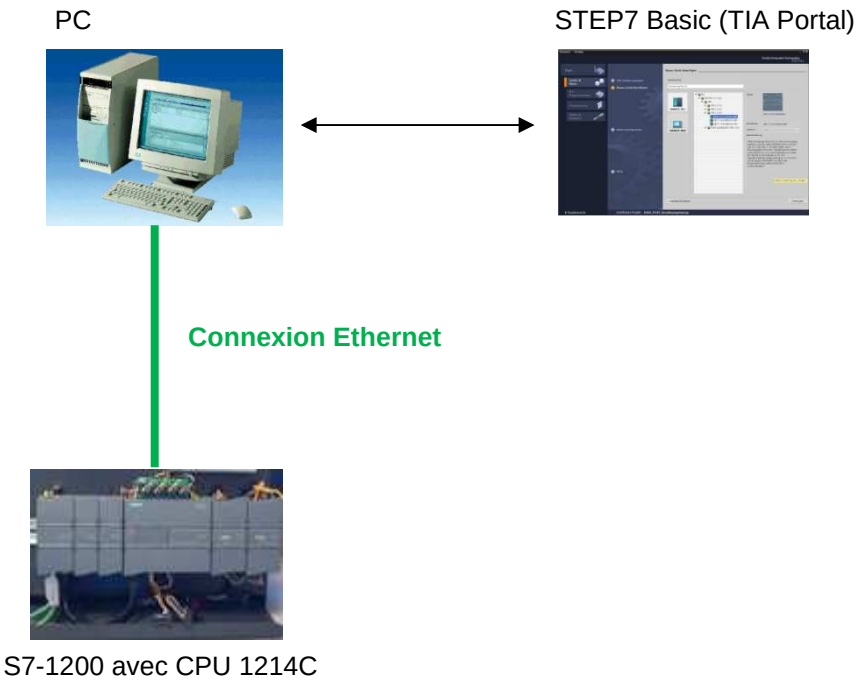
# 1 Objectif

Dans ce TP, vous allez apprendre comment programmer un automate programmable industriel (API) SIMATIC S7-1200, grâce au logiciel de programmation TIA Portal. Ce TP fournit les notions de base et montre les différentes étapes à suivre pour programmer l'API, en utilisant un exemple détaillé.

**Configurations matérielles et logicielles requises :**

1. Logiciel STEP7 Basic VX.X (Totally Integrated Automation (TIA) Portal VX.X)
2. Connexion Ethernet entre le PC et la CPU 1214C
3. API SIMATIC S7-1200 ; par exemple, la CPU 1214C. Les entrées doivent être mises en évidence sur un pupitre.




# 2 Notes concernant la programmation du SIMATIC S7-1200

## 2.1 Automate SIMATIC S7-1200

L'automate SIMATIC S7-1200 est un mini-contrôleur modulaire utilisé pour les petites performances.
Il existe un éventail complet de modules pour une adaptation optimisée à la tâche d'automatisation. Le contrôleur S7 est composé d'une CPU qui est équipée d'entrées et de sorties de signaux numériques et analogiques.
Des modules additionnels d'entrées/sorties (modules IO) peuvent être installés si les entrées et sorties intégrées ne sont pas suffisantes pour l'application désirée.
Si besoin est, des modules de communication RS232 ou RS485 sont ajoutés.
Une interface TCP/IP intégrée est obligatoire pour toutes les CPU.

Avec le programme S7, l'API surveille et contrôle une machine ou un process.
Les modules IO sont interrogés dans le programme S7 au moyen d'adresses d'entrées (%I) et référencés au moyen d'adresses de sorties (%Q).

Le système est programmé avec le logiciel STEP 7 Basic VX.X.

## 2.2 Logiciel de programmation STEP 7 Basic VX.X (TIA Portal VX.X)

Le logiciel STEP 7 Basic VX.X est l'outil de programmation pour le système d'automatisation :

- SIMATIC S7-1200.

Avec STEP 7 Basic VX.X, les fonctions suivantes peuvent être utilisées pour automatiser un système:

- Configuration et paramétrage du matériel
- Paramétrage de la communication
- Programmation
- Test, mise en service et dépannage avec les fonctions Démarrer et En ligne & Diagnostic.
- Documentation
- Génération d'écrans de visualisation pour les Basic Panels SIMATIC

Toutes les fonctions sont détaillées dans l'aide en ligne.

# 3 Connexion à la CPU via le protocole TCP/IP, et retour aux paramètres d'usine

Pour programmer le SIMATIC S7-1200 à partir d'un PC ou d'un ordinateur portable, vous avez besoin d'une connexion TCP/IP.

Pour que le PC et le SIMATIC S7-1200 communiquent entre eux, il est aussi important que leurs adresses IP correspondent.

Dans un premier temps, voici comment paramétrer l'adresse IP de l'ordinateur.

- Dans le menu « **Démarrer** », allez dans « **Paramètres** > **Connexions réseau** », puis clic-droit « **Propriétés** » de la connexion au réseau local.
- Sélectionnez « **Protocole Internet TCP/IP** » dans la liste puis cliquez sur « **Propriétés** ».
- Vous pouvez maintenant configurer l'**adresse IP** et le **masque de sous-réseau**
  Rentrez les paramètres suivants :
  _ **Adresse IP :** ***demandez à votre enseignant***
  _ **Masque de sous-réseau :** ***demandez à votre enseignant***

# Rappels

**Adresse MAC :**

L'adresse MAC est composée de 6 nombres variant de 0 à 255 (48 bits), représentée pour plus de simplicité sous la forme hexadécimale XX.XX.XX.XX.XX.XX où X varie de 0 à F, avec une partie fixe et une partie variable. La partie fixe (les 3 premiers groupes XX.XX.XX) indique le fabricant de l'équipement réseau (SIEMENS, 3COM…). La partie variable, quant à elle, différencie les différentes stations Ethernet et doit être assignée de façon unique dans le monde. Sur chaque module, une adresse MAC est imprimée spécifiquement par l'usine.

**Plage de valeurs pour l'adresse IP :**

L'adresse IP est composée de 4 nombres décimaux variant de 0 à 255, séparés par un point.
Par exemple, « 192.168.0.1 ».

**Plage de valeurs pour le masque de sous-réseau :**

Ce masque est utilisé pour reconnaître si une station ou une adresse IP appartient au sous-réseau local, ou si elle ne peut être joignable qu'avec un routeur.
Le masque de sous-réseau est composé de 4 nombres décimaux variant aussi de 0 à 255, séparés par un point. Par exemple, « 255.255.255.0 ».
**Dans leur représentation binaire, les 4 nombres décimaux du masque doivent contenir une suite ininterrompue de « 1 » à gauche, et le reste à droite étant des « 0 ».**
Les « 1 » indiquent la partie de l'adresse IP pour l'adresse du sous-réseau, et les « 0 » la partie de l'IP pour le numéro d'hôte.
*Exemple :*

| | |
|---|---|
| Masques corrects : | 255.255.255.0 base 10 ↔ 1111.1111.1111.1111.1111.1111.0000.0000 base 2 |
| | 255.255.128.0 base 10 ↔ 1111.1111.1111.1111.1000.0000.0000.0000 base 2 |
| | 255.254.0.0 base 10 ↔ 1111.1111.1111.1110.0000.0000.0000.0000 base 2 |
| Masque incorrect : | 255.255.1.0 base 10 ↔ 1111.1111.1111.1111.0000.0001.0000.0000 base 2 |

**Plage de valeurs pour l'adresse de la passerelle (routeur) :**

L'adresse comporte là encore 4 nombres décimaux variant de 0 à 255, séparés par un point.
Par exemple, « 141.80.0.1 ».

**Relation entre les adresses IP, routeurs et masques de sous-réseaux :**

**Les adresses IP et de passerelle ne doivent être différentes qu'aux endroits où il y a un « 0 » dans le masque de sous-réseau.**

*Exemple :*

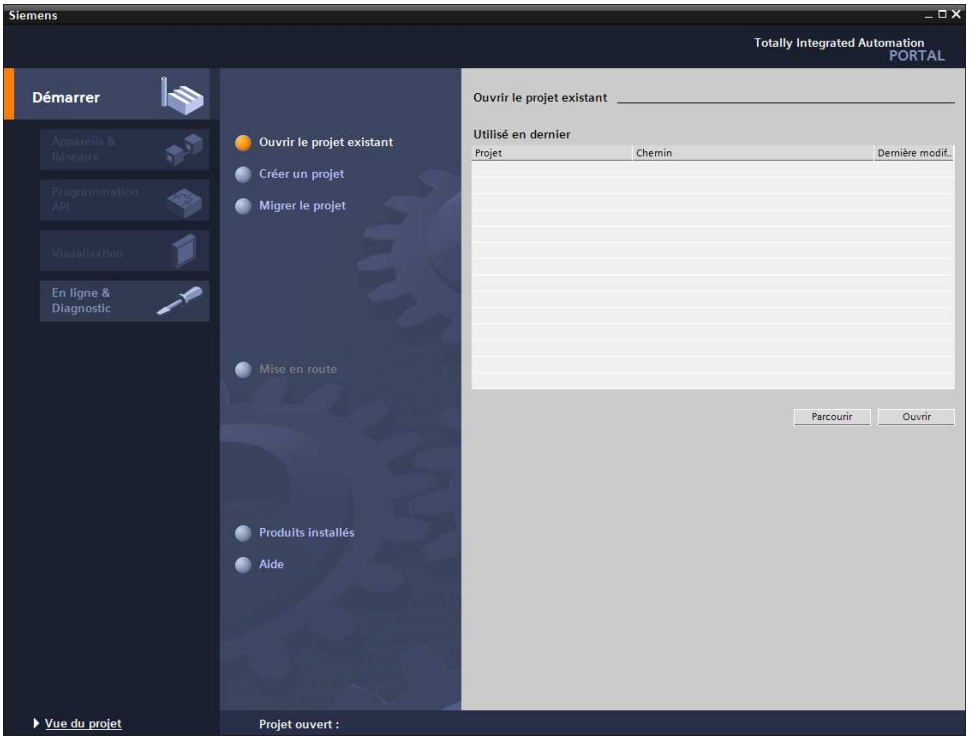
Vous entrez les paramètres suivants : masque de sous-réseau 255.255.255.0, adresse IP 141.30.0.5, et adresse du routeur 141.30.128.1.
L'adresse IP et l'adresse du routeur doivent ici avoir des valeurs différentes seulement pour le 4ème nombre décimal. Cependant, dans l'exemple, le nombre en 3ème position diffère déjà.
Cela signifie qu'on peut changer, par exemple :

- soit le masque de sous-réseau par 255.255.0.0,
- soit l'adresse IP par 141.30.128.5,
- soit la passerelle par 141.30.0.1.

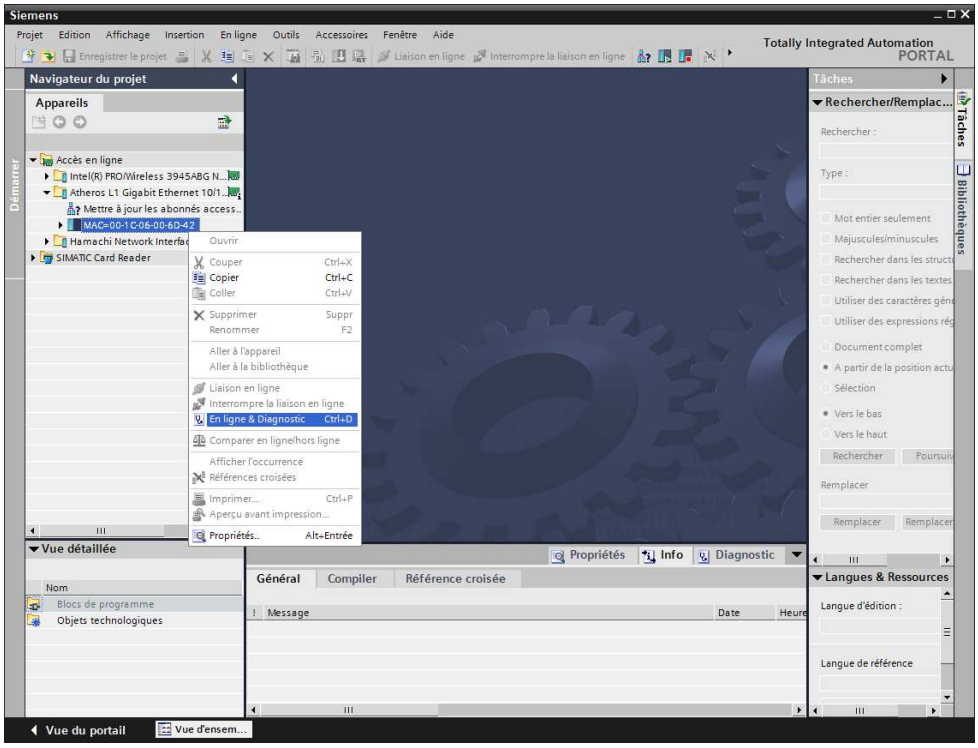
L'adresse IP du SIMATIC S7-1200 est paramétrée comme suit :

- Double-cliquez sur l'icône « **Totally Integrated Automation Portal VX** » pour lancer le logiciel Step 7 Basic VX.X.

- Puis, sélectionnez « **Vue du projet** ».



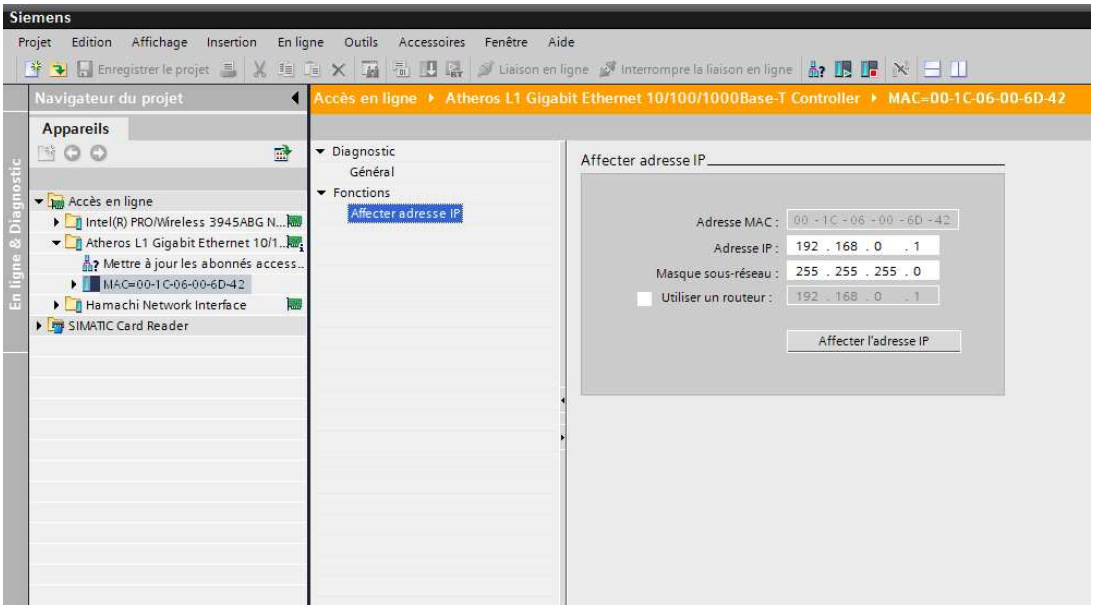

- Ensuite, dans le navigateur du projet, sélectionner dans l'arborescence de « **Accès en ligne** » la carte réseau paramétrée précédemment. Si vous cliquez sur « **Mettre à jour les abonnés accessibles** », vous verrez l'adresse MAC du SIMATIC S7-1200 connecté. Faites alors un clic-droit sur cette adresse et cliquez sur « **En ligne et Diagnostic** ».



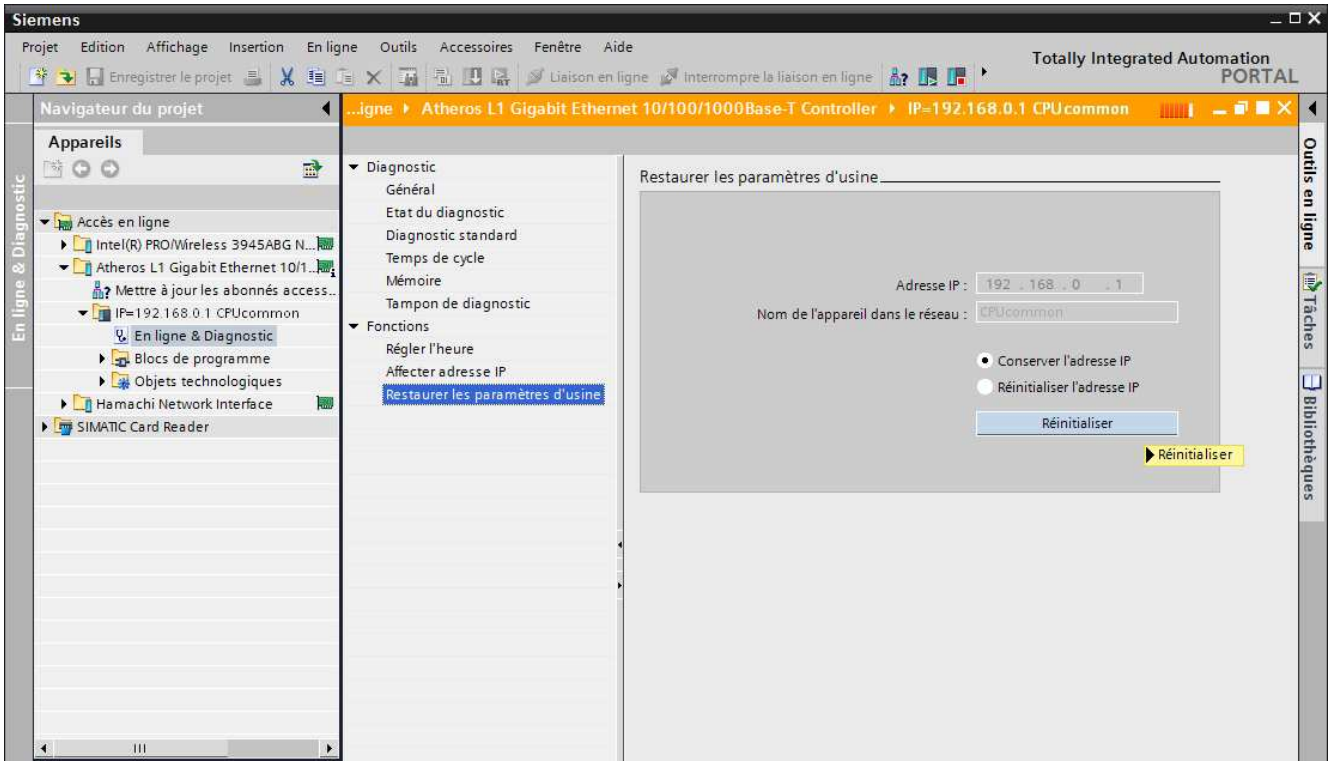
## Indication

Si l'adresse IP a déjà été paramétrée sur la CPU, vous verrez cette adresse au lieu de l'adresse MAC.

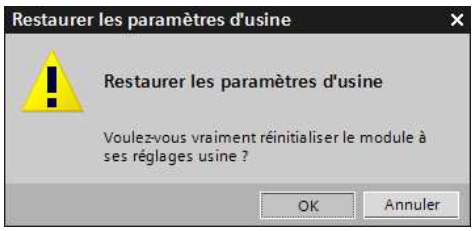
- Dans « **Fonctions** », vous verrez l'option « **Affecter adresse IP** ». Ici, entrez l'**adresse IP** « ***demandez à votre enseignant*** » et le **masque sous-réseau** « ***demandez à votre enseignant*** ». Ensuite, cliquez sur le bouton « **Affecter l'adresse IP** » pour que le SIMATIC S7-1200 prenne cette nouvelle adresse.




- Toujours dans « **Fonctions** », cliquez sur « **Restaurer les paramètres d'usine** ». Choisissez l'option « **Conserver l'adresse IP** » et cliquez sur « **Réinitialiser** ».



- Confirmez votre choix en cliquant sur « **OK** ».

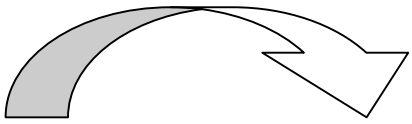



# 4 Rappels : qu'est-ce qu'un API, et à quoi ça sert ?

## 4.1 Que signifie le terme API ?

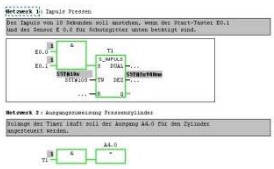
« API » est l'abréviation pour « **A**utomate **P**rogrammable **I**ndustriel ». Il s'agit d'un appareil qui commande un processus (par exemple une presse d'imprimerie pour l'impression des journaux, une installation de remplissage de sacs de ciment, une presse d'injection plastique, etc…). Ceci est réalisé grâce aux instructions d'un programme stocké dans la mémoire de l'appareil.

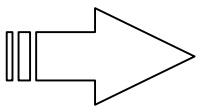
*Le programme est chargé dans la mémoire de l'API…*



*…et permet d'automatiser la machine.*

***Instructions*** ***Mémoire de l'API*** ***Machine automatisée***





## 4.2 Comment l'API commande-t-il le processus ?

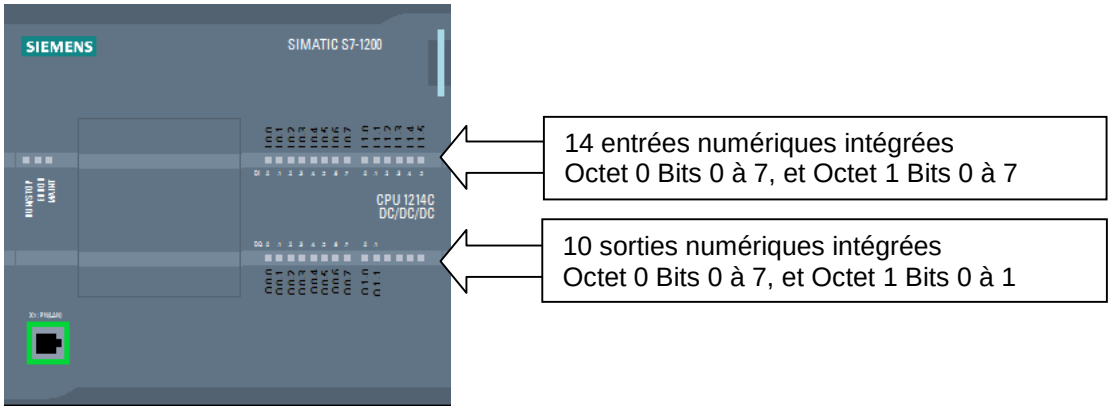
L'API commande le processus comme suit, à travers les connexions de l'API appelées **sorties**. Par exemple, il applique une tension de +24V aux **actionneurs** via les points de connexion de l'automate appelés **sorties**. Ceci permet de démarrer ou d'arrêter un moteur, de faire monter ou descendre des électrovannes, ou d'allumer ou éteindre des lampes.

## 4.3 Comment l'API reçoit-t-il les informations sur les états du processus ?

L'API reçoit les informations du processus à partir de ce qu'on appelle des **capteurs** qui sont reliés aux **entrées** de l'API. Ces capteurs de signaux peuvent être, par exemple, des capteurs qui reconnaissent si une pièce d'usinage se trouve à une position donnée, ou de simples commutateurs ou boutons poussoirs, qui peuvent être ouverts ou fermés, appuyés ou relâchés. Il est également fait la distinction entre les **contacts à ouverture** qui sont fermés au repos et les **contacts à fermeture** qui sont ouverts au repos.

## 4.4 Comment le SIMATIC S7-1200 adresse les signaux d'entrée/sortie ?

La déclaration d'une entrée ou sortie donnée à l'intérieur d'un programme s'appelle **l'adressage**.
Les entrées et sorties des automates sont la plupart du temps regroupées en groupes de huit entrées ou sorties numériques. Cette unité de huit entrées ou sorties est appelée un **octet**. Chaque groupe reçoit un numéro que l'on appelle l'**adresse d'octet**.
Afin de permettre l'adressage d'une entrée ou sortie à l'intérieur d'un octet, chaque octet est divisé en huit **bits**. Ces derniers sont numérotés de 0 à 7. On obtient ainsi l'**adresse du bit**.
L'automate programmable représenté ici a les octets d'entrée 0 et 1 ainsi que les octets de sortie 4 et 5.




Par exemple, pour adresser la 5ème entrée en partant de la gauche, on définit l'adresse suivante :

%E 0.4

%E indique ici que l'adresse est de type « entrée », 0 l'adresse d'octet, et 4 l'adresse de bit.
Les adresses d'octet et de bit sont toujours séparées par un point.

**Indication :** L'adresse du bit de la cinquième entrée est un 4 car la numérotation commence à zéro.

Pour adresser la dernière sortie, par exemple, on définit l'adresse suivante :

%A 1.1

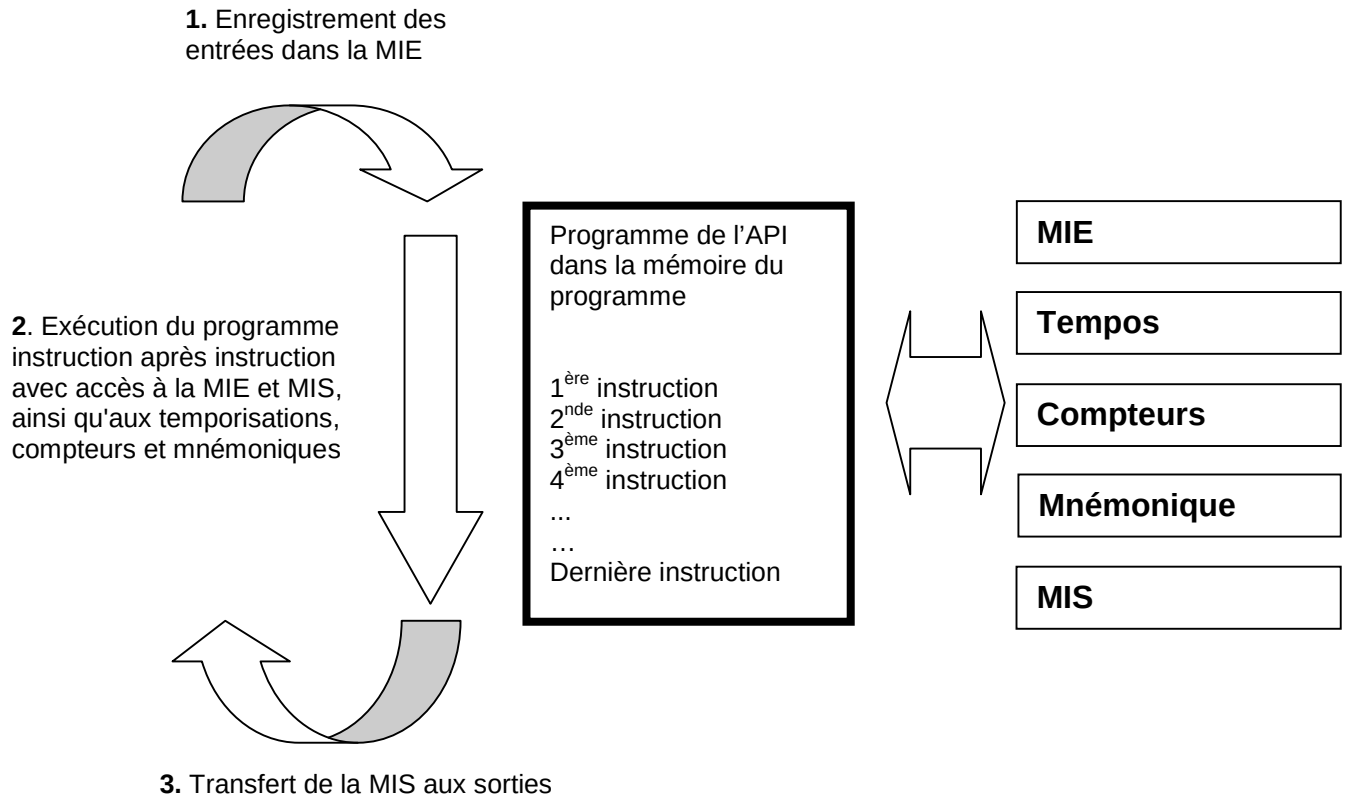
%A indique ici que l'adresse est de type « sortie », 1 l'adresse d'octet et 1 l'adresse de bit.
Les adresses d'octet et de bit sont toujours séparées par un point.

**Indication :** L'adresse du bit de la dernière sortie est un 1 car la numérotation commence à zéro.

## 4.5 Comment le programme est-il traité dans l'API ?

**Le traitement du programme dans l'automate est cyclique et se déroule comme suit :**

1. Après la mise sous tension de l'automate programmable, le processeur qui constitue pour ainsi dire le cerveau de l'automate vérifie si chaque entrée est sous tension ou non. L'état de ces entrées est enregistré dans la mémoire image des entrées (MIE). Si l'entrée est sous tension, l'information 1 ou "High" sera enregistrée. Si l'entrée n'est pas sous tension, l'information 0 ou "Low" sera enregistrée.
2. Ce processeur exécute le programme stocké en mémoire de programme. Celui-ci est constitué d'une liste d'instructions et d'opérations logiques exécutées de manière séquentielle. L'information d'entrée requise à cet effet est prélevée dans la mémoire image des entrées lue auparavant et les résultats logiques sont écrits dans une mémoire image des sorties (MIS). Durant l'exécution du programme le processeur accède également aux zones de mémoire des compteurs, temporisations et mnémoniques.
3. Dans la troisième étape, l'état est transmis après l'exécution du programme utilisateur de la MIS aux sorties, activant ou désactivant celles-ci. L'exécution du programme revient ensuite au point 1.




**Indication :** Le temps requis par le processeur pour l'exécution du programme s'appelle le temps de cycle. Ce dernier dépend entre autres du nombre et du type d'instructions.

## 4.6 À quoi ressemblent les opérations logiques dans le programme de l'automate ?

Les opérations logiques servent à définir des conditions pour l'activation d'une sortie.
Elles peuvent être créées dans le programme de l'automate programmable dans les langages de programmation « Schéma des circuits » (**CONT**) ou « Logigramme » (**LOG**).
Nous nous limiterons en vue de simplification ici au langage LOG.
Il existe de nombreuses opérations logiques pouvant être mises en œuvre dans des programmes d'automatisation.
L'opération **ET** et l'opération **OU** ou bien la **NEGATION** d'une entrée sont les opérations les plus fréquemment utilisées et seront expliquées ici à l'appui d'un exemple.

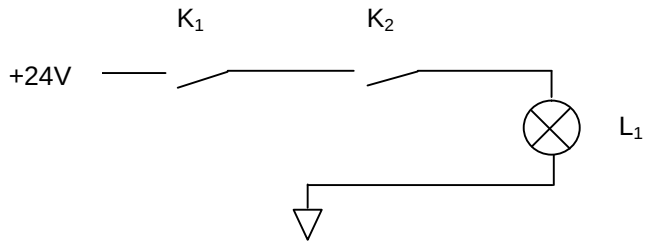
**Indications :** Pour obtenir rapidement et efficacement des informations sur les autres opérations logiques, consultez l'aide en ligne.

### 1. Opérateur *ET*

**Exemple d'une opération ET :**

Une lampe doit s'allumer quand les deux interrupteurs sont fermés simultanément.

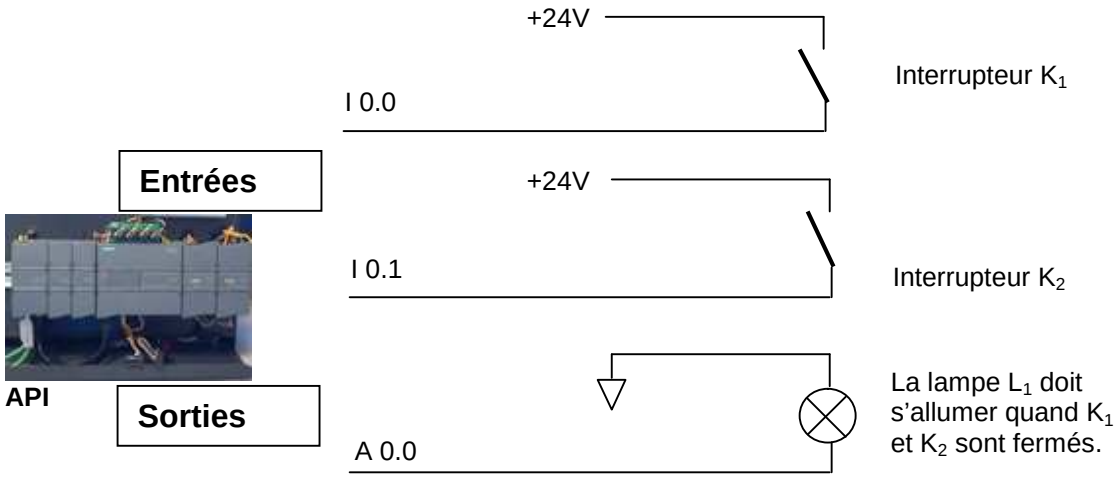
*Schéma :*




**Explication :**
La lampe s'allume uniquement quand les deux interrupteurs sont fermés.
C'est-à-dire, quand $K_1$ **ET** $K_2$ sont fermés, alors la lampe est allumée.

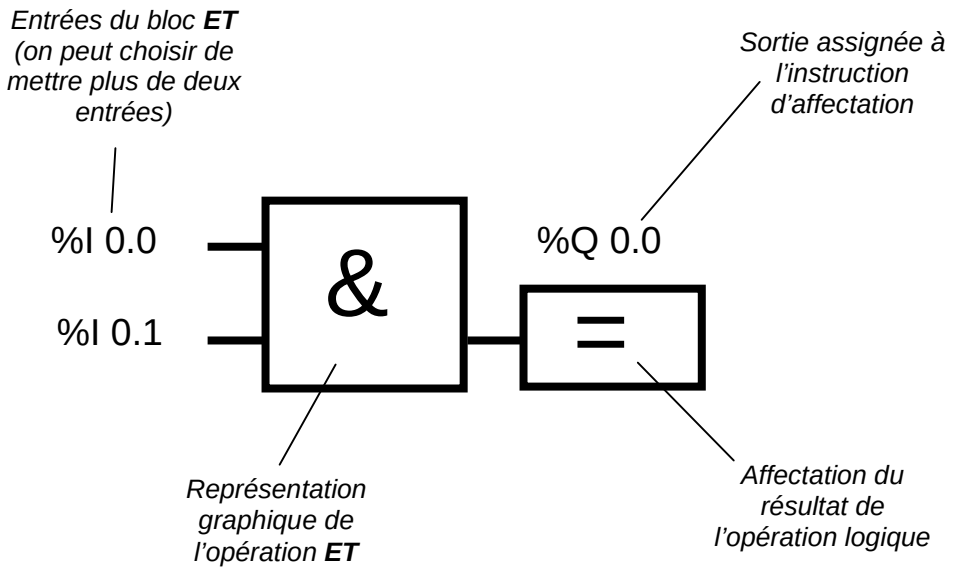
**Câbler l'API :**

Pour appliquer cette opération au programme de l'API, les deux commutateurs doivent être connectés aux entrées de l'API. Ici, $K_1$ est relié à l'entrée I 0.0 et $K_2$ à l'entrée I 0.1.
De plus, la lampe $L_1$ doit être connectée à une sortie, par exemple Q 0.0.




**Opérateur ET dans LOG :**

Dans le logigramme LOG, l'opérateur **ET** est programmé par le symbole ci-dessous et est représenté de la manière suivante :

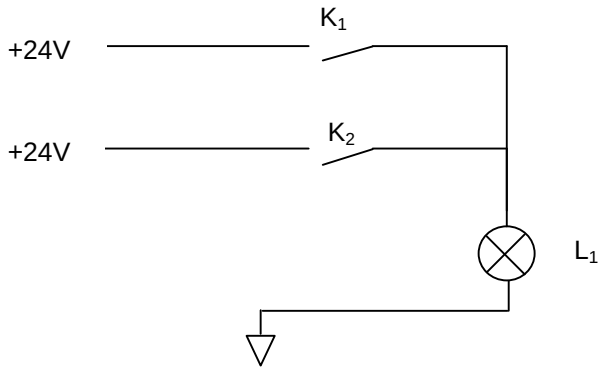


Opérateur *OU*

**Exemple d'une opération OU :**

Une lampe doit s'allumer si au moins un des deux interrupteurs est fermé.

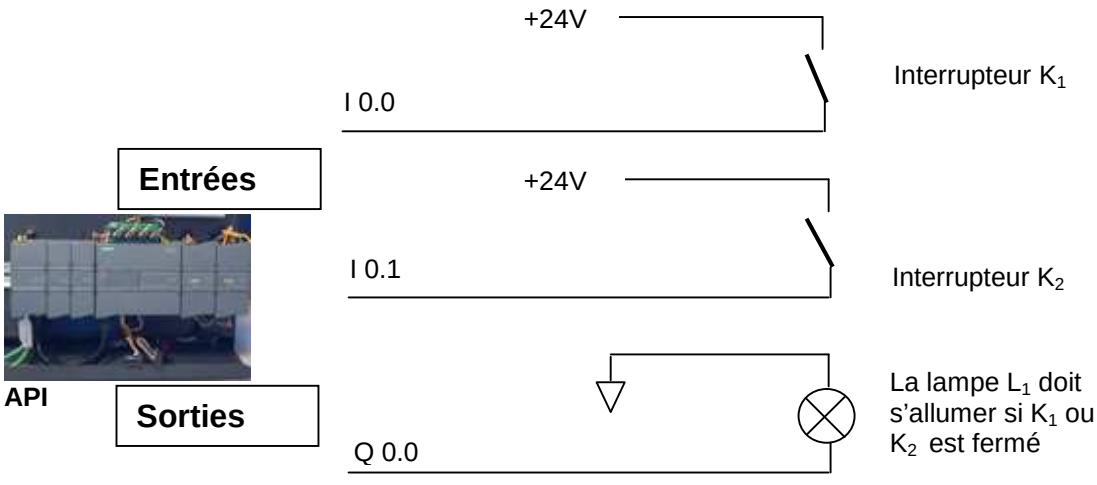
*Schéma*




**Explication :**
La lampe s'allume à partir du moment où un des deux interrupteurs est fermé.
C'est-à-dire, si $K_1$ **OU** $K_2$ est fermé, alors la lampe $L_1$ est allumée.

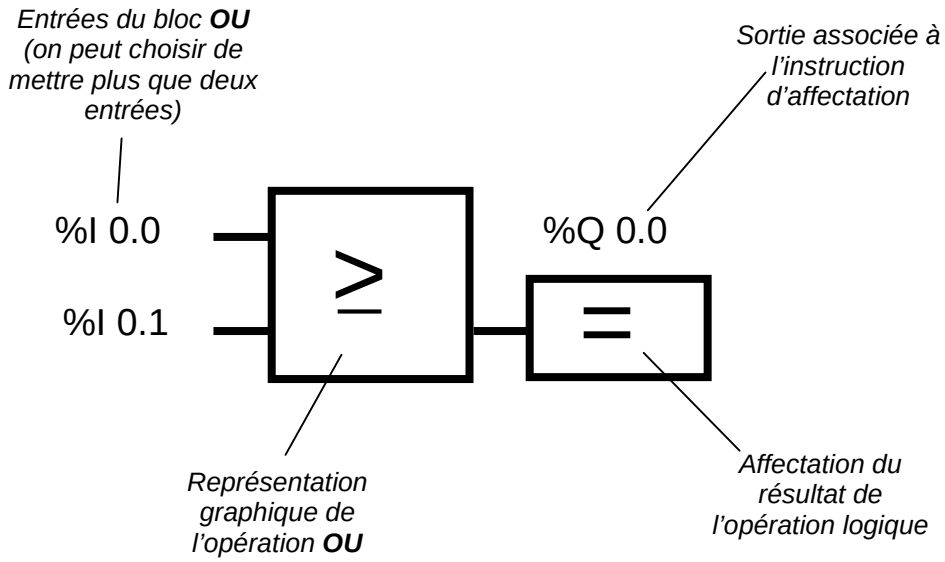
**Câbler l'API :**

Pour appliquer cette opération au programme de l'API, les deux commutateurs doivent être connectés aux entrées de l'API. Ici, $K_1$ est relié à l'entrée I 0.0 et $K_2$ à l'entrée I 0.1.
De plus, la lampe $L_1$ doit être connectée à une sortie, par exemple Q 0.0.



**Opérateur OU dans LOG**

Dans le logigramme LOG, l'opérateur OU est programmé est programmé par le symbole ci-dessous et est représenté de la manière suivante :



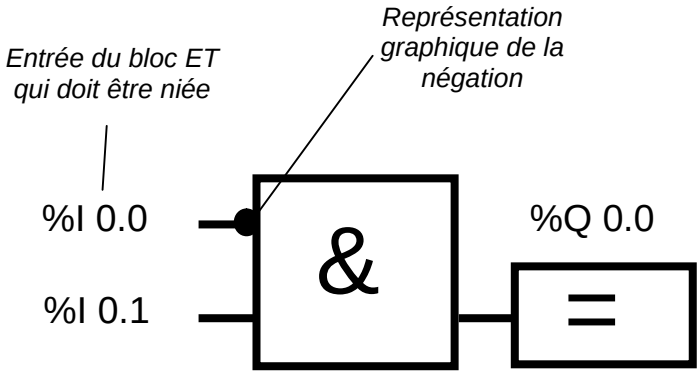

## 2. Négation

Il est souvent nécessaire dans les opérations logiques d'interroger l'état d'un contact pour savoir :

- dans le cas d'un **contact à fermeture** si celui-ci **n'a pas été activé**,
- ou dans le cas d'un **contact à ouverture** s'il **a été activé**,

et donc pour savoir si la tension est appliquée à la sortie ou non.
Ceci peut être réalisé par la programmation d'une **négation** à l'entrée de l'opération ET ou OU.

Dans le logigramme LOG, la négation de l'entrée (ou inversion) sur un opérateur ET est programmé de la façon suivante :

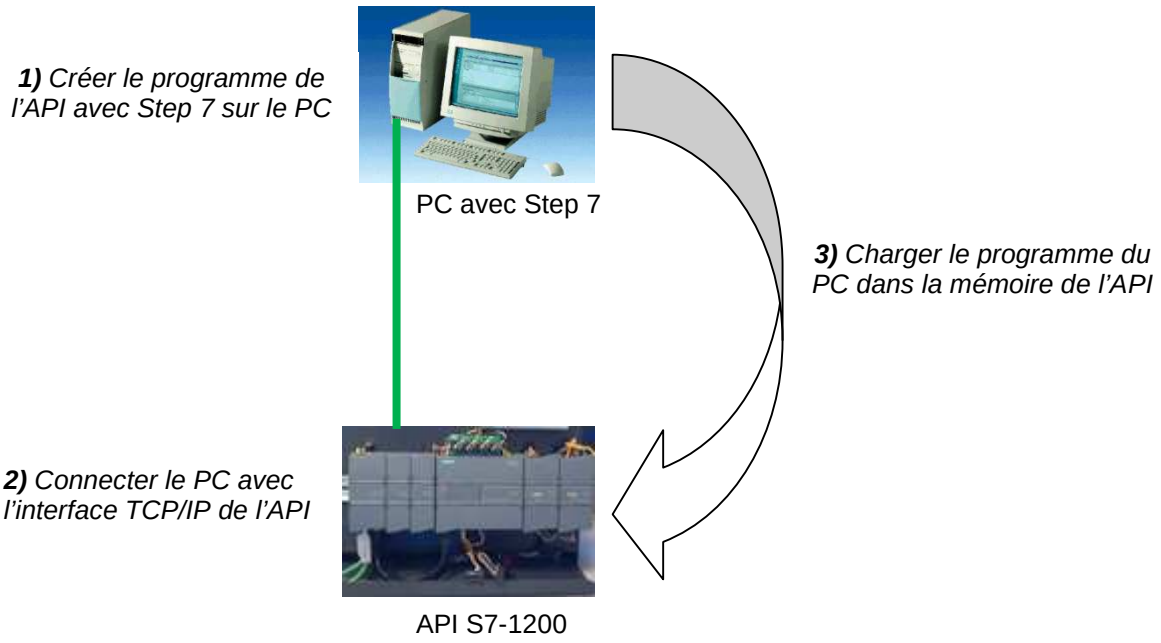



Ceci signifie qu'une tension est appliquée à la sortie %Q 0.0 uniquement si %I 0.0 est à « 0 » **et** si %I 0.1 est à « 1 ».

## 4.7 Comment est généré le programme pour l'API ? Comment est-il envoyé vers la mémoire de l'API ?

Le programme de l'API est généré sur un PC en utilisant le logiciel Step 7, et y est temporairement enregistré. Après la connexion du PC avec l'interface TCP/IP de l'API, le programme peut être transféré grâce à une fonction de chargement dans la mémoire de l'API.

Le PC n'est plus utile une fois que le programme est chargé dans l'API, ce dernier le gère tout seul.




**Indication :**

Cette procédure est décrite dans les chapitres ci-dessous.

# 5 Réglage et commande du SIMATIC S7-1200

**Présentation des différents modules :**

Le SIMATIC S7-1200 est un automate modulaire et tout un éventail de modules l'accompagnent. Les voici :

- Modules centraux CPU (*Central Processing Unit*) avec différentes capacités, entrées/sorties intégrées, et un interface PROFINET (par exemple, la CPU 1214C)

- Module de puissance PM (*Power Module*) avec une entrée AC 120/230V, 50Hz/60Hz, 1.2A/0.7A, et une sortie DC 24V/2.5A

- « Signal Boards » SB pour ajouter des entrées ou sorties analogiques ou numériques, la taille de la CPU étant fixée

(les « *signal boards* » peuvent être utilisés avec les CPU 1211C/1212C et 1214C)

- Modules de signal SM (*Signal Module*) pour les entrées et sorties analogiques et numériques

(pour les CPU 1212C un maximum de 2 SM peuvent être utilisés, pour la 1214C max. 8)

- Modules de communication CM (*Communication Module*) pour une communication série RS 232 / RS 485

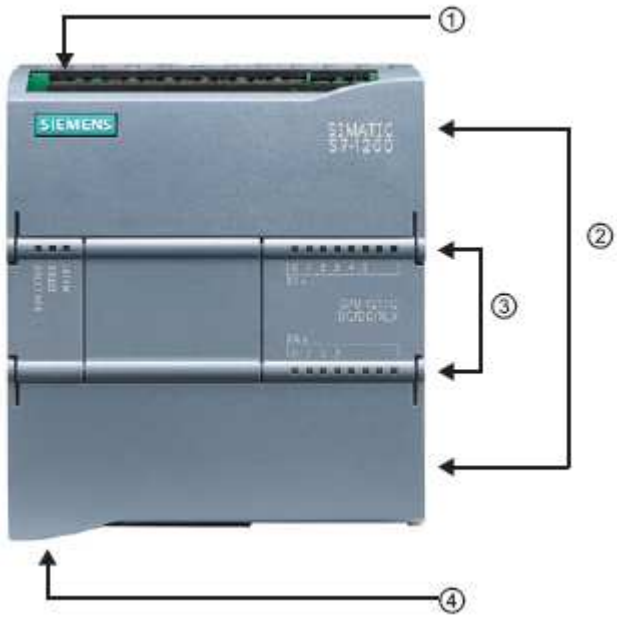
(pour les CPU 1211C/1212C et 1214C, jusqu'à 3 CM peuvent être utilisés)

- Les cartes mémoire 2Mo ou 24Mo pour stocker les données du programme et pour un remplacement simple des CPU lors des maintenances

## Eléments importants de la CPU

Avec une alimentation intégrée de 24V et des entrées et sorties numériques intégrées, la CPU du S7-1200 est prête à l'emploi, sans avoir besoin de composants supplémentaires.

Pour communiquer avec l'appareil de programmation, la CPU est équipée d'un port TCP/IP intégré.
Au moyen d'un réseau ETHERNET, la CPU est capable de communiquer avec des appareils de commande IHM et d'autres CPU.




(1) Alimentation 24V
(2) Borniers insérables pour un câblage utilisateur (derrière les caches plastiques)
(3) Diodes électroluminescentes pour les I/O intégrées et le mode de fonctionnement de la CPU
(4) Connexion TCP/IP (sous la CPU)

La carte mémoire SIMATIC MC (*Memory Card*) stocke le programme, les données, les données système, les fichiers et les projets. Elle peut être utilisée pour les opérations suivantes :

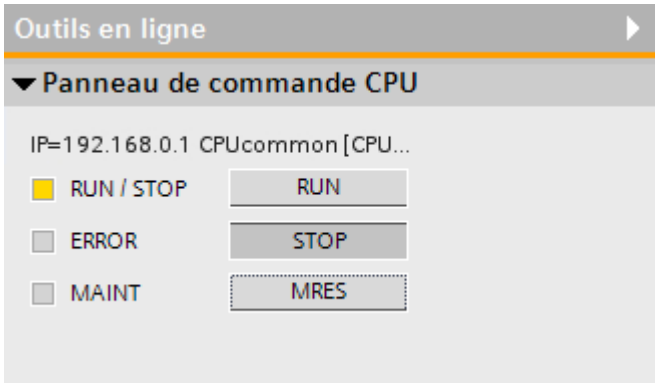
- Transfert du programme aux différentes CPU
- Mise à jour du micrologiciel des CPU, des modules de signal SM et des modules de communication CM

## Modes de fonctionnement de la CPU

La CPU a les modes de fonctionnement suivants :

- En mode « **STOP** », la CPU n'exécute pas le programme, et vous pouvez charger un projet.
- En mode « **STARTUP** », la CPU entame une procédure de démarrage.
- En mode « **RUN** », le programme est exécuté de façon cyclique. Les projets ne peuvent pas être chargés dans une CPU en mode RUN.

La CPU n'a pas de commutateur physique pour changer de mode de fonctionnement. Le mode **STOP** ou **RUN** se change en utilisant le bouton sur le panneau de commande du logiciel Step 7 Basic. De plus, le panneau de commande est muni d'un bouton **MRES** pour faire une réinitialisation générale de la mémoire et il affiche l'état actuel des LED de la CPU.




La couleur de la LED des états **RUN/STOP** sur la face avant de la CPU indique le mode de fonctionnement actuel.

- Une lumière **JAUNE** indique le mode **STOP**.
- Une lumière **VERTE** indique le mode **RUN**.
- Une lumière **CLIGNOTANTE** indique le mode **STARTUP**.

En outre il y a les LED « **ERROR** » et « **MAINT** » qui indiquent respectivement si une erreur est survenue et si une maintenance est requise.

# 6 Exemple d'application : Contrôle d'une presse

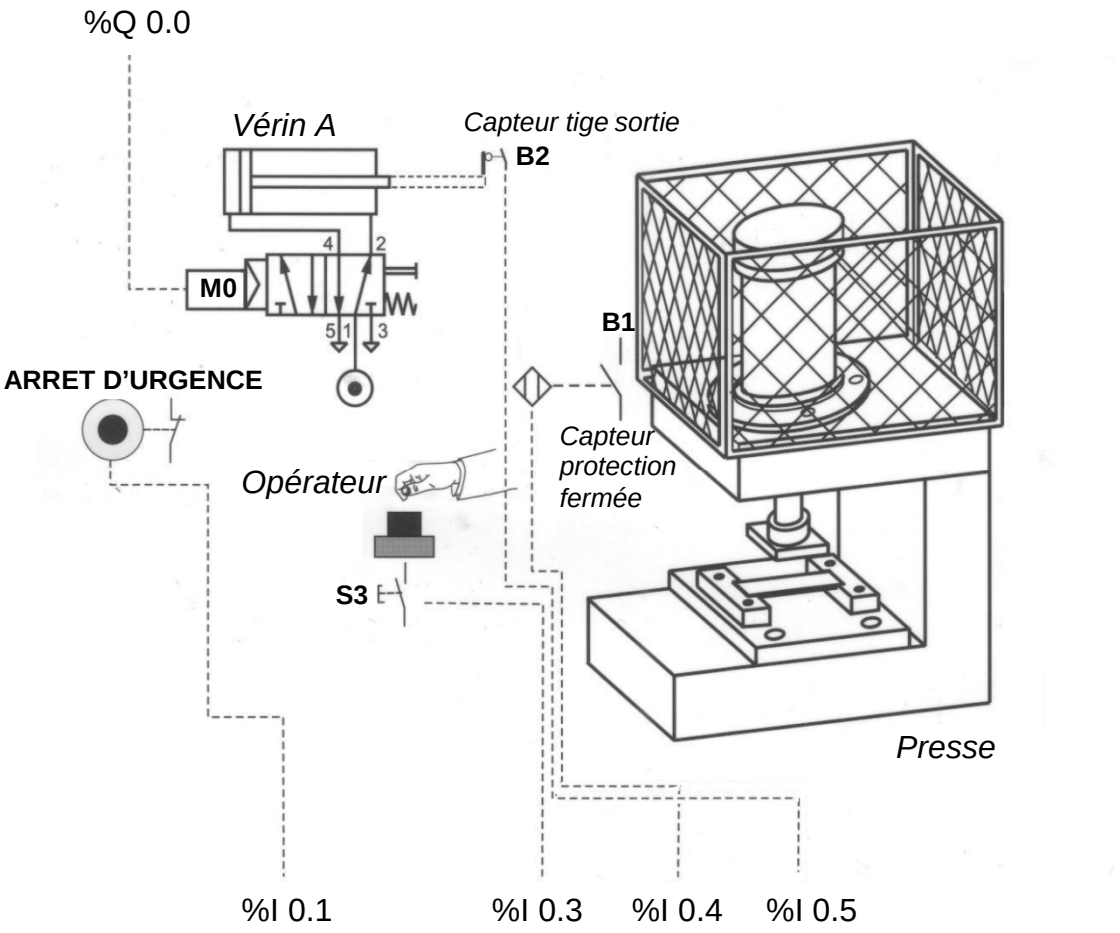
Notre première application va consister à programmer le contrôle d'une presse.

Une presse avec un capot de protection doit être activée avec un bouton **START S3** uniquement si l'écran de protection est fermé. Cette condition est surveillée avec un capteur **PROTECTION FERMEE B1**. Si c'est le cas, un distributeur 5/2 **M0** alimentant le vérin de la presse est activé, afin que la forme plastique puisse ensuite être pressée.

La presse doit se retirer de nouveau quand le bouton **ARRET D'URGENCE** (contact NF) est actionné, quand le capteur **PROTECTION FERMEE B1** ne répond plus, ou quand le capteur **VERIN TIGE SORTIE B2** répond.

**Tableau d'affectations**

| Adresses | Variables | Commentaires |
|---|---|---|
| %I 0.1 | ARRET D'URGENCE | Bouton d'arrêt d'urgence (contact NF) |
| %I 0.3 | S3 | Bouton de démarrage S3 (contact NO) |
| %I 0.4 | B1 | Capteur écran de protection fermé (contact NO) |
| %I 0.5 | B2 | Capteur vérin A tige sortie (contact NO) |
| %Q 0.0 | M0 | Sortir tige du vérin A |



# 7 Programmation de la presse pour le SIMATIC S7-1200

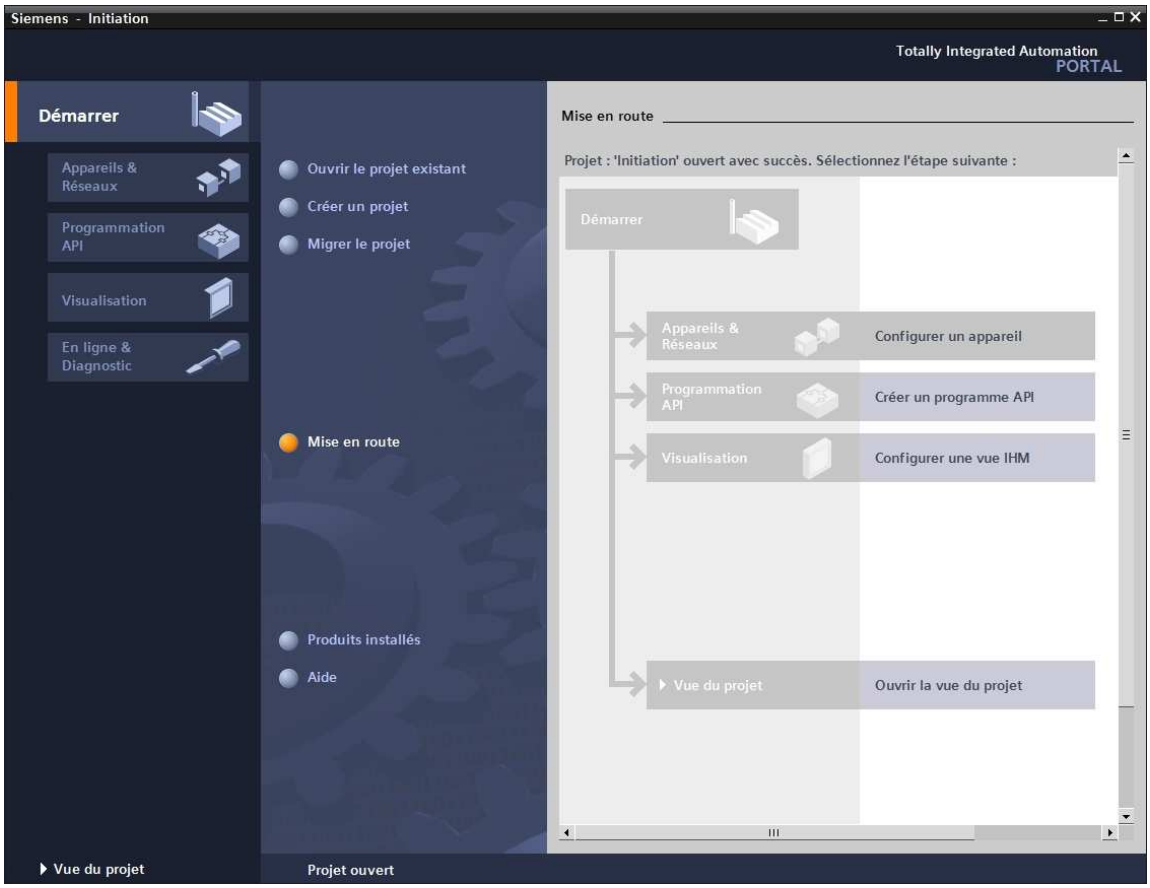
La gestion du projet et sa programmation se font grâce au logiciel « **Totally Integrated Automation Portal** ».

Là, sous une même interface, les éléments tels que le contrôleur, la visualisation et la mise en réseau de la solution d'automatisation sont créés, paramétrés et programmés.
Les outils en ligne sont disponibles pour les diagnostics d'erreur.

Le logiciel **TIA Portal** possède deux vues différentes : la vue du portail et la vue du projet.

## 7.1 Vue du portail

La vue du portail fournit une vue d'ensemble du projet et un accès aux outils qui permettent de l'élaborer. Vous pouvez trouver rapidement ce que vous souhaitez faire, et appeler l'outil qui servira à accomplir la tâche voulue. Si vous le souhaitez, un changement vers la vue du projet s'effectue automatiquement pour la tâche sélectionnée. Cette vue simplifie donc principalement la préparation et la mise en place du projet.

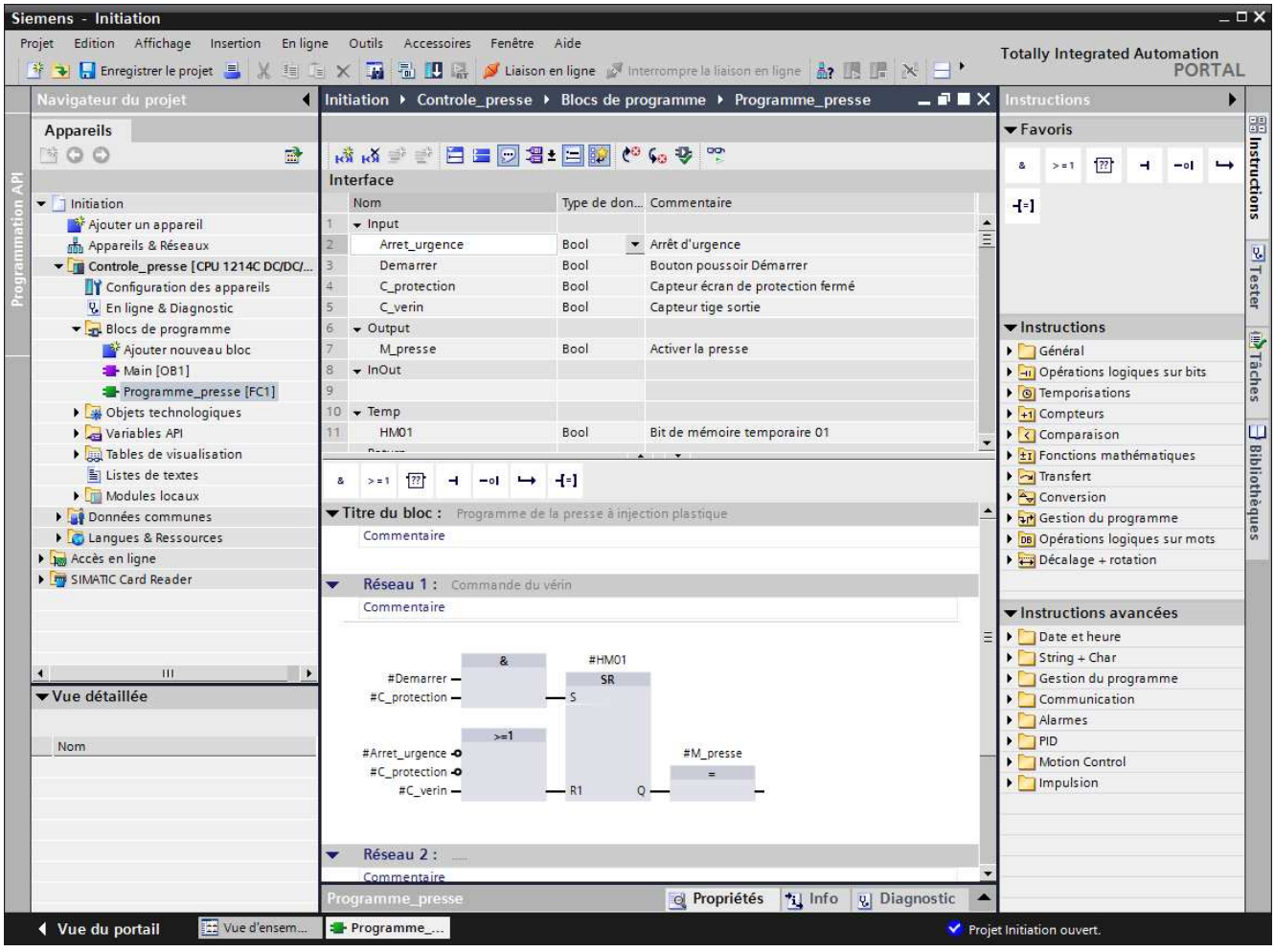



Indication : En bas à gauche de la fenêtre, on peut basculer de la vue du portail vers la vue du projet.

## 7.2 Vue du projet

La vue du projet est une vue structurée de tous les éléments constituant le projet. La barre de menu avec les barres de fonction est située comme le veut la norme en haut de la fenêtre, le navigateur du projet et tous les éléments du projet sont sur la gauche, et les menus associés aux différentes tâches (avec les instructions et les librairies, par exemple) sur la droite.

Si un élément (par exemple ici le bloc FC1) est sélectionné dans le navigateur du projet, il est affiché au centre et peut y être manipulé.

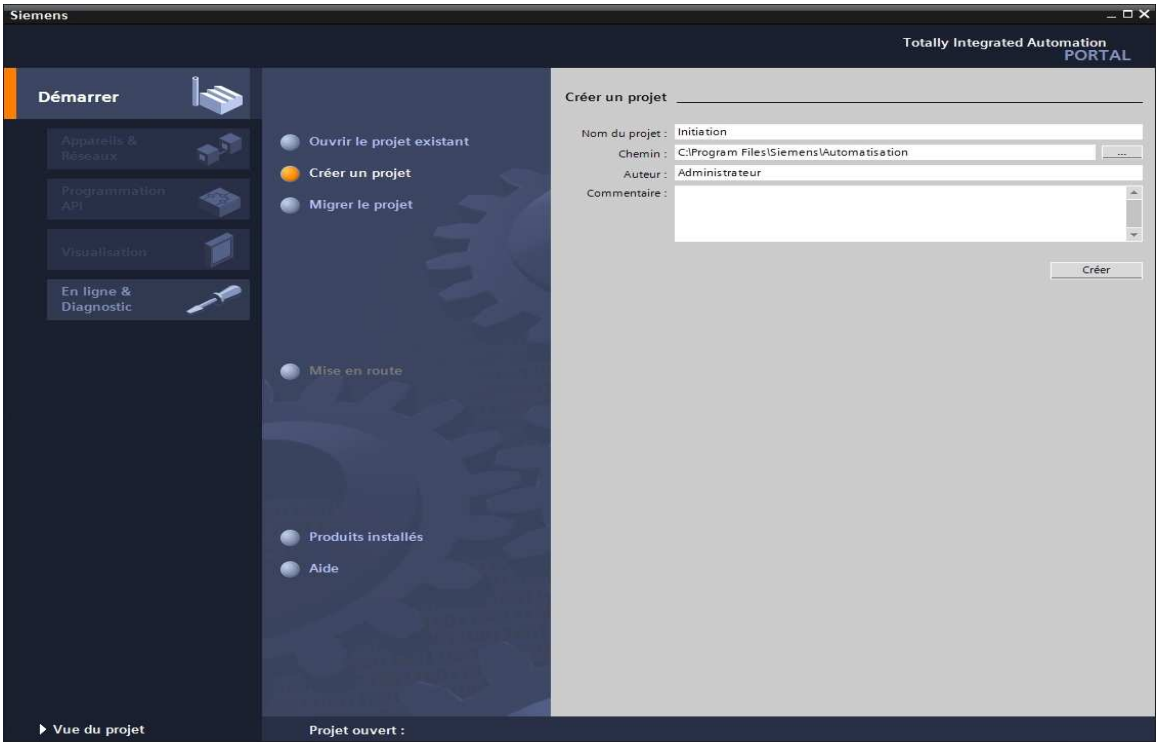



**Indication :** En bas à gauche, on peut retourner sur la vue portail !

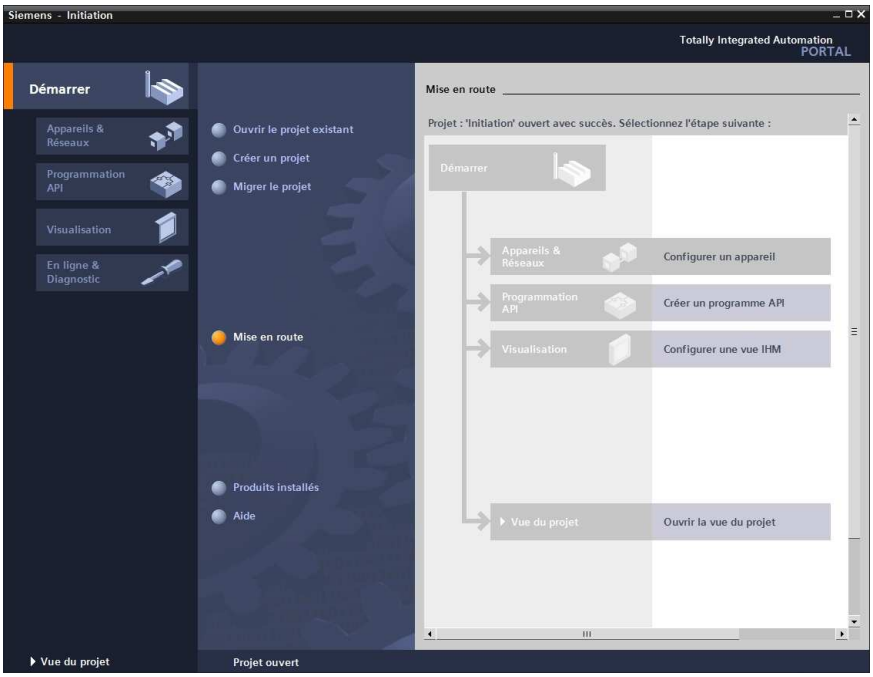
Les étapes ci-dessous montrent comment créer un projet pour SIMATIC S7-1200 et programmer la solution pour cette application.

1. L'outil que nous allons utiliser est « **Totally Integrated Automation Portal** », que l'on appelle ici d'un double-clic.

2. Les programmes pour SIMATIC S7-1200 sont gérés sous forme de projets. Nous allons maintenant créer un nouveau projet via la vue portail (« **Créer un projet** > **Nom** : *Initiation* > **Créer** »)

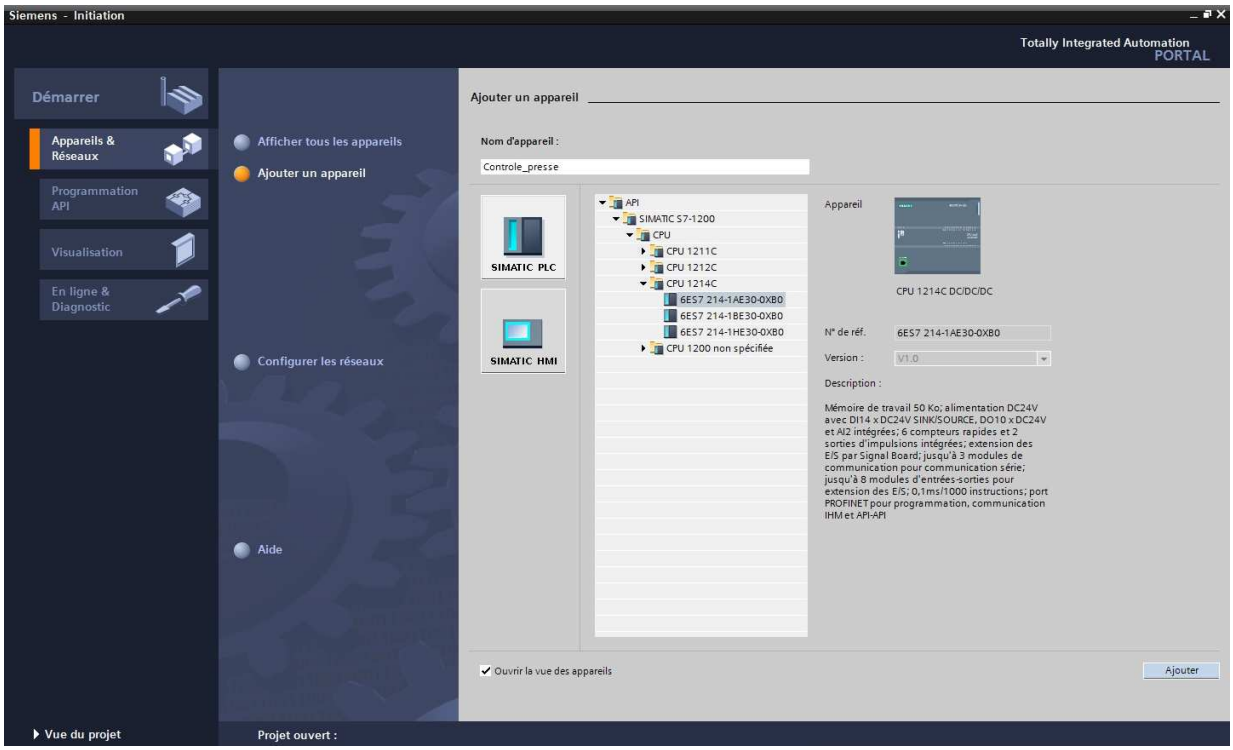



3. « **Mise en route** » est recommandée pour le début de la création du projet.
Premièrement, nous voulons « **Configurer un appareil** » (« **Mise en route** > **Configurer un appareil** »).

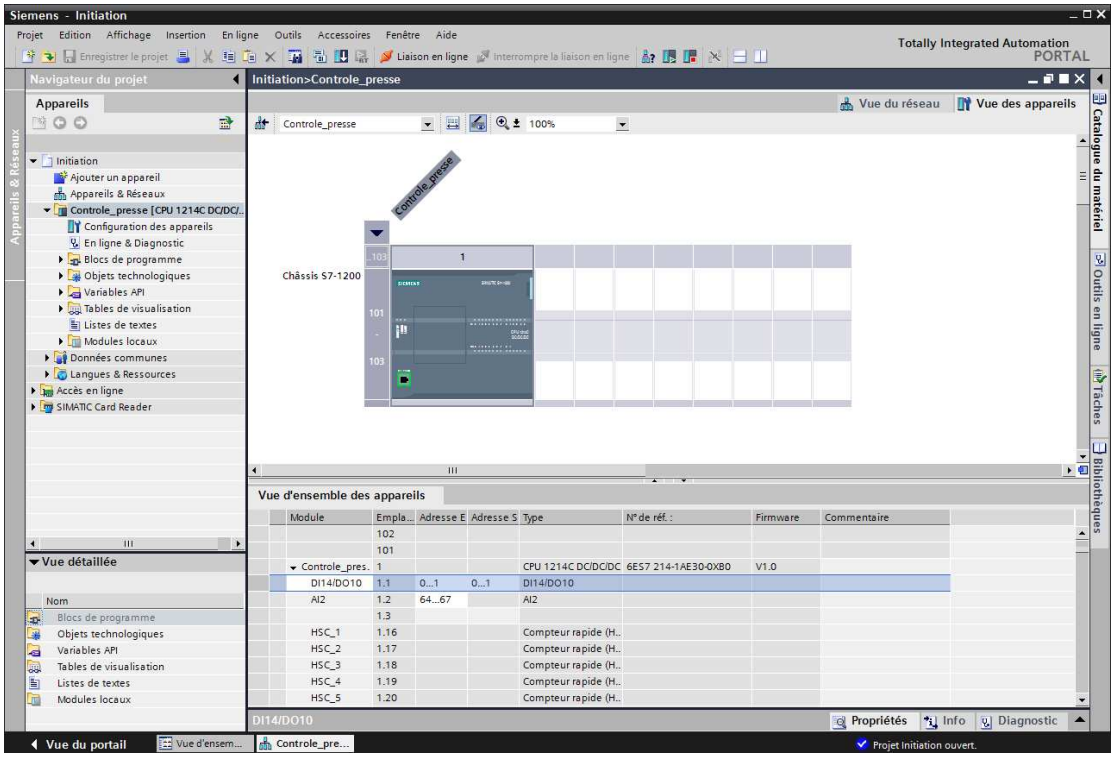


Puis « **Ajouter un appareil** » avec le nom d'appareil : *Controle_presse*. Choisissez alors dans le catalogue la « **CPU 1214C** » avec la bonne combinaison de lettres derrière.

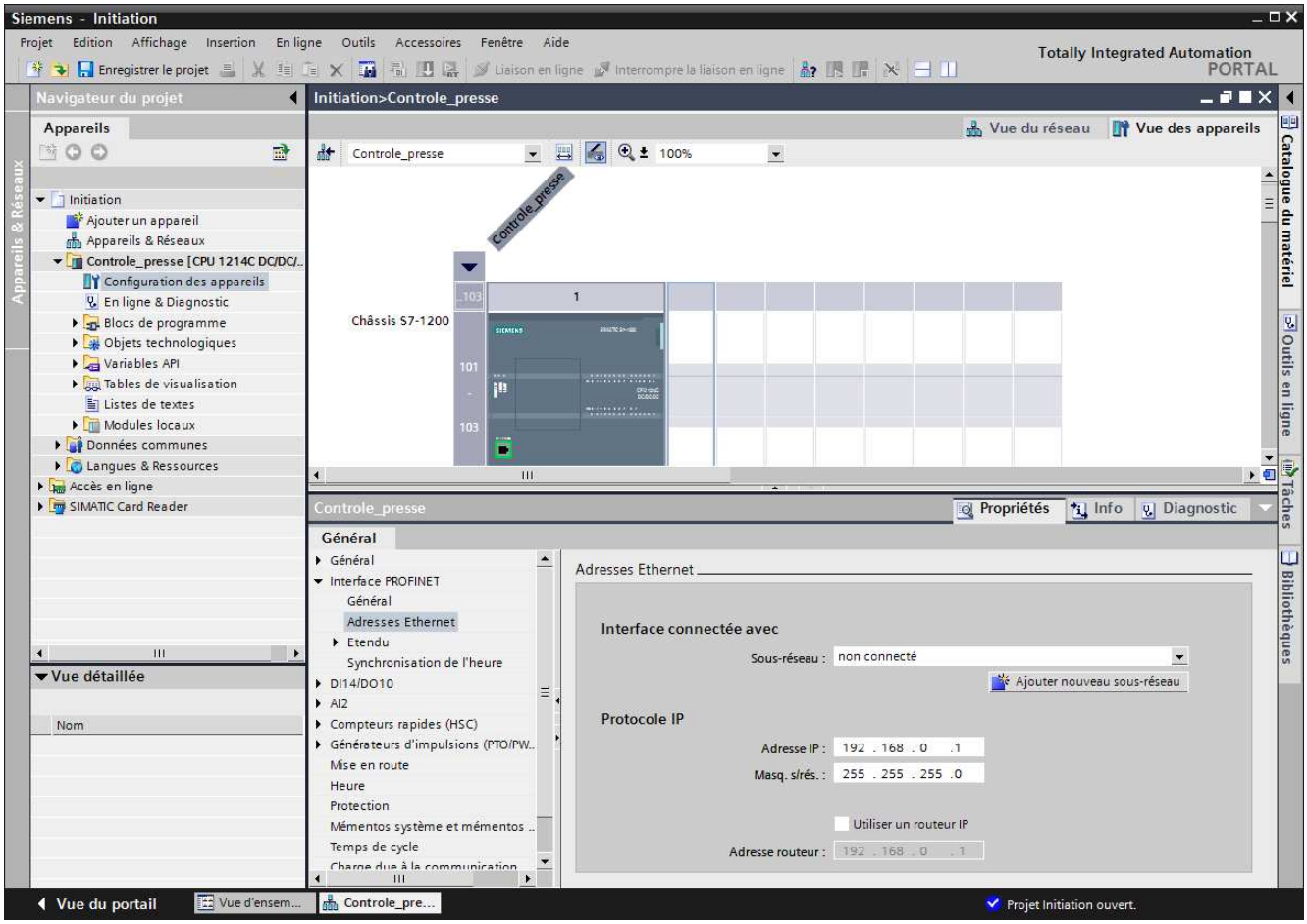
(« **Ajouter un appareil** > **SIMATIC PLC** > **CPU 1214C** > **6ES7 214-1AG40-0XB0**> **Ajouter** »)




4. Le logiciel bouge automatiquement vers la vue du projet avec la configuration matérielle ouverte. Ici, on peut ajouter des modules supplémentaires depuis le **Catalogue du matériel** (fenêtre de droite), et dans la **Vue d'ensemble des appareils**, les adresses d'entrée/sortie peuvent être visualisées. Dans notre cas, les entrées intégrées à la CPU ont des adresses allant de %I 0.0 à %I 1.5 (soit 14 entrées) et les sorties intégrées des adresses allant de %Q 0.0 à %Q 1.1 (soit 10 sorties).



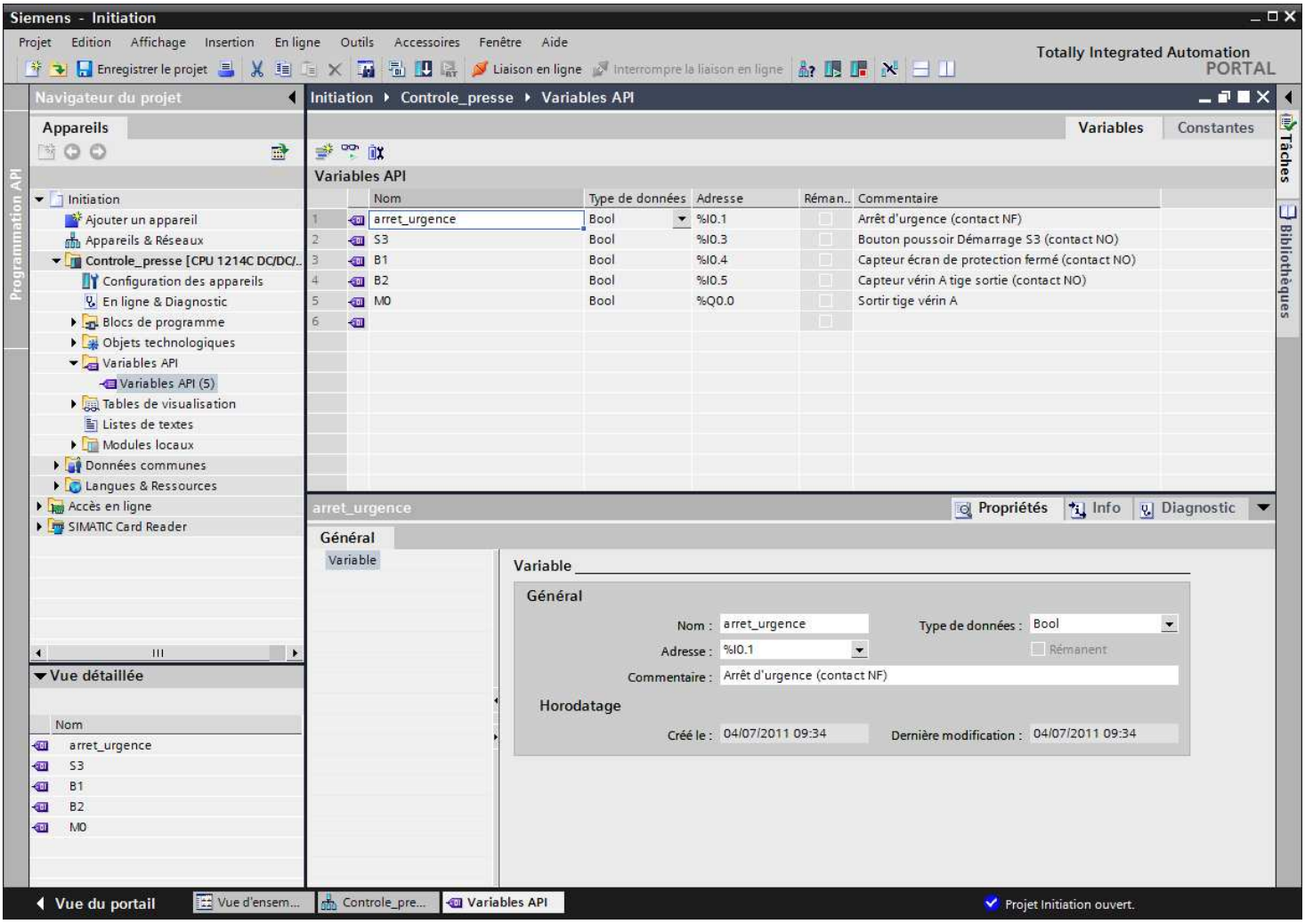
5. Afin que le logiciel puisse accéder dans la suite à la bonne CPU, son adresse IP et le masque de sous-réseau doivent être paramétrés (« **Propriétés** > **Général** > **Interface PROFINET** > **Adresses Ethernet** > **Adresse IP** : ***demandez à votre enseignant*** et **Masq. s/rés.** : ***demandez à votre enseignant***»).



6. Puisque de nos jours on programme avec des variables plutôt qu'avec des adresses absolues, on doit spécifier les **variables globales de l'API**.

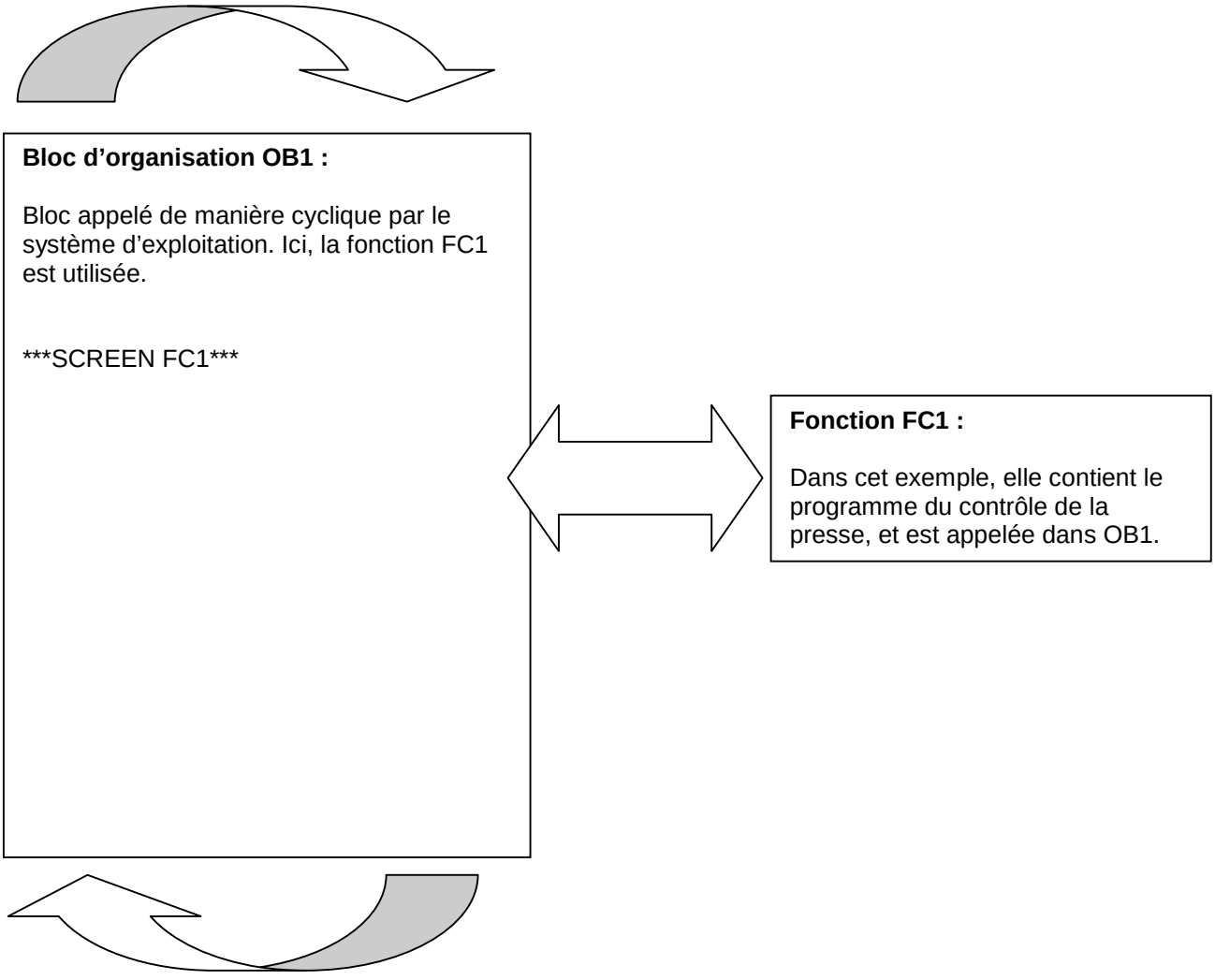
   Ces variables API globales sont des noms descriptifs et des commentaires pour ces entrées et sorties utilisées dans le programme. Plus tard, pendant la programmation, on pourra accéder à ces variables API via leurs noms.
   Ces variables globales peuvent être utilisées partout dans le programme, dans tous les blocs.

   À cette fin, sélectionnez dans le navigateur du projet « **Controle_citerne [CPU 1214C DC/DC/DC]** » puis « **Variables API** ». Avec un double-clic, ouvrez la table des variables API et entrez, comme montré ci-dessous, les noms des entrées et des sorties.

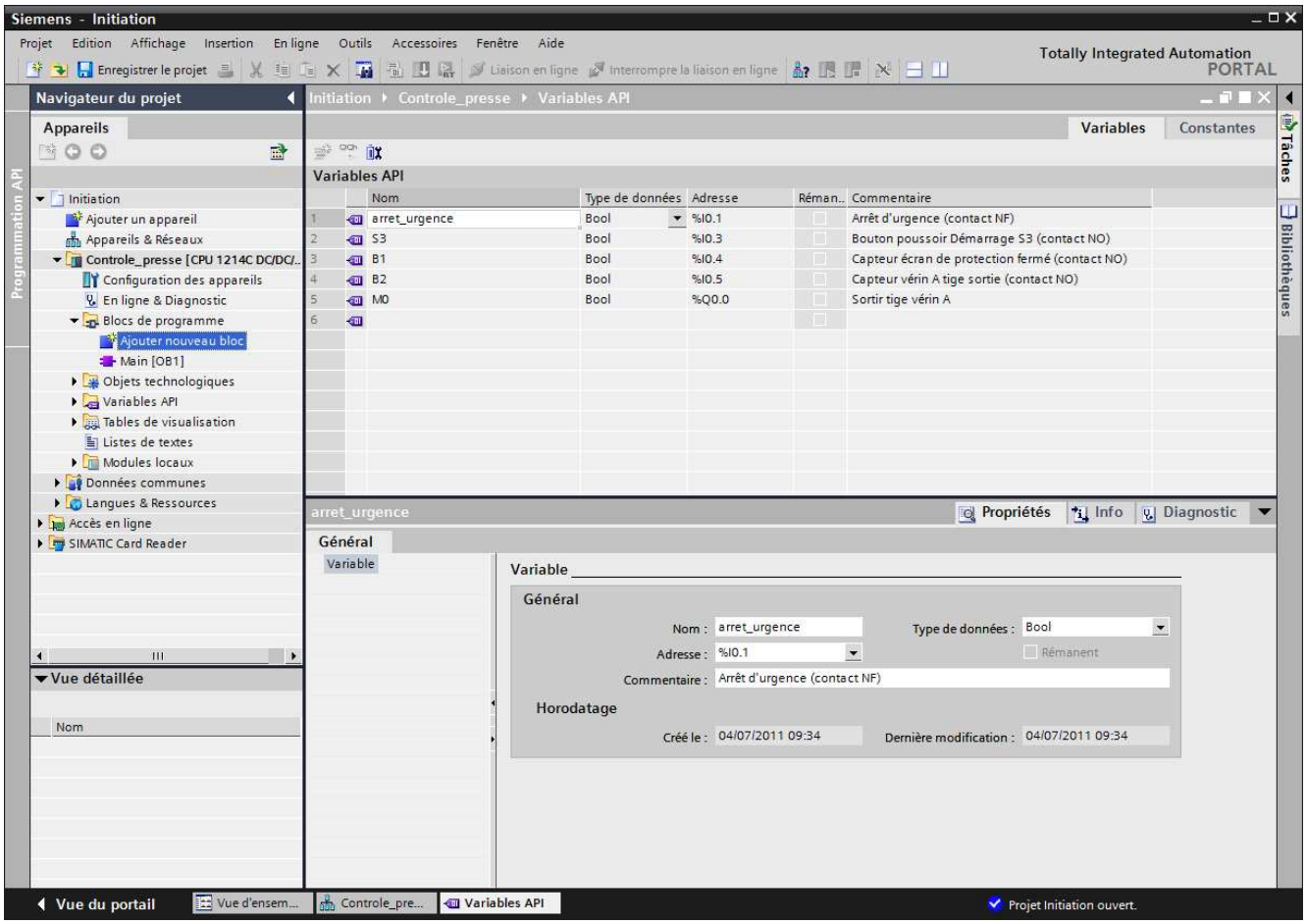


7. Les séquences du programme sont écrites dans ce qu'on appelle des blocs. De base, un bloc d'organisation OB1 est créé lors de l'ajout d'une CPU.
   Ce bloc représente l'interface du système d'exploitation de la CPU. Il est appelé automatiquement par celle-ci, et est traité de manière cyclique.
   À partir de ce bloc d'organisation OB1, des blocs supplémentaires peuvent être appelés à leur tour pour structurer le programme, comme la fonction FC1.
   Le but est de diviser une tâche globale en plusieurs sous-tâches, ce qui permet de programmer et de tester leur fonctionnalité plus facilement.

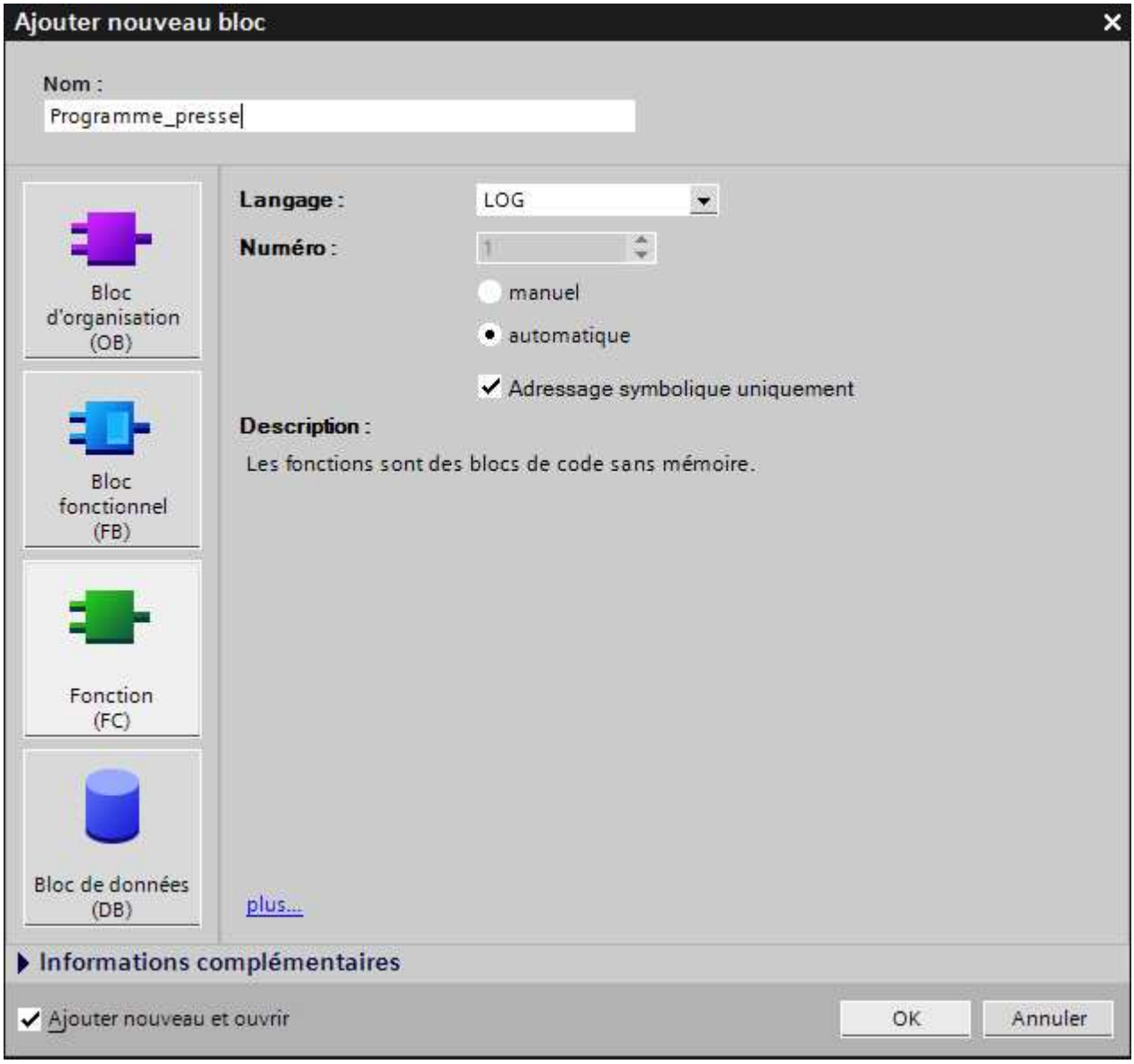
**Structure du programme de l'exemple :**



8. Pour créer le bloc fonction FC1, sélectionnez dans le navigateur du projet « **Controle_presse [CPU 1214 C DC/DC/DC]** » puis « **Blocs de programme** ». Double-cliquez ensuite sur « **Ajouter nouveau bloc** ».



9. Dans la nouvelle fenêtre, choisissez « **Fonction (FC)** » et donnez-lui le nom « *Programme_**presse*** ». Comme langage de programmation, choisissez « **LOG** ». La numérotation est automatique. Puisque FC1 est appelée de toute façon par son nom symbolique, le numéro n'a plus beaucoup d'importance. Acceptez les saisies avec « **OK** ».



10. Le bloc « **Programme_presse [FC1]** » s'ouvre automatiquement. Avant de pouvoir écrire le programme, cependant, on doit déclarer les variables locales, qui ne sont connues que dans le bloc.

    **Les variables sont divisées en 2 groupes :**

- **Les paramètres qui forment l'interface du bloc pour les appels dans le programme :**

| Type | Nom | Fonction | Reconnu dans |
|---|---|---|---|
| Paramètres d'entrée | Input | Paramètres dont la valeur est lue par le bloc | Fonctions, blocs de fonction et quelques types de blocs d'organisation |
| Paramètres de sortie | Output | Paramètres dont la valeur est écrite par le bloc | Fonctions et blocs de fonction |
| Paramètres d'entrée/sortie | InOut | Paramètres dont la valeur est lue par le bloc quand elle est appelée, et qui après traitement est écrite dans le même paramètre | Fonctions et blocs de fonction |

- **Les données locales utilisées pour un stockage des résultats intermédiaires :**

| Type | Nom | Fonction | Reconnu dans |
|---|---|---|---|
| Données locales temporaires | Temp | Variables utilisées pour un stockage temporaire des résultats intermédiaires. Les données temporaires sont conservées pour un cycle seulement | Fonctions, blocs de fonction et blocs d'organisation |
| Données locales statiques | Static | Variables utilisées pour un stockage statique des résultats intermédiaires dans le bloc de données d'instance. Les données statiques sont conservées jusqu'à leur réécriture, soit pour plusieurs cycles | Fonctions et blocs de fonction |

11. Déclarons maintenant les variables locales nécessaires pour notre exemple :

**Input :**

| | |
|---|---|
| Arret_urgence | Ici, la commande d'arrêt d'urgence est entrée |
| Demarrer | Le bouton de démarrage est entré ici |
| C_protection | L'état du capteur pour l'écran de protection est entré ici |
| C_verin | L'état du capteur pour la tige du vérin est entré ici |

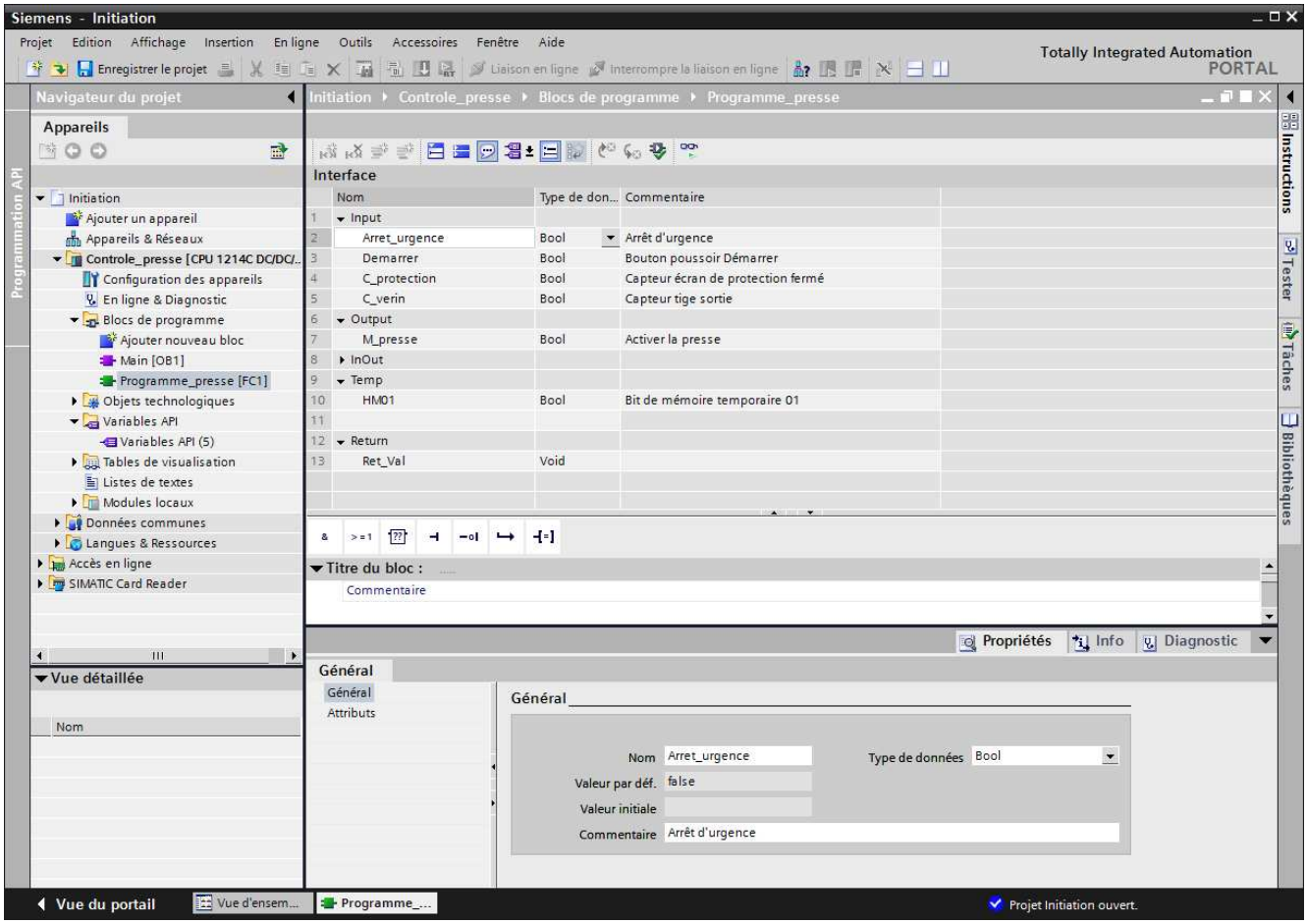
**Output :**

| | |
|---|---|
| M_presse | L'état de la sortie Vérin est écrite ici |

**Temp :**

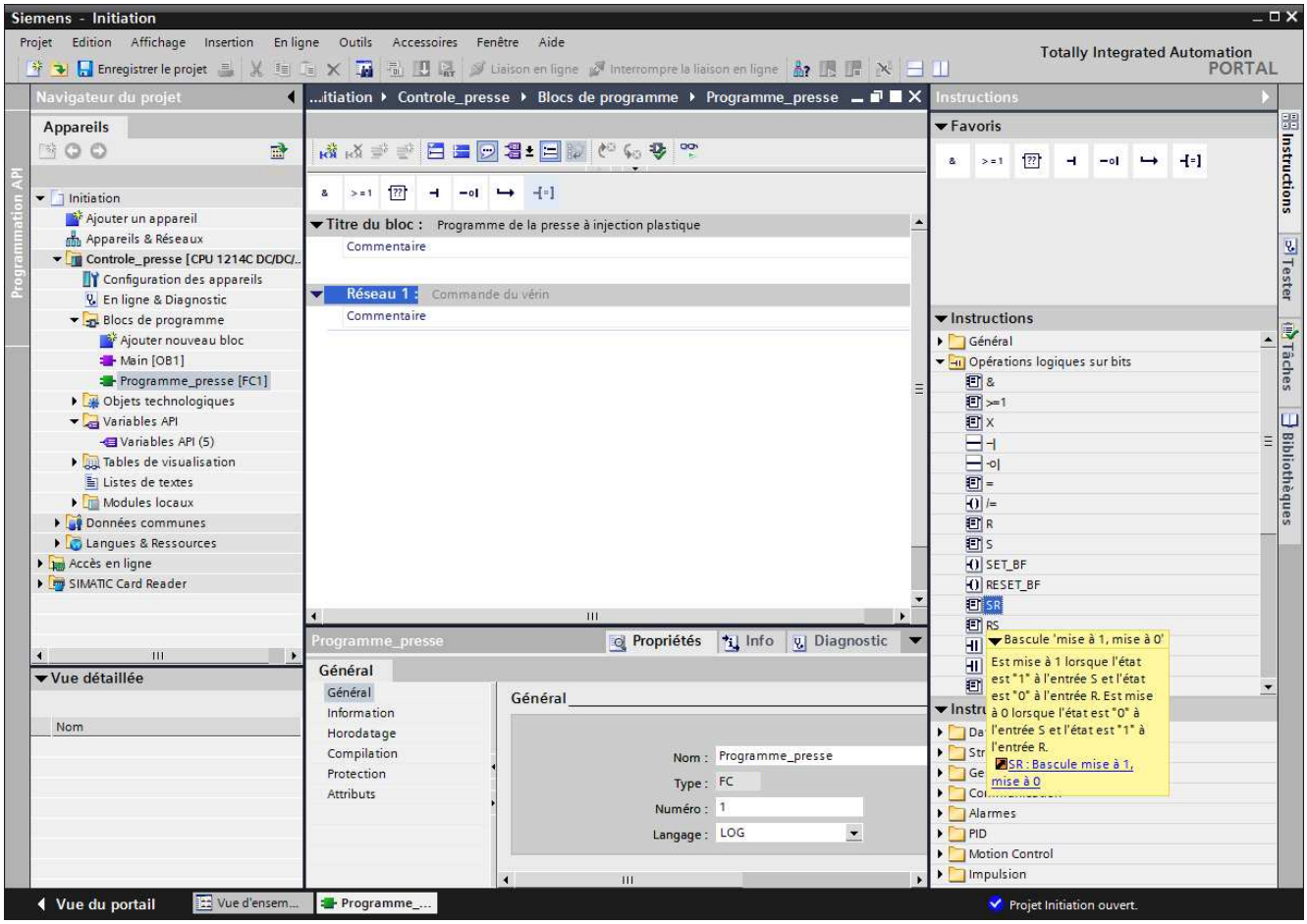
| | |
|---|---|
| HM01 | Mnémonique 01 pour la bascule RS |

Toutes les variables de cet exemple sont de type « Bool », ce qui veut dire qu'elles ne peuvent prendre que les valeurs « 1 » (vrai) ou « 0 » (faux).
Pour une meilleure compréhension, il est préférable d'écrire des commentaires pour chaque variable.

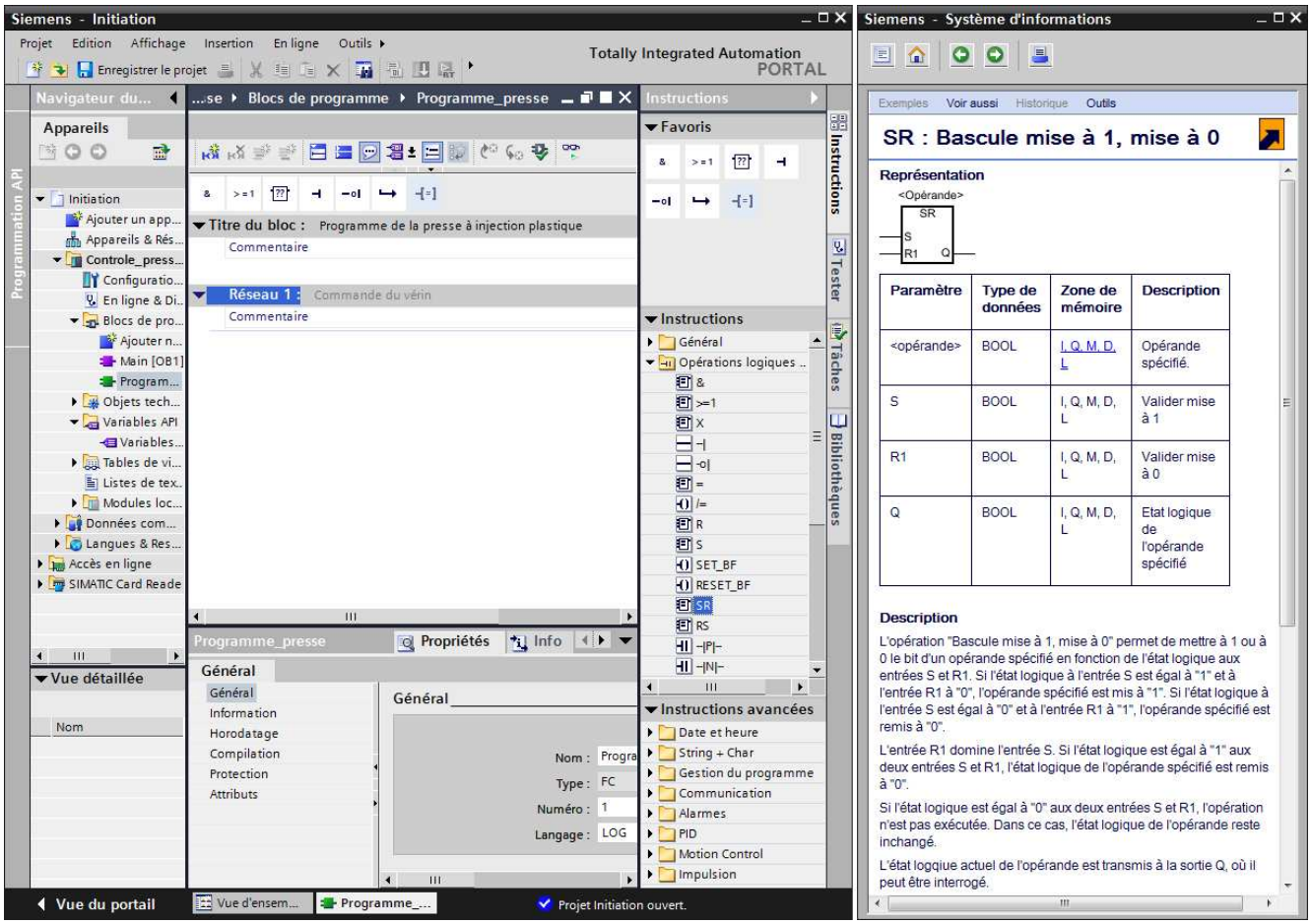


12. Maintenant qu'on a déclaré les variables locales, on peut commencer à programmer. Pour une meilleure vue d'ensemble, on programme sur des réseaux. Un nouveau réseau peut être insérer en cliquant sur le symbole « **Insérer réseau** ». Comme le bloc lui-même, chaque réseau doit être titré. Si une longue description est nécessaire, on peut le faire dans la partie « **Commentaire** ».

    Pour programmer notre solution, on a besoin d'une « **bascule SR** ». Elle se situe dans la fenêtre de droite, « **Instructions** > **Opérations logiques sur bits** > **SR** ». Si vous laissez la souris sur un objet tel que la bascule SR, des informations détaillées de l'objet s'affichent.



13. Si vous surlignez un objet et que vous appuyez sur **F1** sur votre clavier, une fenêtre d'aide en ligne apparaît à droite et vous fournit des informations sur l'objet sélectionné.

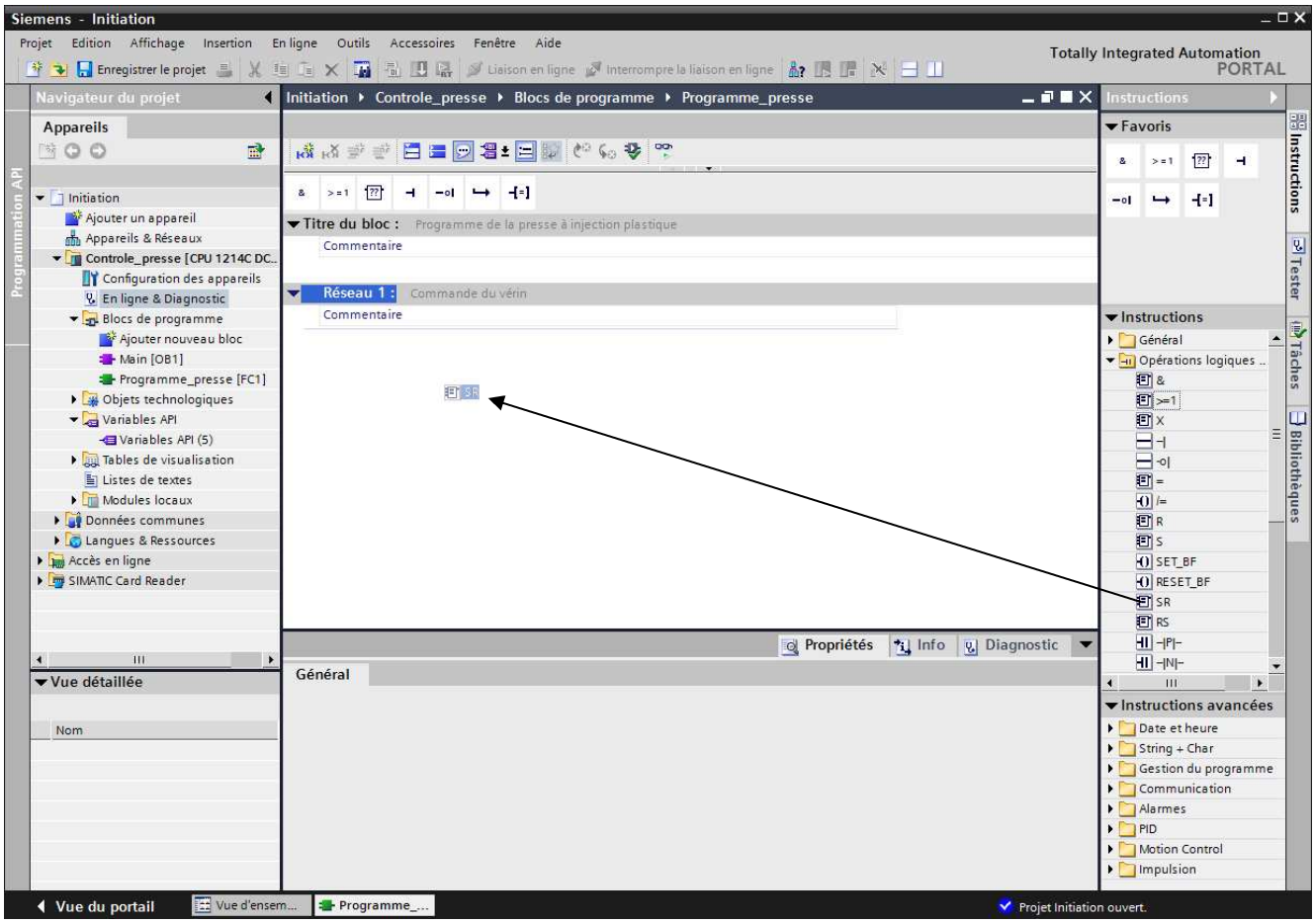



| Paramètre | Type de données | Zone de mémoire | Description |
|---|---|---|---|
| <opérande> | BOOL | I, Q, M, D, L | Opérande spécifié. |
| S | BOOL | I, Q, M, D, L | Valider mise à 1 |
| R1 | BOOL | I, Q, M, D, L | Valider mise à 0 |
| Q | BOOL | I, Q, M, D, L | Etat logique de l'opérande spécifié |

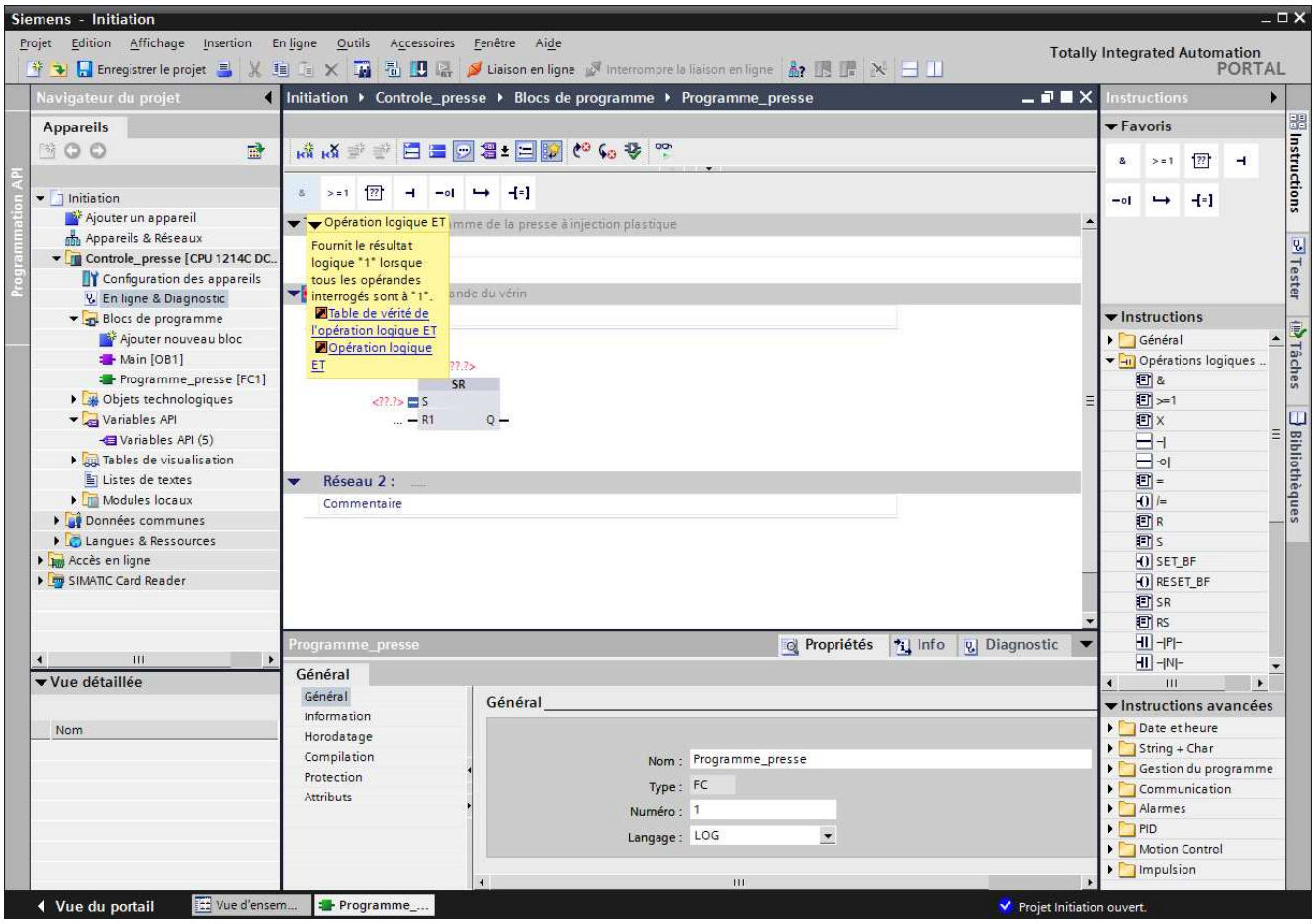
**Indication**

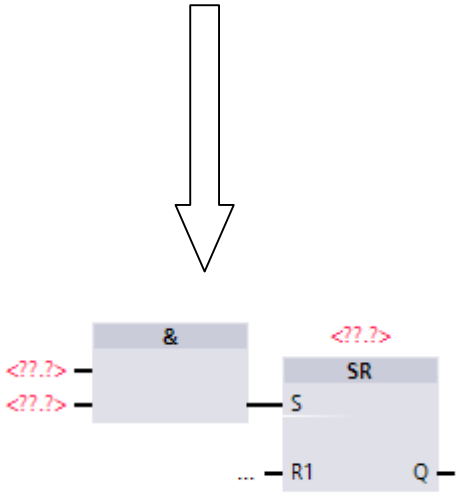
Ici, des informations détaillées sont données dans l'aide en ligne à propos de la fonction et du câblage d'une bascule SR.

14. Maintenant, glissez la bascule SR sous le réseau 1.

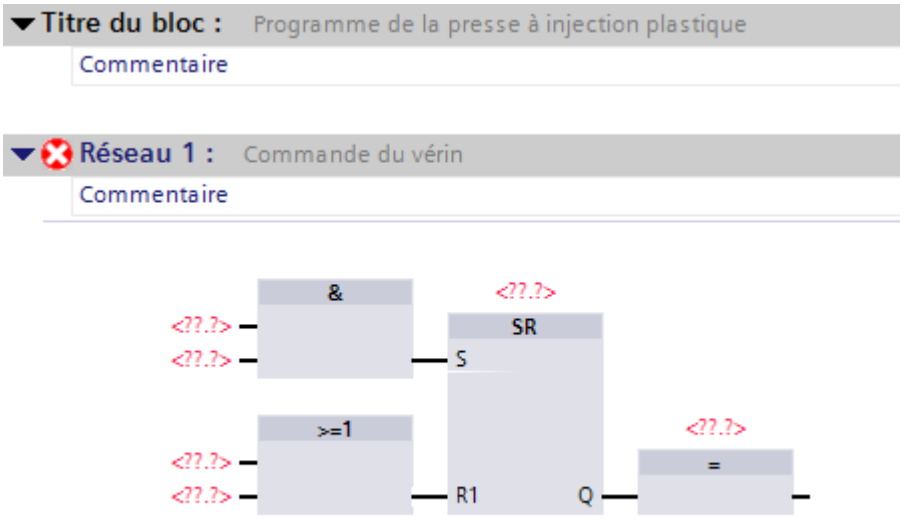


15. Ensuite, surlignez l'entrée **S**et de la bascule SR et cliquez sur le bloc « [ & ] » (**opération logique ET**) dans la liste des favoris.

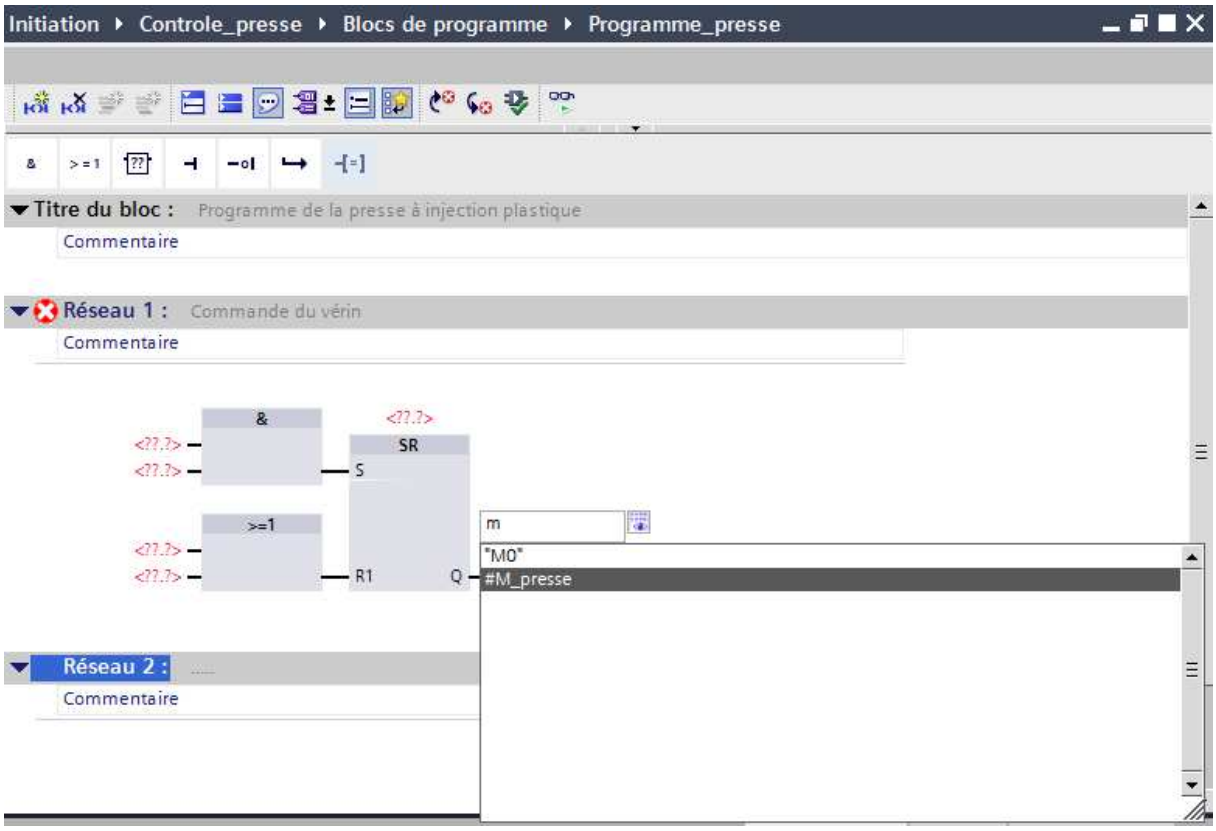


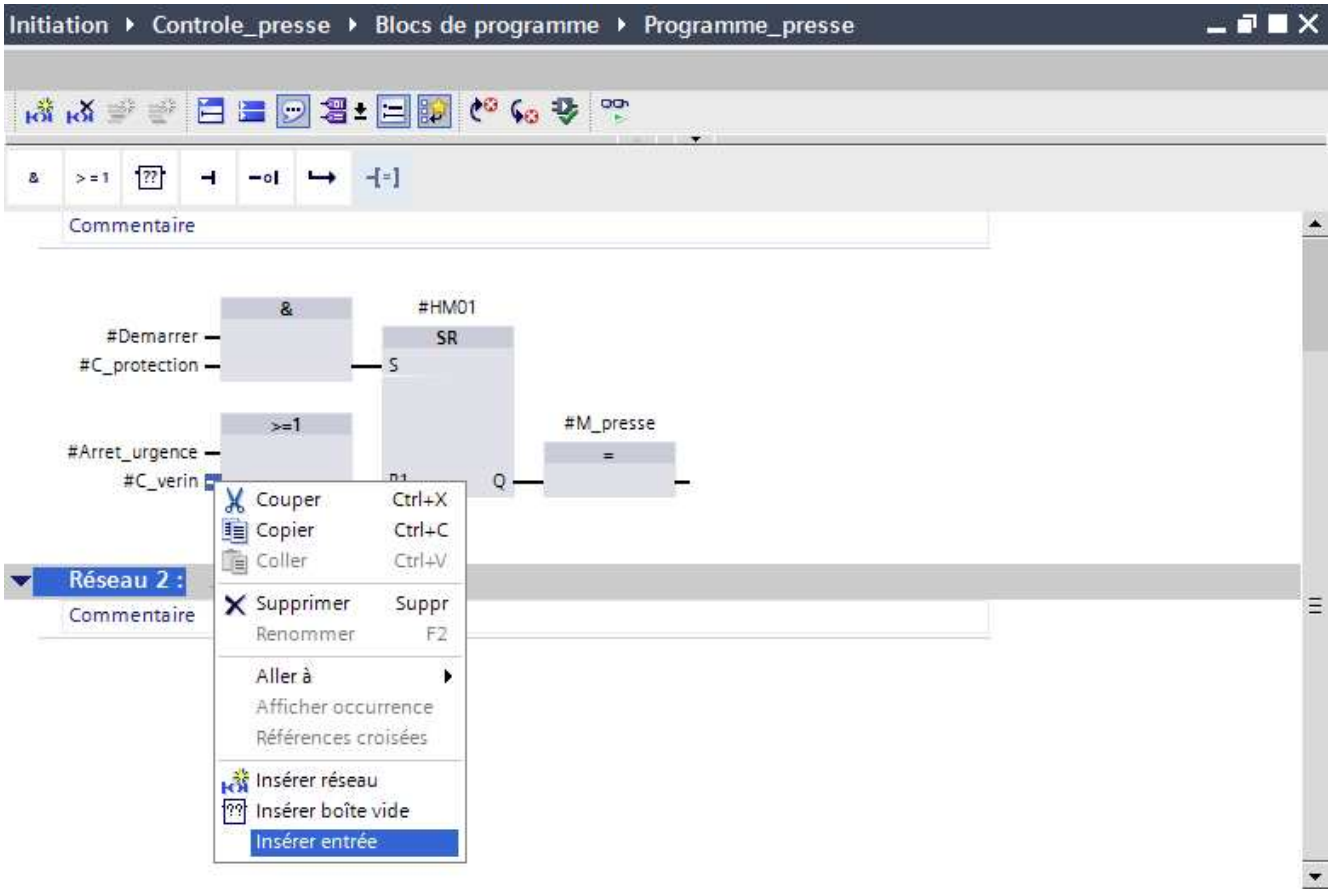
16. De la même manière, placez le bloc « [ >=1 ] » (**opération logique OU**) à l'entrée **R1** et le bloc « [ = ] » d'affectation à la sortie **Q** de la bascule.



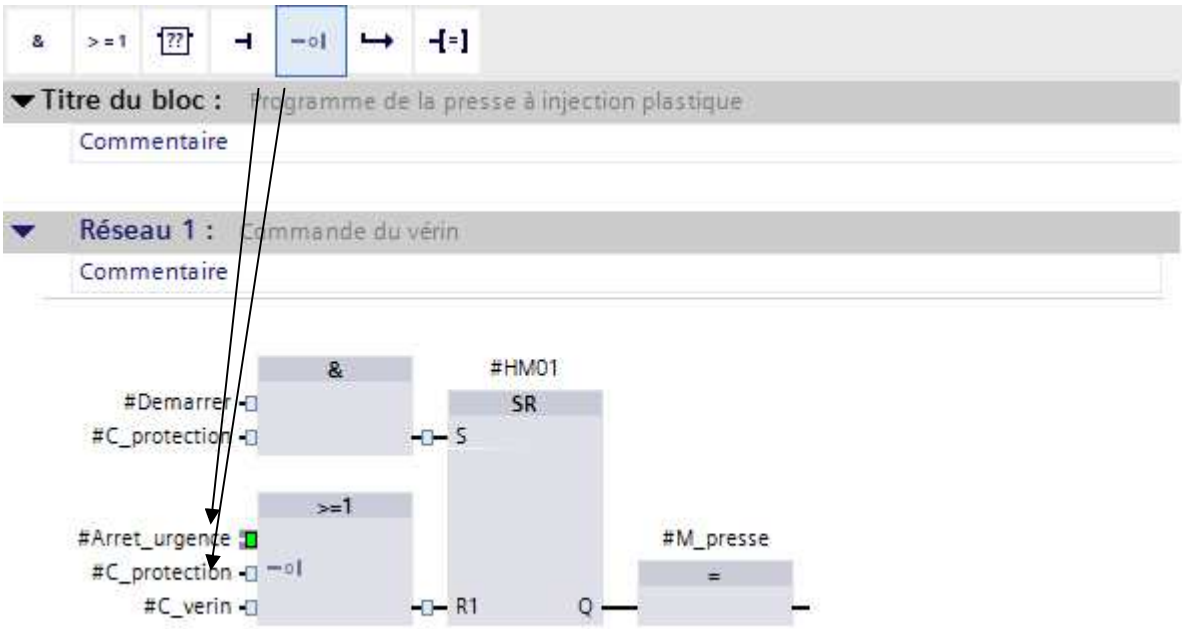

17. Maintenant, on appelle les variables locales. Il suffit d'entrer la première lettre de la variable dans le champ de saisie pour faire apparaître une liste, où on sélectionne ensuite la variable désirée.
Les variables locales sont toujours identifiées par le symbole « # » précédant leur nom (exemple : *#M_presse*).

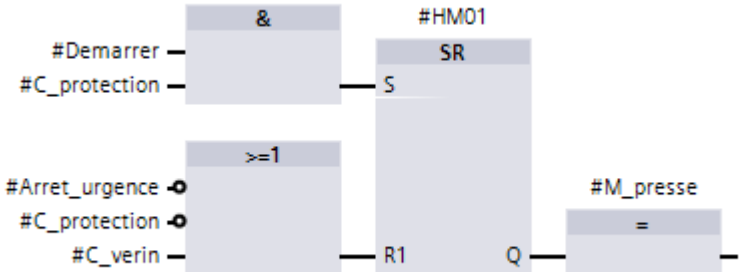


18. Ajoutez de même les autres variables locales. Pour l'opération OU, une autre entrée doit être insérée. Pour cela, surlignez l'entrée du dessous et avec un clic-droit sélectionnez « **Insérer entrée** », et assignez la variable.

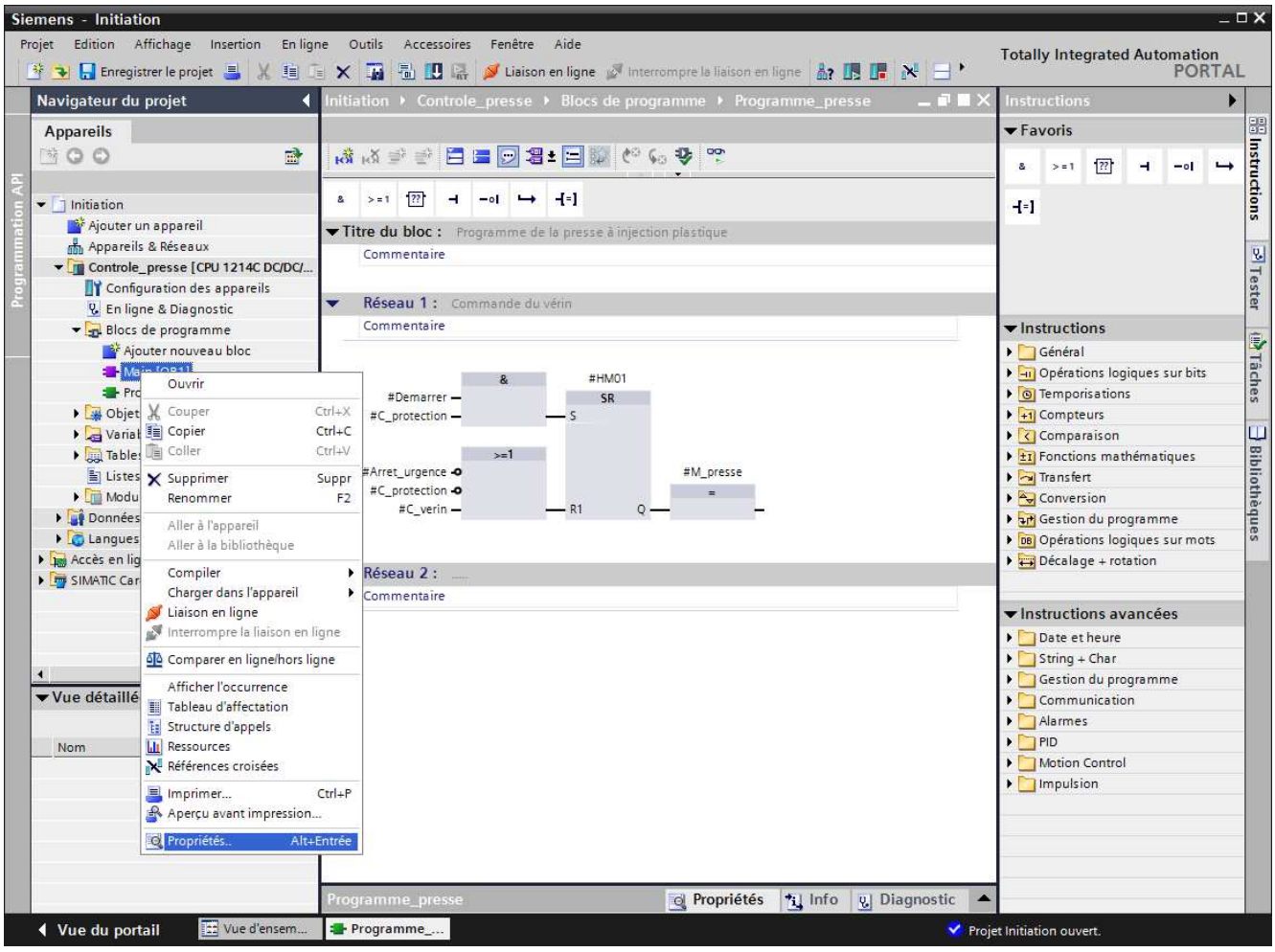


Pour inverser une entrée, glissez simplement le symbole « **Négation** » à partir de la barre des favoris sur l'entrée à inverser. On doit inverser les entrées de « **#Arret_urgence** » et de « **#C_protection** ».

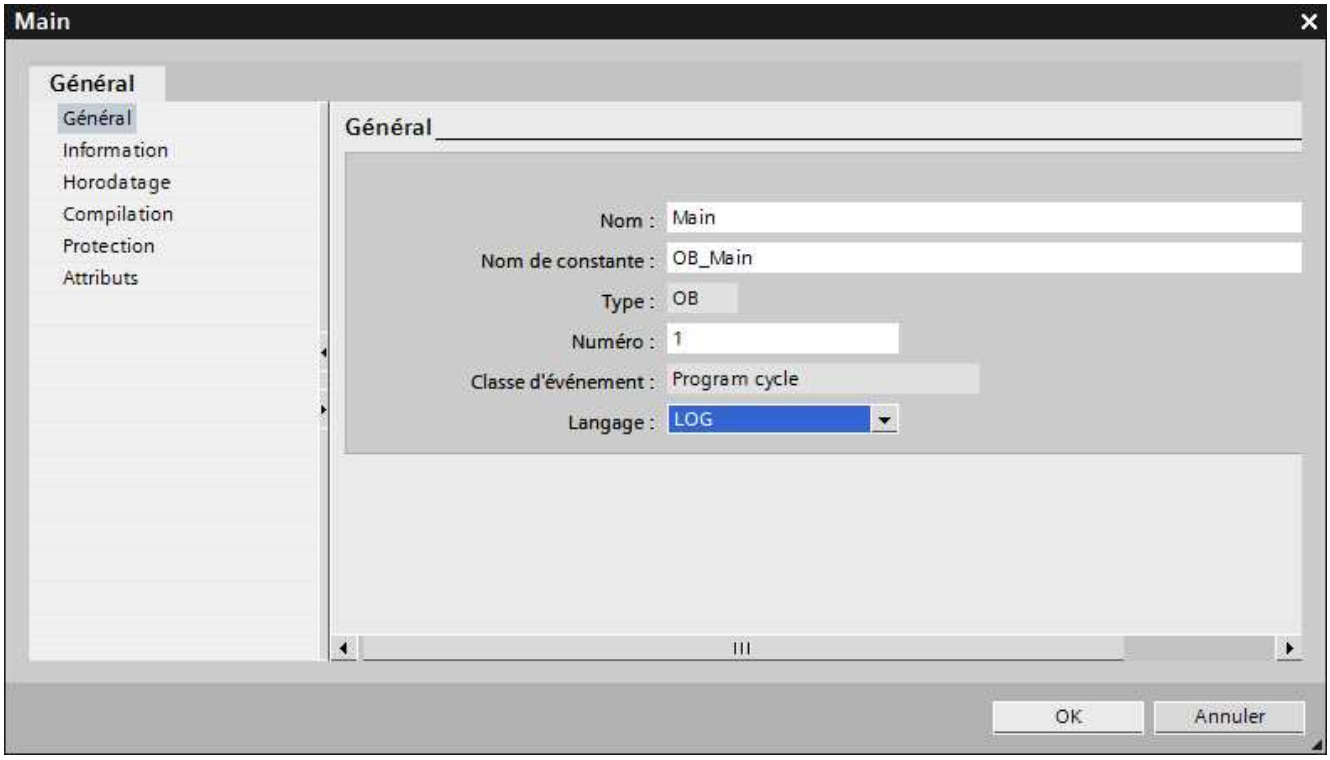


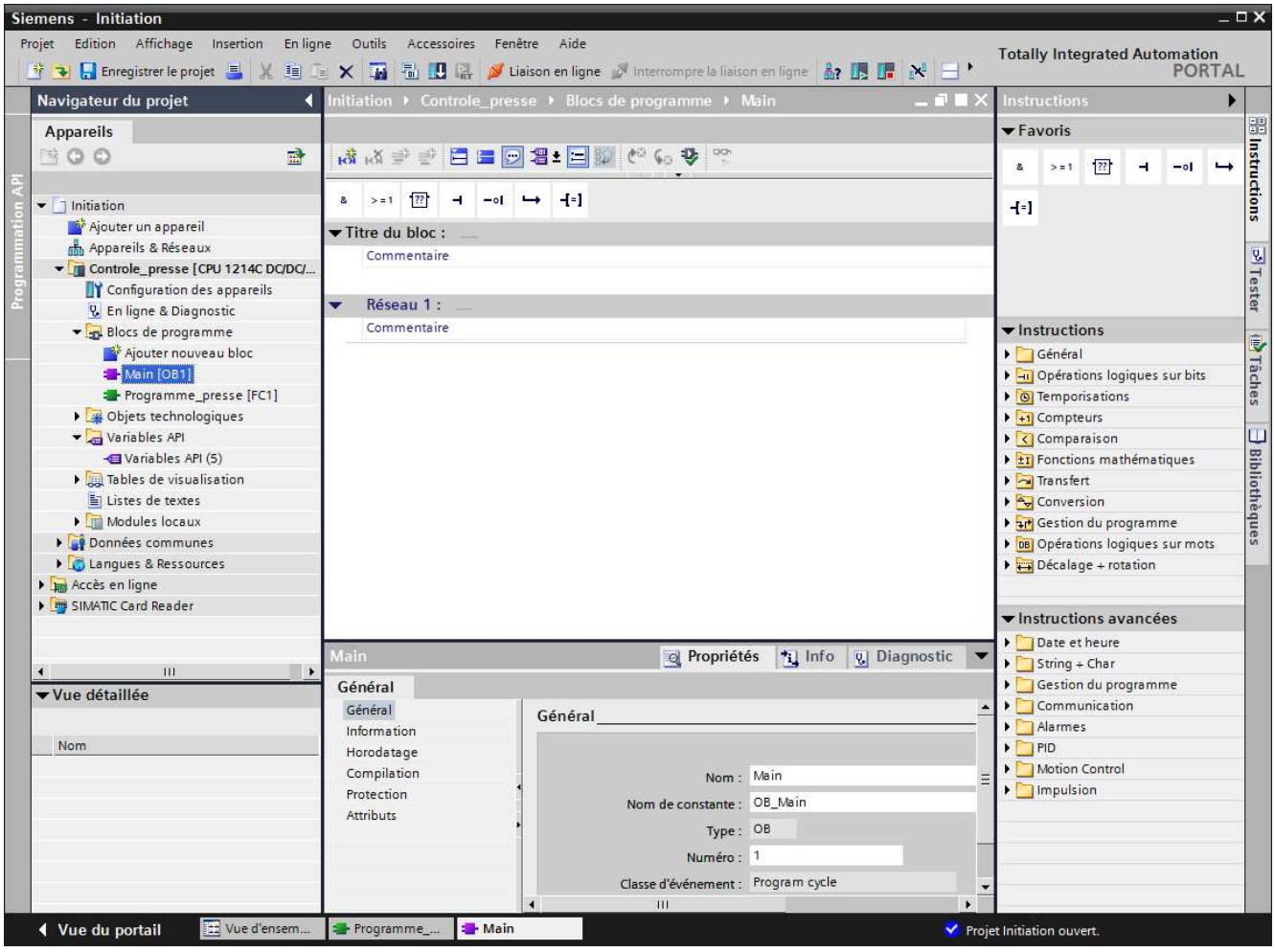
19. Ensuite, allez dans les « **Propriétés** » du bloc d'organisation « **Main [OB1]** ». Les propriétés du bloc peuvent y être modifiées.



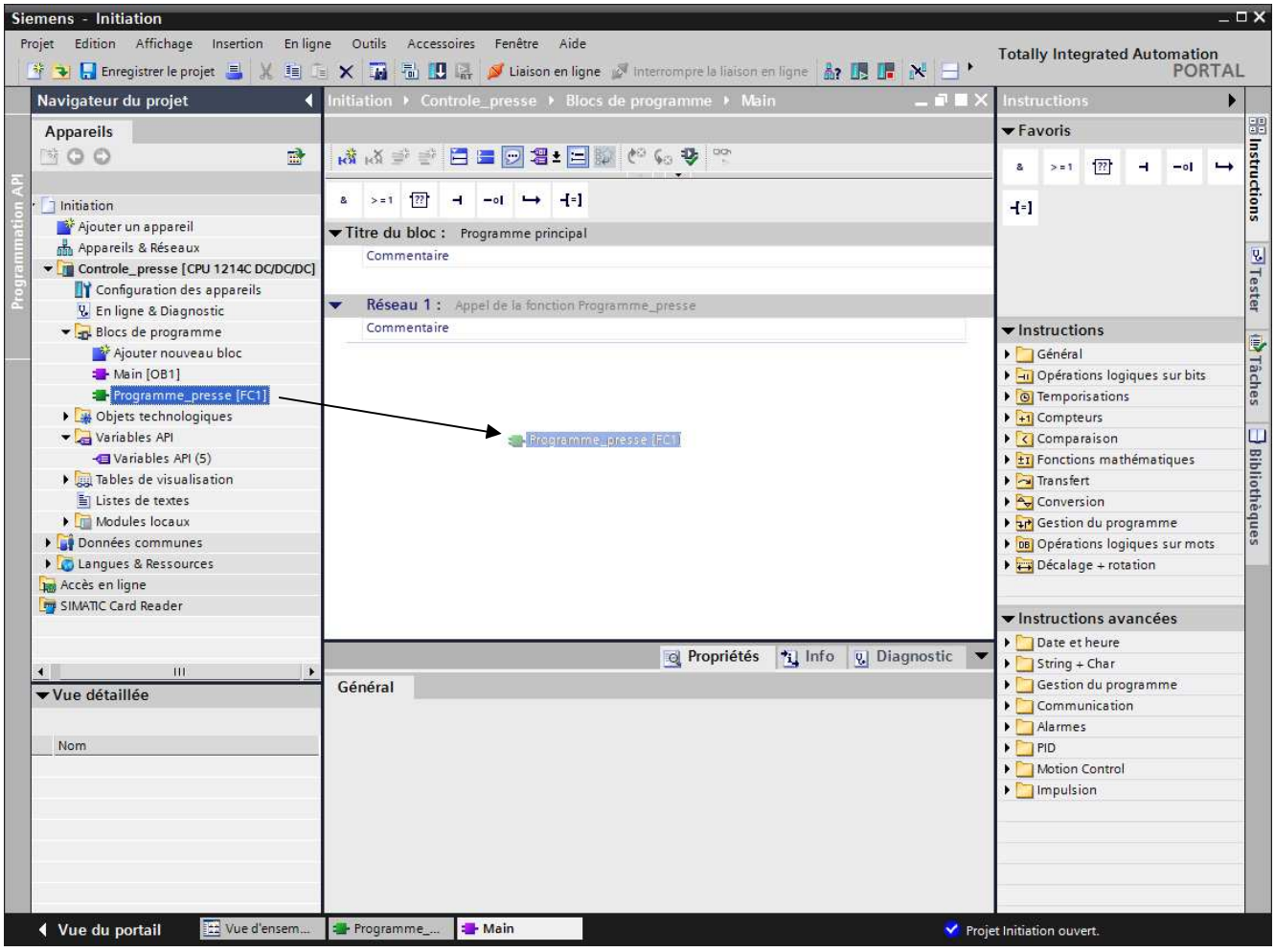
20. Dans les propriétés, sélectionnez le **langage** de programmation « **LOG** ».



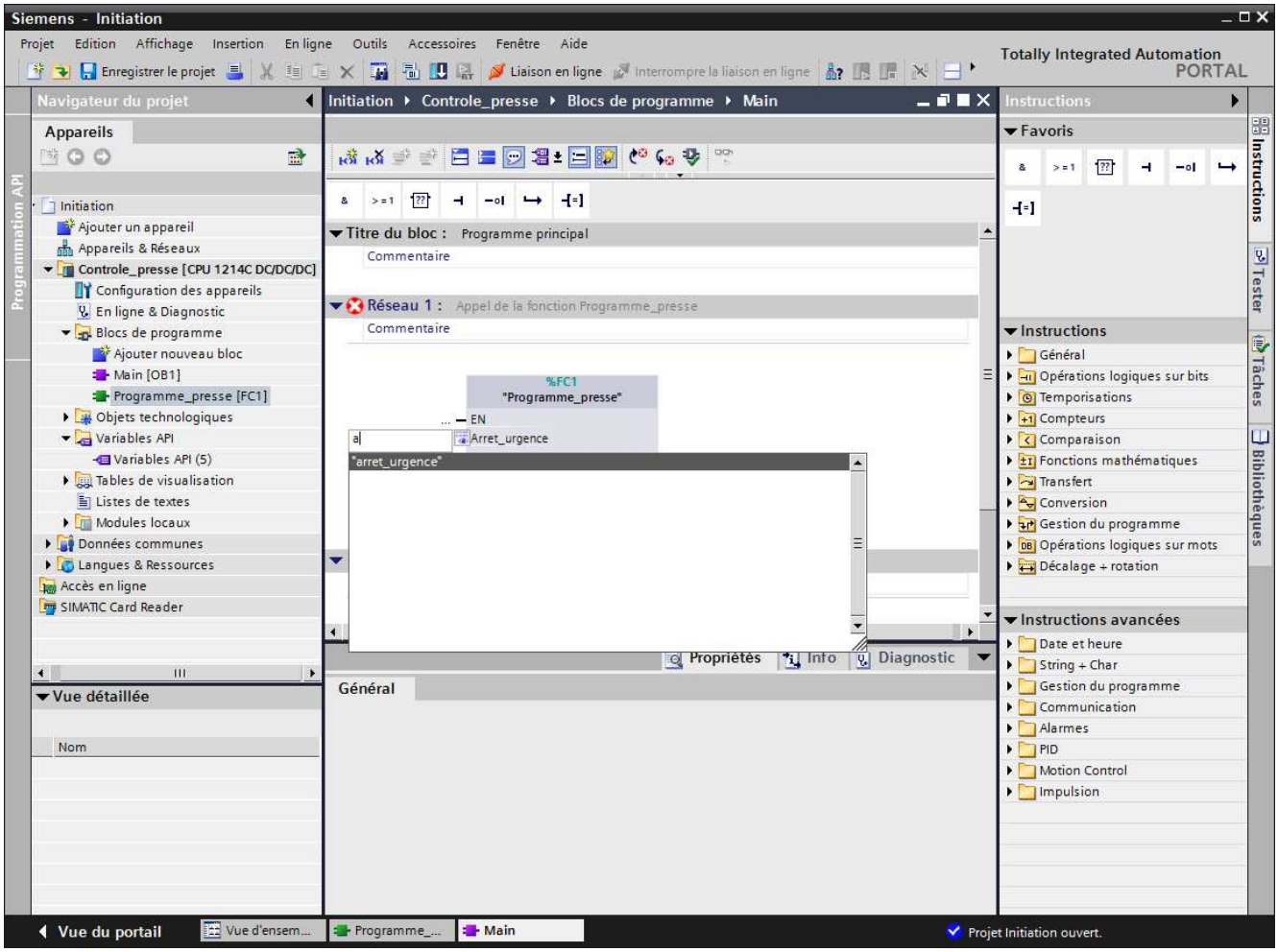
21. Comme indiqué précédemment, le bloc « **Programme_presse** » doit être appelé depuis le bloc Main [OB1]. Autrement, le bloc ne serait pas pris en compte du tout. Ouvrez-le en double-cliquant sur « **Main [OB1]** ».



22. Le bloc « **Programme_presse** » peut ensuite être simplement déposé dans le réseau 1 de Main [OB1] grâce à un glisser-déposer. N'oubliez pas de commenter les réseaux !

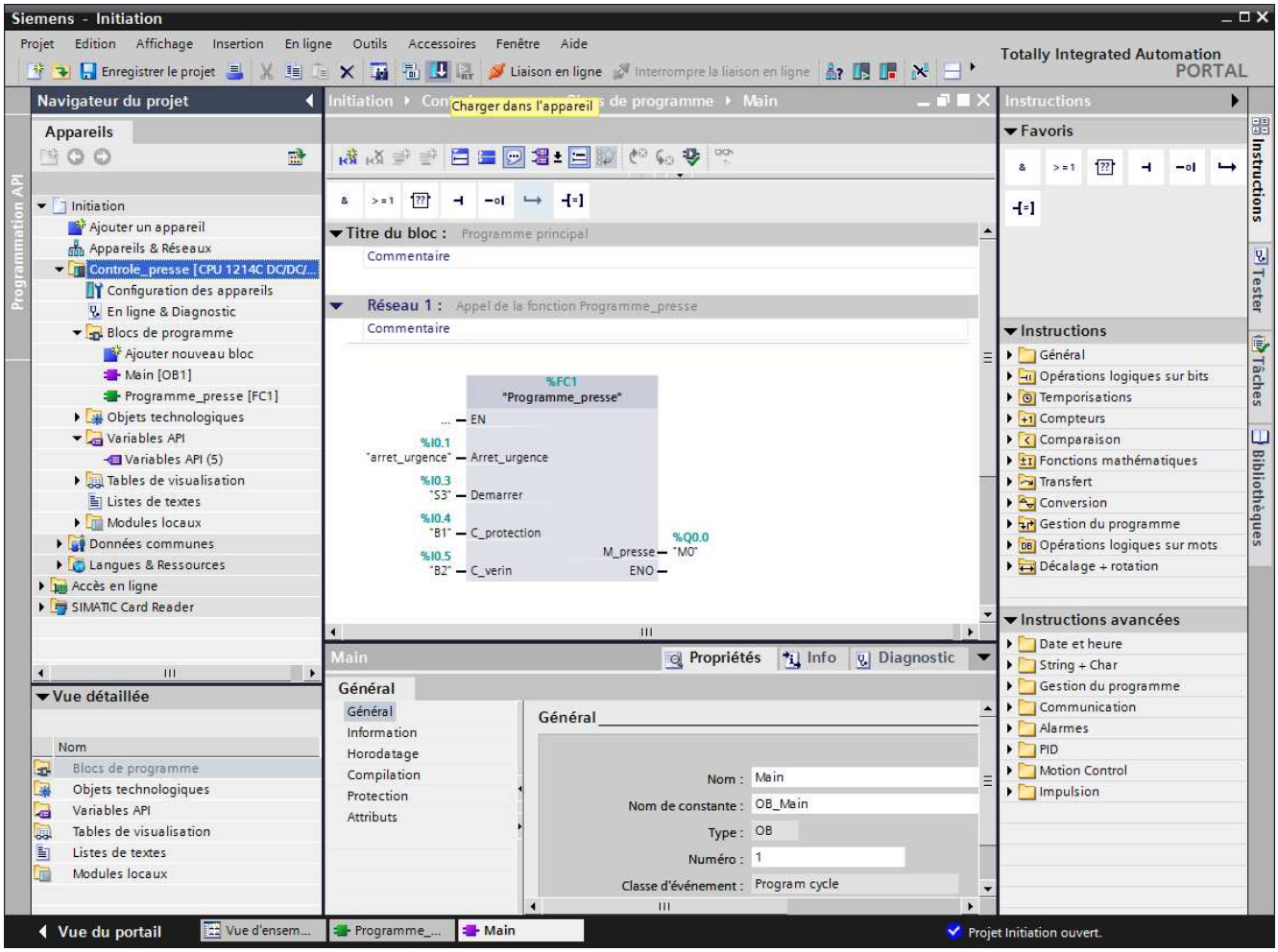


23. Ensuite, les paramètres de l'interface du bloc « Programme_presse » doivent être connectés aux variables globales de l'API. Il suffit là aussi d'entrer la première lettre de la variable globale pour faire apparaître une liste, dans laquelle on sélectionne la variable voulue.

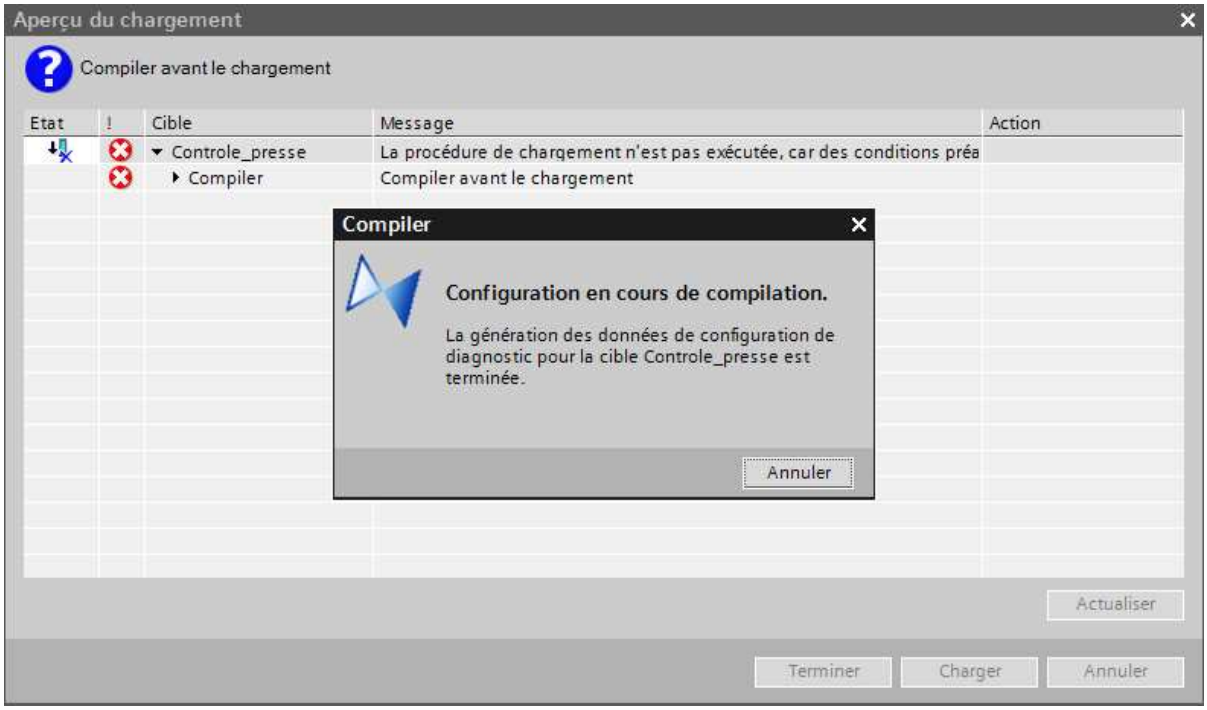


24. De cette façon, ajoutez les autres variables globales comme indiqué sur la capture d'écran ci-dessous. Il sera alors temps de sauvegarder votre projet, en cliquant sur Enregistrer le projet.

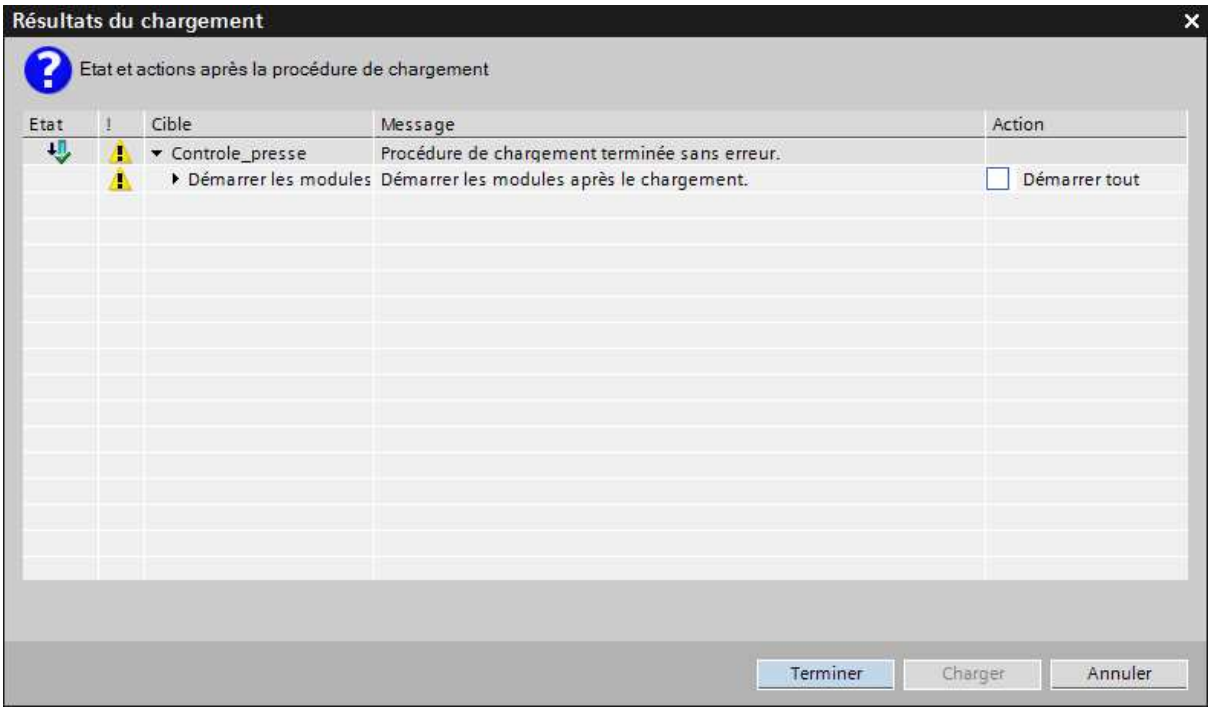
25. Pour charger le programme entier dans la CPU, surlignez d'abord le dossier « **Contrôle_presse** » puis cliquez sur le symbole « **Charger dans l'appareil** ». Cela va ouvrir une nouvelle fenêtre.



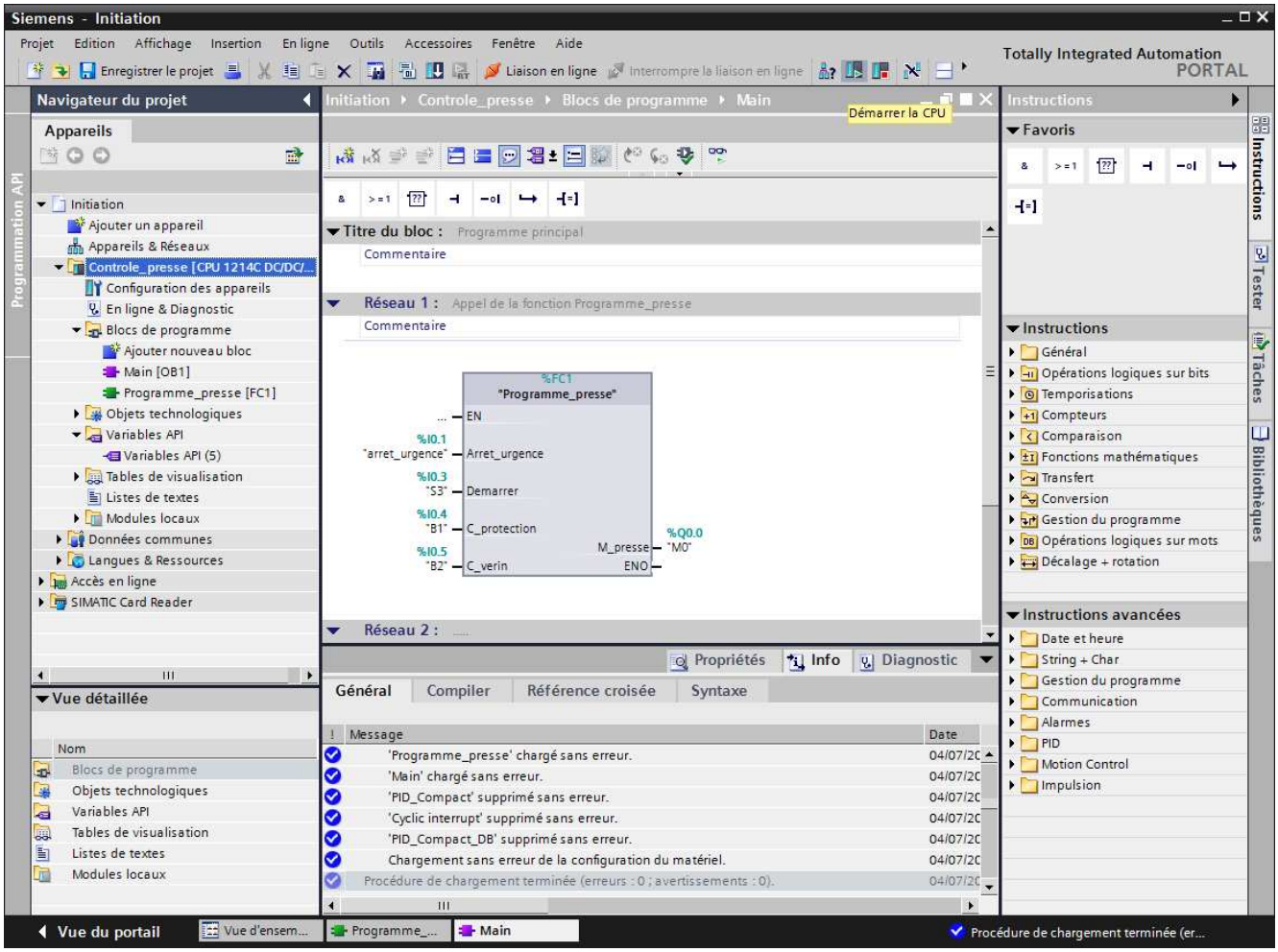
26. Pendant la compilation, l'état de progression est affiché dans une fenêtre.



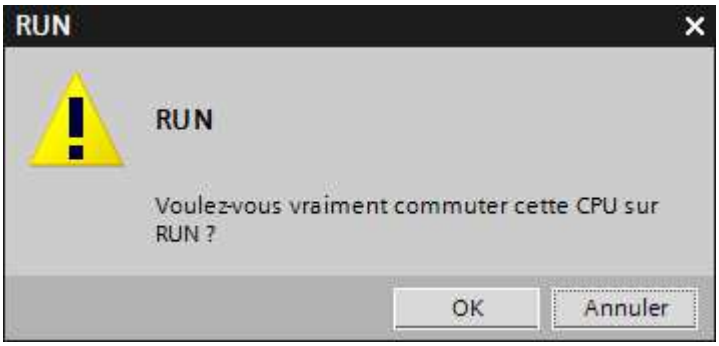

27. Si la compilation s'est correctement déroulée, cela s'affiche dans la fenêtre. Cliquez maintenant sur « **Charger** », puis « **Terminer** ».

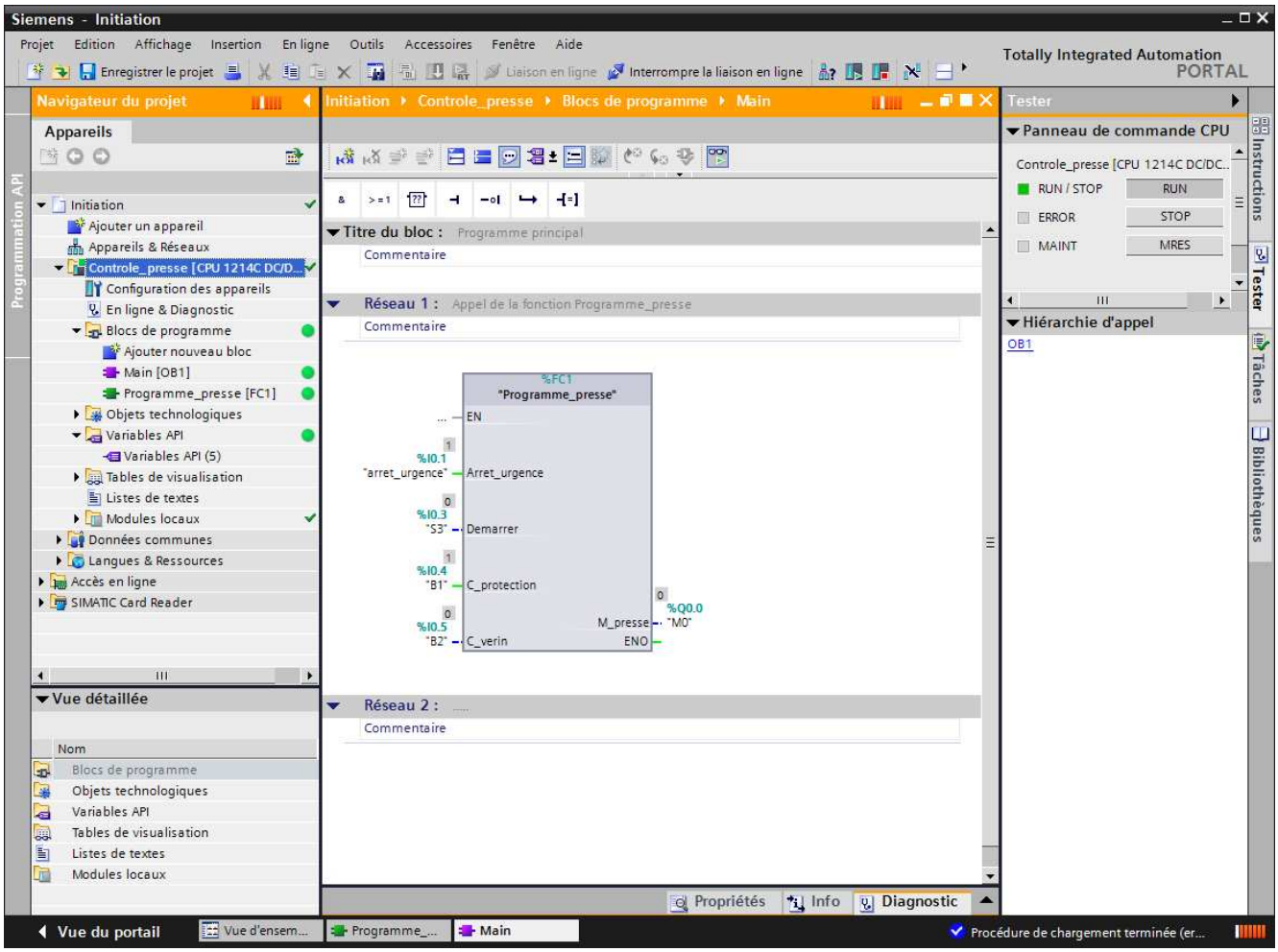


28. Maintenant, démarrez la CPU en cliquant sur le symbole « **Démarrer la CPU** ».




29. Confirmez le fait que vous vouliez vraiment commuter la CPU sur *RUN* en cliquant sur « **OK** ».



30. Cliquez finalement sur l'icône « **Activer/désactiver visualisation du programme** ».
Grâce à ce bouton, il est possible de surveiller l'état des variables pendant que vous testez le programme en commutant les interrupteurs de la maquette.

Remarquez que la fenêtre « *Navigateur du projet* » est devenue orange, ce qui signifie que vous travaillez désormais en ligne avec l'automate.