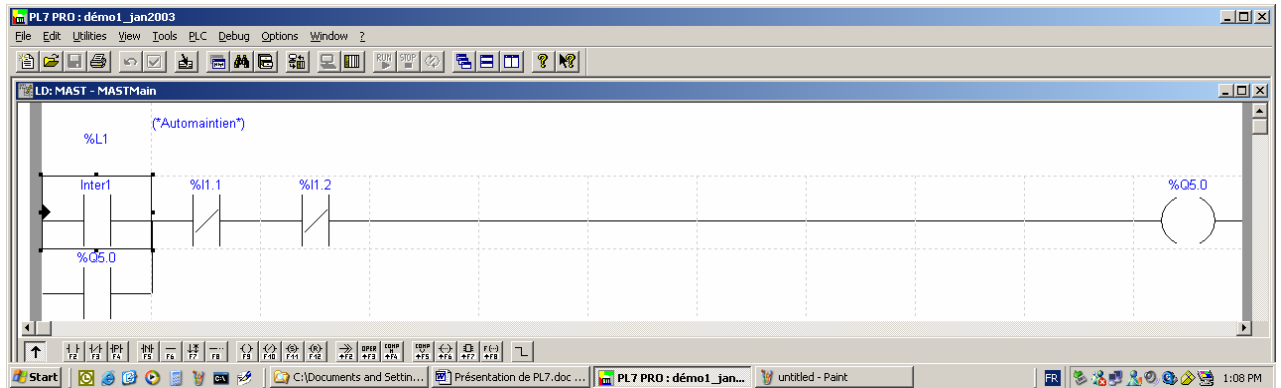
L'option de vue symbolique est excessivement pratique pour faciliter la lecture des diagrammes, il est donc fortement conseillé de donner un symbole à toutes les adresses fréquemment utilisées. Habituellement, les adresses fréquemment utilisées sont :

- les entrées;
- les sorties;
- les variables que vous utilisez (ex. : %M0, %MD0, etc.);
- les blocs fonctionnels que vous utilisez (ex. : %T0).

## Programmation des diagrammes à relais

Une fois les adresses correctement gérées, nous pouvons traiter de la programmation comme telle d'une application dans PL7. Le premier type de programmation que nous utiliserons est le diagramme à relais. Nous verrons au cours les notions relatives à cette programmation. Notre but ici est d'illustrer l'implantation d'un diagramme à relais dans PL7.

Tout d'abord, nous devons ouvrir la fenêtre de programmation. À partir de la fenêtre de navigation de l'application, aller dans le répertoire « Program » et ouvrir le répertoire avec ses sous répertoires jusqu'à la fin de l'arborescence. Vous retrouverez alors un ou plusieurs éléments avec un icône « LD » ou « G7 » à leur gauche. Les éléments avec un icône « LD » sont des fenêtres de programmation en diagramme à relais, simplement en choisir un et l'ouvrir. Dans l'exemple de la démonstration 1 (« Démo1\_sep2003.stx »), il n'y a qu'un seul élément, une fois ouvert, vous obtenez une fenêtre comme celle de la figure 10. L'élément nouveau de cette fenêtre est la barre d'outils au bas. Ce sont les éléments qui serviront à la programmation des diagrammes à relais. Le tableau qui suit donne une brève explication de chaque élément.



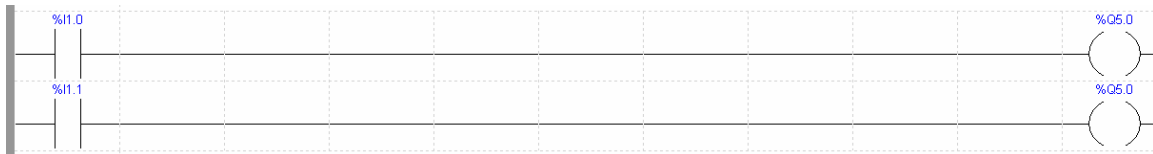
**Figure 10 : Fenêtre de programmation en diagramme à relais**

Boutons	Description
	Contact normalement ouvert (vrai si contenu de l'adresse est vrai)
	Contact normalement fermé (vrai si contenu de l'adresse est faux)
	Contact à front montant (vrai si contenu de l'adresse passe de faux à vrai)
	Contact à front descendant (vrai si contenu de l'adresse passe de vrai à faux)
	Lien horizontal d'une case (pour effectuer les connexions)
	Lien vertical entre deux lignes (pour effectuer les connexions)
	Lien horizontal rapide (trace un lien jusqu'à une case du bout de la ligne)
	Bobine en logique positive (agit seulement si la ligne est vraie)
	Bobine en logique négative (agit tant que la ligne est fausse)
	Bobine d'enclenchement (active la bobine correspondante jusqu'à sa désactivation)
	Bobine de déclenchement (désactive la bobine correspondante)
	Lien de saut (saut vers un autre « rung » au cas où la ligne est trop courte pour la condition logique)
	Fonction « Operate » (pour effectuer une opération sur des variables)
	Fonction « Compare » (pour effectuer une comparaison entre des variables)
	Comme précédemment, mais avec plusieurs sorties (ne fonctionne qu'avec des variables de type « Mot » [« Word »])
	Fonctions spéciales d'appel de fonction (inutilisé dans nos programmes)
	Blocs fonctionnels (temporisateur et autres)
	Fonction complexe (inutilisé dans nos programmes)
	Lien rapide (permet de tracer des liens entre les éléments en cliquant sur les cases à lier (plus utile que lien vertical présenté plus tôt))

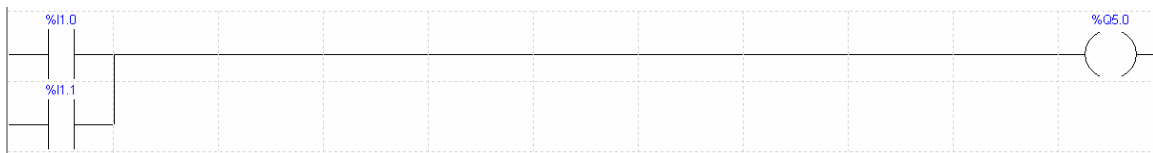
Pour entrer un élément, vous devez être en mode de modification. Pour y entrer, simplement double cliquer sur une ligne. Tous les éléments deviendront rouges, ce qui indique que le « rung » est en cours de modification. Ensuite, sélectionnez un élément et placez-le à l'endroit désiré dans le « rung ». Le logiciel demandera alors l'adresse à associer à l'élément, spécifiez donc une adresse ou un symbole et validez avec la touche

« Enter » sur le clavier. Si le logiciel refuse de valider, c'est que l'adresse ou le symbole est invalide. Pour continuer sans spécifier d'adresse, simplement appuyer sur la touche « ESC » du clavier. **Notez que vous ne pourrez valider les modifications si les adresses ne sont pas entrées et validées.** Pour valider le diagramme, simplement appuyer sur la touche « Enter », si tous les éléments reprennent leur couleur noire et bleue, c'est que la validation est effectuée; le cas échéant, il y a une erreur que vous devez corriger.

Un « rung » est une division du diagramme à relais. Il est constitué d'un maximum de 16 lignes avec un identificateur et un commentaire au début. Il est très important de bien comprendre la notion de « rung ». Si vous utilisez la démonstration 1, vous avez vu en ouvrant la fenêtre de programmation que toutes les lignes étaient collées les unes contre les autres. Or, quand vous êtes entré en mode de modification, remarquez que les lignes après le diagramme en cours sont disparues... En fait, elles sont descendues 16 lignes plus bas afin de vous donner de l'espace pour programmer. **Cette notion est très importante : si vous placez toutes vos lignes les unes après les autres, vous ne pourrez pas modifier facilement votre programme.** Il faut toujours prévoir de l'espace. Dans la démonstration 1, vous pouvez voir que la majorité des « rung » ne contiennent en fait qu'une seule bobine de sortie. Cette pratique est à préconiser pour réaliser un bon programme : une seule sortie par « rung ». Dans le pire des cas, plusieurs sorties assurant la même fonction peuvent être regroupées ensemble si chacune ne prend pas trop de place. Le cas échéant, vous ne pourrez plus ajouter de conditions à une bobine de sortie. **D'ailleurs, ne réutilisez jamais deux fois la même adresse de sorties à deux endroits différents** car le logiciel ne saurait plus quelle est la condition d'activation de la sortie. Par exemple, supposons que je veux que les adresses %I1.0 ou %I1.1 active la sortie %Q5.0 (donc  $\%Q5.0 = \%I1.0 + \%I1.1$ ). Si je programme la séquence de la figure 10, je n'obtiendrai pas le comportement désiré puisque seule la dernière condition sera prise en compte par le logiciel (donc la suite  $\%Q5.0 = \%I1.0; \%Q5.0 = \%I1.1$ ). La bonne façon est de faire le diagramme de la figure 12 à la place.



**Figure 11 : Exemple de mauvaise programmation en diagramme à relais**



**Figure 12 : Exemple de bonne programmation en diagramme à relais**

En résumé, toujours utilisez chaque adresse de sortie une seule fois et toujours placer autant que possible une seule bobine de sortie par « rung ». La meilleure façon de prouver la nécessité de programmer selon cette technique est l'utilisation des moteurs dans le laboratoire. Par exemple, dans votre projet, une multitude de situations demanderont à ce que le moteur de déplacement à gauche soit activé, donc plusieurs

conditions activeront la sortie en question (vous verrez laquelle dans l'énoncé). Si vous regroupez toutes les sorties des moteurs dans le même « rung », vous ne pourrez pas ajouter de nouvelles conditions puisque l'espace sera déjà occupé. Toutefois, si un « rung » contient seulement une bobine de moteur, alors vous aurez amplement d'espace.

## Programmation des Grafjets

Le deuxième type de programmation disponible est le Grafjet (Graphe Fonctionnel de Cheminement Étape Transition). Encore une fois, nous verrons le détail de cette programmation au cours. Le but de cette section est d'illustrer l'implantation de ce genre de diagramme dans PL7. Comme pour les diagrammes à relais, vous devez aller dans le répertoire « Program », mais cette fois ce sont les éléments avec l'icône « G7 » qui nous intéressent. Pour avoir un exemple, ouvrez la démonstration 2 ou 3 (Démon2\_sep2003.stx) et ouvrez le Grafjet contenu dans la démonstration. Vous obtiendrez une fenêtre comme celle de la figure 13.

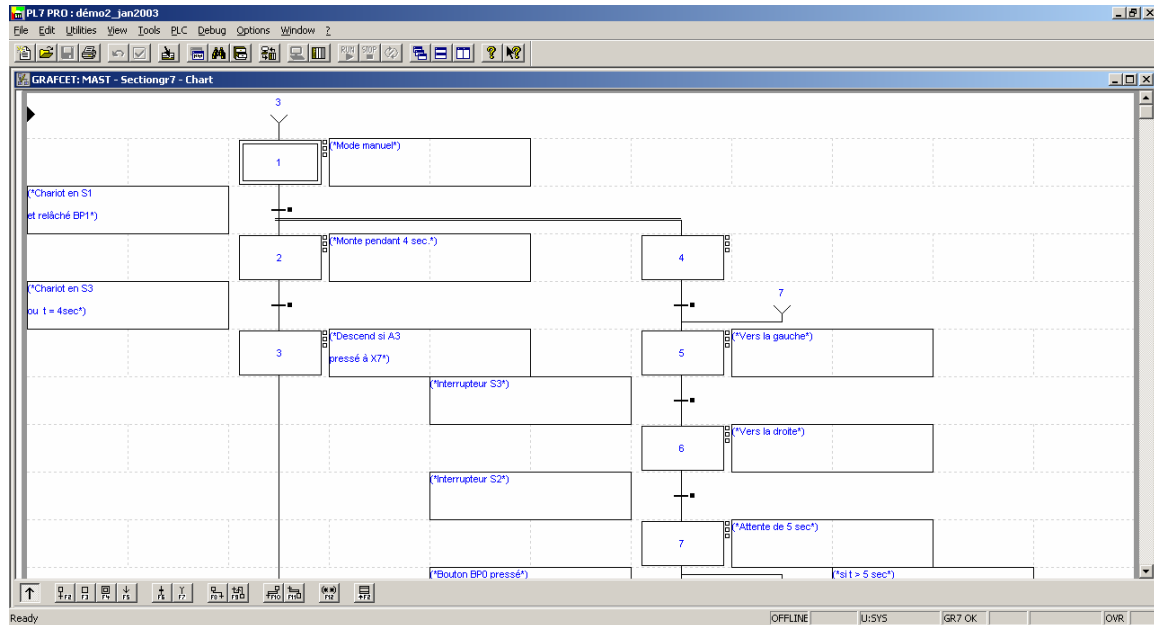









Figure 13 : Exemple de fenêtre de programmation en Grafjet

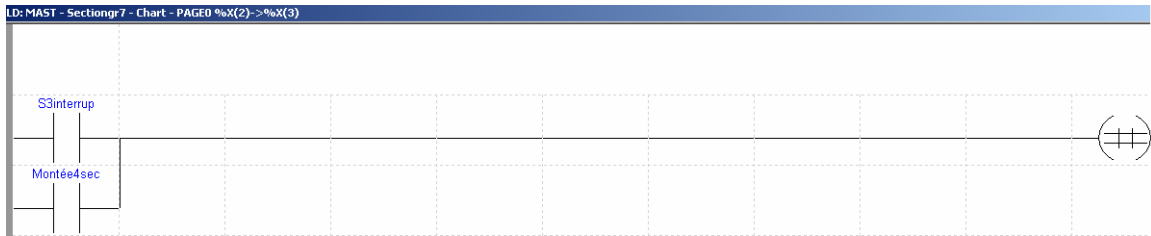
Cette fenêtre ressemble beaucoup à la fenêtre de programmation en diagramme à relais, mais les outils au bas de la fenêtre sont différents. Le tableau qui suit illustre les différents outils.

Boutons	Description
	Insérer une étape immédiatement suivie d'une transition
	Insérer une étape
	Insérer une étape initiale
	Insérer un lien sortant (pour effectuer un lien vers un entrant)
	Insérer une transition

	Insérer un lien entrant (pour effectuer un lien à partir d'un sortant)
	Insérer un lien étape transition (cliquer sur l'étape, puis sur la transition)
	Insérer un lien transition étape (cliquer sur la transition, puis sur l'étape)
	Insérer une convergence en « ET » (toujours placer la transition à gauche pour effectuer ce type de lien car vous devez cliquer sur l'étape en premier et faire le lien par la gauche ensuite...)
	Insérer une convergence en « ET » (toujours placer la transition à la gauche pour effectuer ce type de lien car vous devez cliquer sur la transition en premier et faire le lien par la droite ensuite...)
	Insérer une case de commentaire (limiter ce genre de case au minimum car elles occupent de l'espace...)
	Insérer une macroétape (nous verrons le détail de ce type d'étape au cours)

La programmation se fait dans le même style que les diagrammes à relais : vous devez d'abord entrer en mode modification en double cliquant sur une page du Grafcet. Ce faisant, tous les éléments de la page deviennent rouges pour indiquer la présence en mode modification. Dans ce mode, vous pouvez insérer tous les éléments de votre Grafcet en faisant attention à leur disposition. En effet, l'espace disponible dans une page est de 11 colonnes par 15 lignes et vous ne pouvez sortir de la page ni l'agrandir. Une fenêtre peut contenir un maximum de 8 pages, donc vous devez bien prévoir la disposition des Grafcets dans la page. De plus, vous pouvez voir dans la liste précédente la particularité de certains éléments qui demandent une saisie à droite ou à gauche et qui spécifient l'ordre de sélection des éléments. Ces particularités sont plutôt ennuyantes, mais nous devons nous y faire...

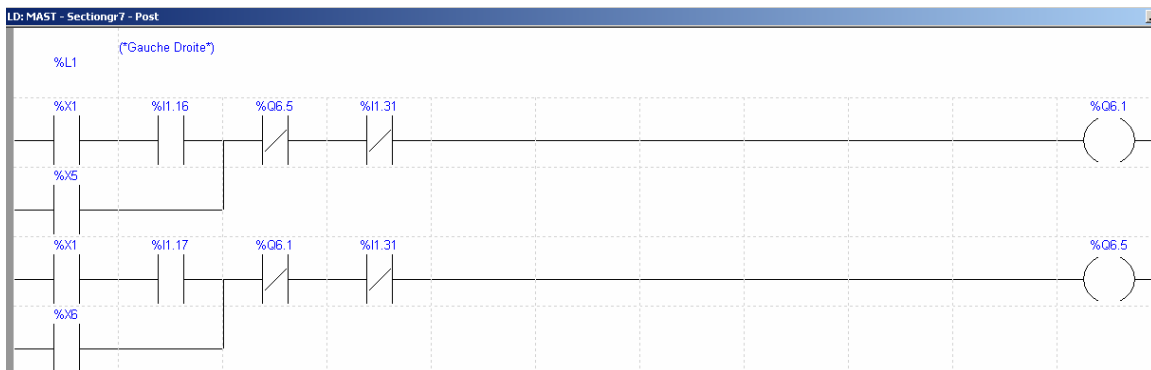
Quand la saisie graphique du Grafcet est terminée, il n'est pas complètement programmé encore : vous devez entrer les conditions des transitions et les actions associées aux différentes étapes. Pour ce faire, vous devez d'abord valider l'entrée du Grafcet en appuyant sur la touche « Enter » du clavier, si tous élément redeviennent bleus et noirs, la validation est effectuée; s'ils restent rouges, il y a un problème que vous devez régler avant de continuer. Lorsque la validation est effectuée, l'entrée des conditions des transitions peut se faire en double-cliquant sur le petit carré juste à côté de la transition à programmer. Cette action devrait ouvrir une fenêtre au bas de l'écran avec les outils de programmation en diagramme à relais. En effet, toutes les conditions des transitions sont programmées en diagramme à relais avec la particularité que la seule bobine de sortie disponible est une bobine avec un symbole « # » à l'intérieur. Cette bobine symbolise la validité de la transition. Vous devez donc entrer la condition de validité de la transition. Par exemple, la figure 14 illustre la transition entre les étapes 2 et 3 du Grafcet de la démonstration 2. La condition est la suivante : transition validée si limite S3 atteinte ou si le délai de montée de 4 secondes est atteint. La première partie est fonction de l'interrupteur S3 et la deuxième d'un délai interne associé à un temporisateur quelconque. Chaque transition est programmée de la même façon. Notez toutefois qu'une seule transition peut être ouverte à la fois, vous devez donc fermer la fenêtre de programmation de la condition pour en ouvrir une autre...



**Figure 14 : Exemple de programmation d'une transition du Grafcet**

Une fois les transitions programmées, il reste encore à fixer les actions associées aux différentes étapes du Grafcet. En effet, le Grafcet ne vise qu'à programmer la séquence logique des actions, pas les actions comme telles. Pour les actions, nous devons activer les sorties qui nous intéressent. Comme nous avons vu plus tôt dans la section précédente, l'activation d'une sortie se fait via la bobine qui lui est associée, donc via son adresse. Il s'agit donc de programmation en diagramme à relais. Nous devons donc ouvrir la fenêtre de programmation en diagramme à relais correspondant aux actions du Grafcet. Cette fenêtre se trouve avec l'objet « G7 » dans le répertoire « Program » du navigateur de l'application. En effet, nous avons ouvert l'objet « G7 » pour programmer le Grafcet, mais il y avait au moins deux autres éléments dans le répertoire : l'un appelé « Prl » et l'autre « Post », tous deux affublés de l'icône « LD ». Le premier servira à programmer des opérations à faire avant de balayer le Grafcet, tandis que le deuxième sert à programmer les actions associées aux étapes de ce dernier. Ainsi, pour programmer les actions, ouvrez la fenêtre « Post ». La fenêtre obtenue est une fois de plus la fenêtre de programmation en diagramme à relais. Nous pouvons maintenant programmer les actions.

Rappelez-vous du conseil visant à regrouper les bobines chacune dans son propre « rung » : cette démonstration en est un mauvais exemple puisque les bobines %Q6.1 et %Q6.5 se retrouvent dans le même « rung » (voir figure 15). Dans ce cas, ce n'est pas critique car il s'agit d'une démonstration à laquelle nous n'ajouterons pas de nouveaux comportements, toutefois ce ne sera pas le cas de votre projet où vous devrez certainement ajouter de nouveaux éléments en cours de route.



**Figure 15 : Exemple de programmation des actions associées au Grafcet**

Vous voyez sur la figure 15 les conditions d'activation des moteurs gauche et droit, associés respectivement aux adresses %Q6.1 et %Q6.5. Dans ces conditions, vous



pouvez voir un type d'adresse que vous n'aviez pas rencontré encore : %X#. Ce type d'adresse effectue la liaison avec les étapes du Grafcet. Dans le cas de la démonstration, nous voulions que l'étape 5 active le moteur de gauche et l'étape 6 le moteur de droite (voir figure 13). Nous voyons clairement cette action à la figure 15 en notant les adresses %X5 et %X6 dans les conditions d'activation des sorties %Q6.1 et %Q6.5. Notez aussi que l'étape 1 (%X1) permet aussi d'activer les sorties des moteurs. Dans le Grafcet, nous parlions d'un mode « manuel » à l'étape 1. Physiquement, nous voyons dans les conditions d'activation que l'étape 1 se trouve à côté des adresses %I1.16 et %I1.17. Sachant que ces adresses sont liées aux boutons de commande gauche et droit de la console, il est clair que l'étape 1 active le mode manuel où ces boutons peuvent activer le moteur. Le reste de chaque condition est une protection de sécurité : %I1.31 est liée au bouton d'arrêt d'urgence et les conditions %Q6.1 et %Q6.5 assurent l'exclusion mutuelle des deux actions, c'est-à-dire que le moteur ne pourra jamais aller à gauche et à droite en même temps... [D'ailleurs, ceci montre que vous pouvez utiliser une sortie comme une entrée dans le cas où la lecture de sa valeur est utile comme condition logique.](#)

À présent, nous avons couvert l'essentiel de la programmation en Grafcet. Il nous reste seulement à voir comment envoyer le résultat à l'automate afin de tester le programme dans un contexte d'utilisation. [En effet, le logiciel ne fournit pas de fonction de simulation, donc vous devrez toujours transférer le programme à l'automate pour le tester.](#) **Ceci implique la nécessité de programmer les fonctions de sécurité au fur et à mesure de l'élaboration du programme afin de toujours pouvoir interrompre les actions avant qu'elles ne causent des dommages.**

## Transfert des programmes vers l'automate

Le test des programmes passe nécessairement par le transfert du programme vers l'automate. Pour se faire, vous allez passer par le lien série qui relie l'ordinateur au module de contrôle de l'automate (module 0). La seule condition pour pouvoir transférer le programme en cours est qu'il soit entièrement validé, c'est-à-dire qu'il ne contienne pas de partie non validée (en rouge). Lorsque c'est le cas, simplement cliquer sur le bouton « Transfer » de la barre d'outils du logiciel (voir la section sur la présentation de l'interface pour trouver cet outil). Le logiciel demande alors si vous voulez effectuer un transfert PLC->PC ou PC->PLC. Sachant que PLC représente l'automate (Programmable Logic Controller) et que PC représente l'ordinateur (Personal Computer), le choix est clair... Le transfert du PLC vers le PC est utile dans un contexte industriel où un programmeur de système veut consulter le programme d'un automate qui n'est pas constamment relié à un ordinateur : il branche un câble entre l'automate et un PC portable et retire les informations de l'automate. Dans notre cas, nous voulons transférer le code vers l'automate, donc PC->PLC. Le transfert débute alors et prend un certain temps car le lien série utilise une vitesse de communication assez faible. Toutefois, le temps de transfert dépend très peu de la taille de votre programme car le logiciel transfère beaucoup d'informations de configuration avec le programme. D'ailleurs, vous verrez que le code final de votre projet occupe à peine 10 Ko, ce qui est très facile à transférer sur disquette. Lorsque le transfert est terminé, vous devez maintenant vous connecter sur l'automate pour le contrôler. Pour y arriver, cliquer sur

l'outil « Connect » de la barre d'outils. À partir de ce moment, si vous entrez dans une fenêtre de programmation, vous verrez plusieurs cases noires. Cette couleur indique la validité de la condition en question. Déjà maintenant vous pouvez effectuer certains tests sur vos programmes. Toutefois, vous noterez que même si la condition d'une ligne est validée (i.e. toutes les cases sont noires), la sortie reste blanche. Ceci est normal car l'automate est automatiquement en mode « Stop » lorsqu'un programme vient d'être transféré. Pour que les sorties puissent s'activer, la dernière action à effectuer est de cliquer sur le bouton « Run » de la barre d'outils. Le logiciel demande alors une confirmation. Cette fois, les lignes validées vont valider aussi la sortie correspondante. Vous pouvez maintenant tester votre programme.

Quand vous avez terminé vos tests, ou pour effectuer une modification à votre code, vous devez interrompre l'exécution. Pour ce faire, cliquer sur le bouton « Stop » de la barre d'outils. Si vous voulez modifier votre code, vous devez aussi vous déconnecter de l'automate afin de ne pas travailler dans sa mémoire. En effet, lorsque vous travaillez en mode connecté, vous effectuez les modifications directement dans la mémoire de l'automate. Ce faisant, l'action de sauver votre programme peut ne pas tenir compte de ces modifications... Pour cette raison, toujours vous déconnecter de l'automate en cliquant sur le bouton « Disconnect » de la barre d'outils. Quand vos modifications seront terminées, vous devrez retransférer le code et suivre la procédure de mise en route à nouveau.

En résumé, la procédure de mise en route est :

- 1) Transfer;
- 2) Connect;
- 3) Run;

et celle de modification est :

- 1) Stop;
- 2) Disconnect.

En terminant, l'option de transfert de l'automate vers le PC peut vous jouer un vilain tour : l'automate conserve ses mémoires même lorsqu'il est éteint. Ainsi, une équipe qui arrive au laboratoire peut récupérer le code de l'équipe précédente si celui-ci est encore sur l'automate... **Vous devez donc toujours charger un code bidon (comme « demo1 sep2003.stx ») lorsque vous quittez le laboratoire afin de ne pas donner votre code aux autres équipes.**