Automatismes industriels

Programmation des automates

Cours 2

2h - v1.0

IUT de Cachan, 9 Avenue de la Division Leclerc, 94234 Cachan

Développement d'un projet en automatisme

Introduction

Le logiciel de développement d'un API (Automate Programmable Industriel) est un logiciel permettant de configurer et de programmer l'API associé. Chaque constructeur d'API impose son propre logiciel de développement.

Schneider	Siemens	Wago	Tend
Unity Pro	Tia Portal	CodeSys	Niagara
EcoStructure		eCockpit	

Table 1: Logiciel de développement pour chaque constructeur.

1 Projet Unity Pro

La Figure 1 est une capture d'écran d'une fenêtre *Unity Pro*. Sur la gauche se trouve l'Arborescence du projet, décrite dans la section suivante et la grande partie à droite est la partie programmation.

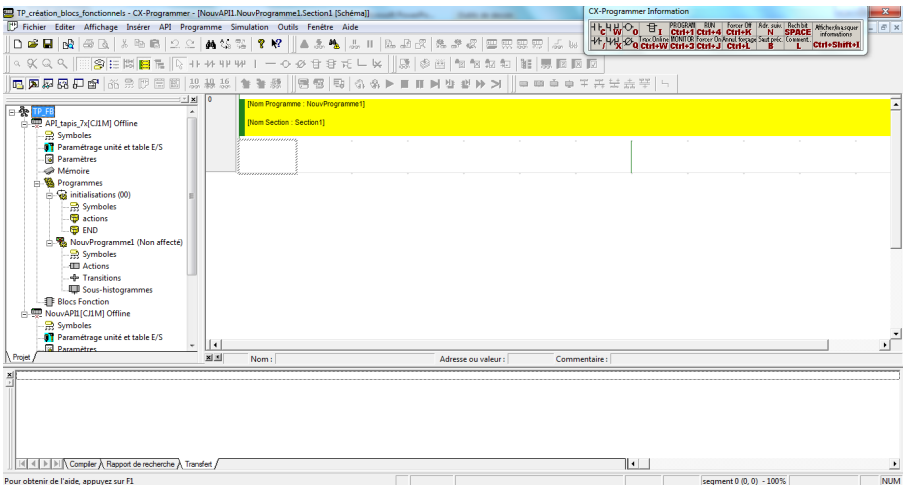
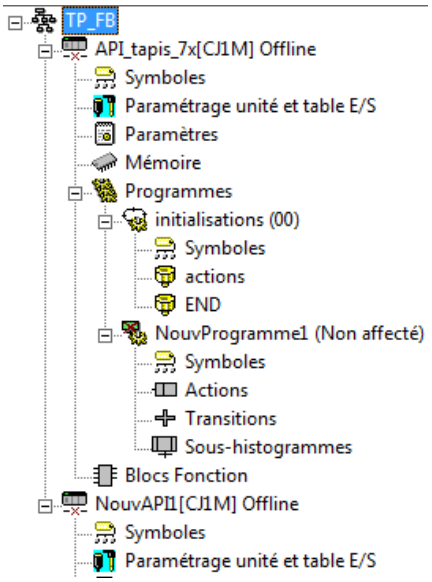


Figure 1: Fenêtre Unity Pro

1.1 L'Arborescence d'un projet Unity




```
Projet
├── API_Tapis[...]/ ..... Une première configuration
│   ├── Symboles ..... Variables globales
│   ├── Paramétrage unité et table E/S
│   ├── Paramètres ..... Paramètres de l'automate
│   ├── Programmes/ ..... Programmes dans l'API
│   │   ├── Programme_1/ ..... Exemple 1 : LADDER
│   │   │   ├── Symboles ..... Variables locales au programme
│   │   │   ├── actions ..... Le programme en tant que tel
│   │   │   └── END
│   │   └── Programme_2/ ..... Exemple 2 : SFC
│   │       ├── Symboles
│   │       ├── Actions
│   │       ├── Transitions
│   │       └── Sous-histogrammes
└── NouvAPI[...]/ ..... Une autre configuration
```

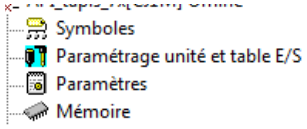
FIGURE 2: Arborescence d'un projet Unity

La Figure 2 présente l'arborescence générale d'un projet *Unity Pro*.

Un projet est constitué de différentes **configurations** (une configuration pour chaque automate relié au projet). Dans l'exemple de la Figure 2, les configurations sont API_tapis_7x[CJ1M] et NouvAPI1[CJ1M].

Dans chaque configuration, on trouve :

- Les ressources
  - ◇ On y trouve la configuration de l'automate, les variables globales (symboles), les déclarations des modules d'entrée-sortie, les registres (mémoire de l'automate) ...



- Les programmes (ou tâche)
  - ◇ tous les programmes d'un automate sont exécutés **simultanément**
  - ◇ chaque progamme peut avoir des variables locales (symboles)
  - ◇ ils peuvent être écrits en SFC, FBD, LD, ST ou IL.

1.2 Les variables

Les variables fournissent un moyen d'identifier des objets de données dont le contenu peut varier comme, par exemple, les données associées aux entrées/sorties.

À retenir

Les variables sont un moyen de stoquer une donnée (le nombre de pièces dans un bac par exemple) ou de connaître l'état d'un capteur (par exemple la présence d'une pièce devant un capteur ou encore la température)

Les variables (symboles) **globales** sont utilisables par toutes les tâches

- Les variables (symboles) **locales** à une tâche ne sont utilisables que par cette tâche.

Remarque

Dans *Unity Pro*, les variables sont appelées Symboles, on trouve les variables **globales** dans les ressources de chaque API et les variables locales à chaque programme dans sa propre arborescence.

Les variables sont donc stoquées dans la mémoire de l'automate. La mémoire d'un automate peut être vue comme une armoire gigantesques composée de différents "tiroirs" dans lesquelles on pourra stoquer une valeur. On représente souvent la mémoire par un tableau dans lequel on stoque une valeur par case. Une variable est donc stoquée dans une des cases du tableau. Le *numéro* de la case est appelé **adresse** de la variable.

À retenir *Adresse d'une variable*

L'**adresse** d'une variable correspond à l'endroit où cette variable est stoquée dans la mémoire de l'automate.

Dans un automate, en plus des variables internes servant à stoquer des infomations, il faut pouvoir accéder aux différentes **entrées/sorties** de l'automate. Ces entrées/sorties sont également associées à des adresses.

Ces adresses sont d'une forme bien précise : **%AttributTypek.i.j**. Par exemple : **%QX2.4.F**.

Attribut	Type	k.j.i
I : Entrée	**X** : Booléen	**k** : numéro de voie
Q : Sortie	**B** : 8 bits	**j** : numéro de carte
M : Mémoire	**W** : 16 bits	**i** : numéro de rack
K : Constante	**D** : 32 bits	
	F : Flottant	

TABLE 2: décomposition d'une adresses

1.2.1 Différents types de variables

En pratique, les variables que l'on stoque n'ont pas toutes le même type et ne nécessite pas la même taille de stockage.

Exemple:

On comprends aisément qu'un booléen (un seul bit qui ne peut valoir que **1** ou **0**) prend moins de place qu'un entier codé sur 8 bits pouvant prendre 2^8 valeurs.

Le Tableau 3 représente les noms et types de données utilisés par les automates.

Nom	**Type de données**	**Taille (bits)**	**Plage de valeurs**
Boolean	**BOOL**	1	0 (faux) ou 1 (vrai)
Short integer	**SINT**	8	$[-128; 127]$
Integer	**INT**	16	$[-32768; 32767]$
Double interger	**DINT**	32	$[-2147483648; 2147483647]$
Unsigned Short integer	**USINT**	8	$[0; 255]$
Unsigned Integer	**UINT**	16	$[0; 65535]$
Unsigned Double interger	**UDINT**	32	$[0; 4294967295]$
Real number	**REAL**	32	$[-2^{1}28; 2^{1}28]$
Bit string of length 8	**BYTE**	8	$[00; FF]$
Bit string of length 16	**WORD**	16	$[0000; FFFF]$
Bit string of length 32	**DWORD**	32	$[0000; FFFFFFFF]$
Time	**TIME**		0.0s à 21 474 836.47s

TABLE 3: Nom et tailles des différents types de données