Blocs Temporisations IEC et Compteurs IEC pour le SIMATIC S7-1200

Objectif

Dans ce TP3, vous allez apprendre à utiliser les instructions supplémentaires pour créer des temporisations et des compteurs lors de la programmation du SIMATIC S7-1200. Le TP montre comment programmer une temporisation dans une fonction avec un DB d'instance (une instance unique comme mémoire affectée). Il montre également comment programmer un compteur, qui lui n'est pas affecté d'un DB d'instance pour mémoire, mais d'une multi-instance.

Pré-requis

Les connaissances suivantes sont requises pour l'étude de ce module :

- Notions de base en programmation d'API avec TIA Portal (par exemple, TP1 – « Initiation à la programmation du SIMATIC S7-1200 avec TIA Portal VX »)
- Les différents types de blocs utilisés pour la programmation du SIMATIC S7-1200 (TP2 – « Types de bloc du SIMATIC S7-1200 »)

1 Notions d'Instance et de Multi-Instances dans la programmation du SIMATIC S7-1200

L'appel d'un bloc fonctionnel est appelé **instance**. Pour chaque appel d'un FB, une zone mémoire lui est affectée, un **DB d'instance**, contenant les données utiles au traitement du bloc. Ainsi, les paramètres locaux et les données statiques des FB sont stockés à l'intérieur.

Les variables déclarées dans le FB déterminent la structure du DB d'instance.

Application des instances uniques et des multi-instances :

Les blocs de données d'instance peuvent être affectés comme suit :

- Appel en tant qu'**instance unique**
  - Un DB d'instance différent pour chaque instance d'un FB
- Appel en tant que **multi-instance**
  - Un seul DB d'instance pour plusieurs instances d'un ou plusieurs FB

1.1 Blocs de données d'instance/Instances uniques

L'appel d'un bloc fonctionnel FB auquel on assigne son propre bloc de données d'instance est appelé **instance unique**.

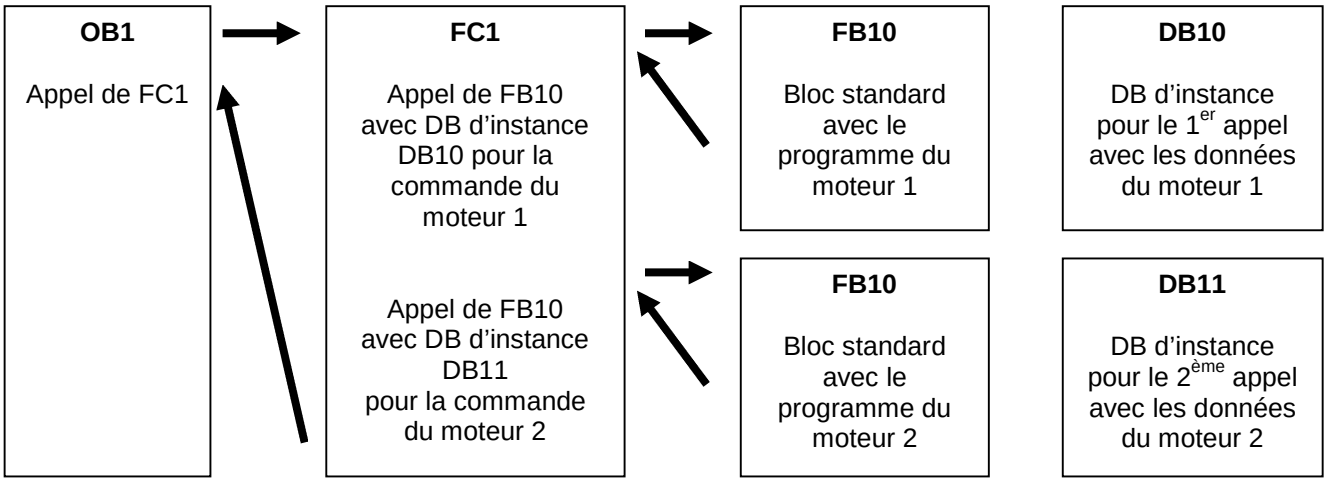
Si le bloc fonctionnel a été créé en suivant les règles des blocs standards (se référer au TP2), il peut être appelé un nombre illimité de fois.

Cependant, pour chaque appel en tant qu'instance unique, vous devez assigner un bloc de données d'instance différent à chaque fois.

Exemple d'instances uniques :

La figure ci-dessous montre deux moteurs contrôlés par un bloc fonctionnel FB10 et deux blocs de données différents.

Les différentes données de chaque moteur (par exemple la vitesse, temps d'allumage, temps total de mise en marche…) sont stockées dans les différents blocs de données d'instance DB10 et DB11.



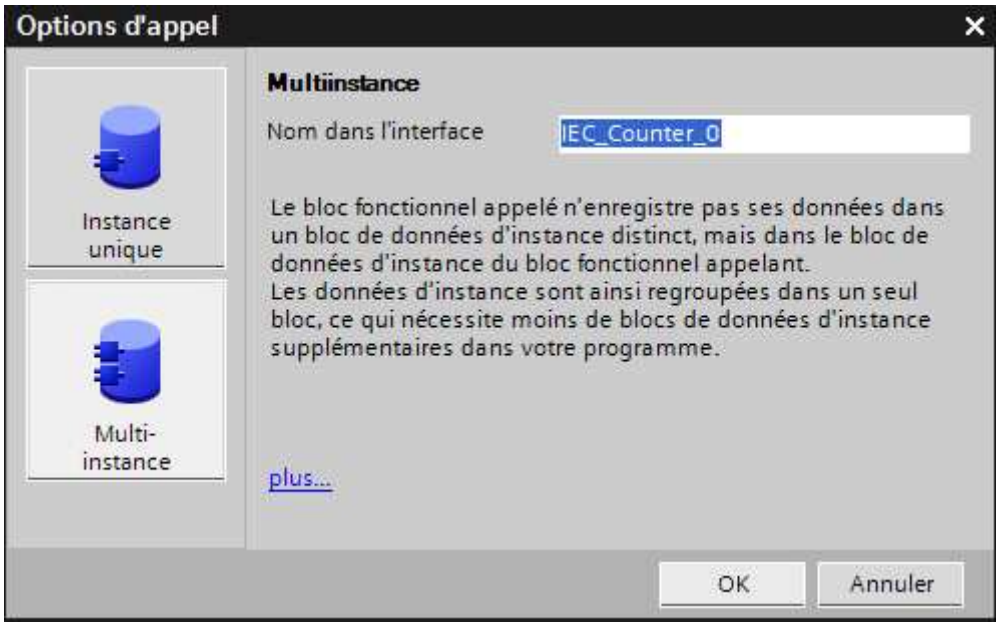
Indication

Quelques instructions comme les temporisations et les compteurs se comportent comme des blocs fonctionnels. S'ils sont appelés, ils représentent aussi des instances et doivent être affectés d'une zone mémoire, sous la forme d'un DB d'instance, par exemple.

1.2 Multi-instances

À cause de la capacité mémoire de la CPU utilisée, il est possible que vous ne vouliez ou que vous ne puissiez allouer qu'un nombre limité de blocs de données pour des données d'instance.

Si d'autres blocs de fonction existants comme les tempos ou les compteurs sont appelés dans un bloc de fonction de votre programme utilisateur, il est possible d'appeler ces FB supplémentaires sans leur DB d'instance propre.
Il suffit pour cela de sélectionner dans les options d'appel « **Multi Instance** ».



Indication :

Les multi-instances permettent de placer les données d'un FB qui a été appelé dans le DB d'instance du FB qui l'a appelé.

Dans ce cas, le bloc qui appelle doit toujours être un bloc fonctionnel.

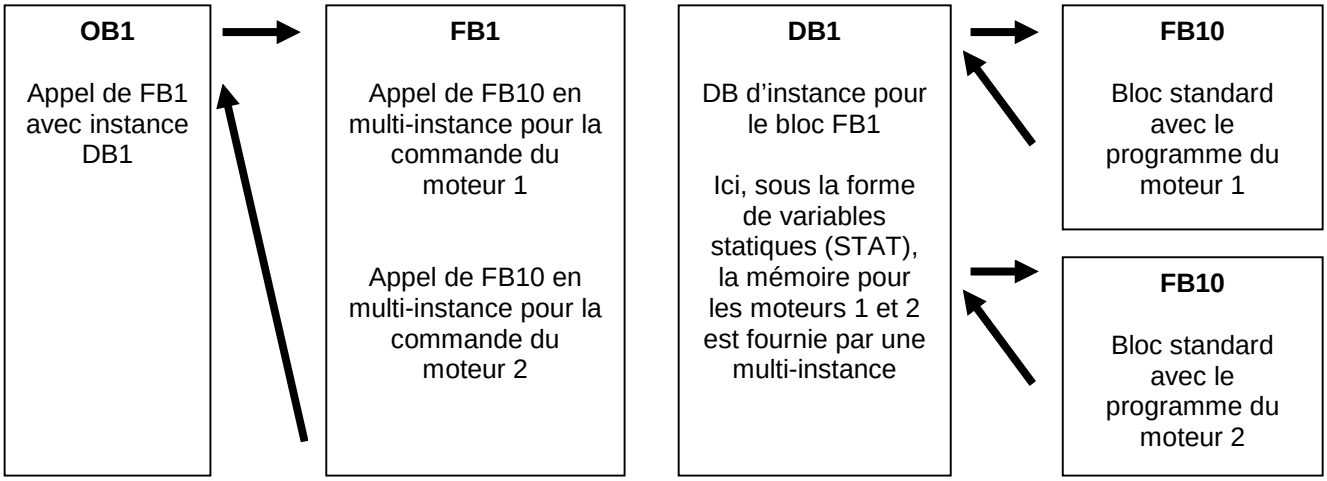
De cette façon, vous concentrez les données d'instance dans un seul bloc de données d'instance, i.e. vous pouvez utiliser le nombre de DB disponibles plus efficacement.

Ceci, en passant, doit toujours être fait si le bloc qui appelle doit être réutilisable comme un bloc standard.

Exemple de multi-instances :

La figure ci-dessous montre deux moteurs contrôlés par un bloc fonctionnel FB10 qui est appelé deux fois.

Les différentes données des deux moteurs (par exemple la vitesse, temps d'allumage, temps total de mise en marche…) sont stockées en tant que multi-instances dans un seul bloc de données d'instance DB1 du bloc fonctionnel FB1 appelant.



Indication

Quelques instructions comme les temporisations et les compteurs se comportent comme des blocs fonctionnels. S'ils sont appelés, ils représentent aussi des instances et doivent être affectés d'une zone mémoire, qui peuvent également être fournies en multi-instances.

2 Exemple d'application : Contrôle d'une presse avec Tempo et DB d'instance

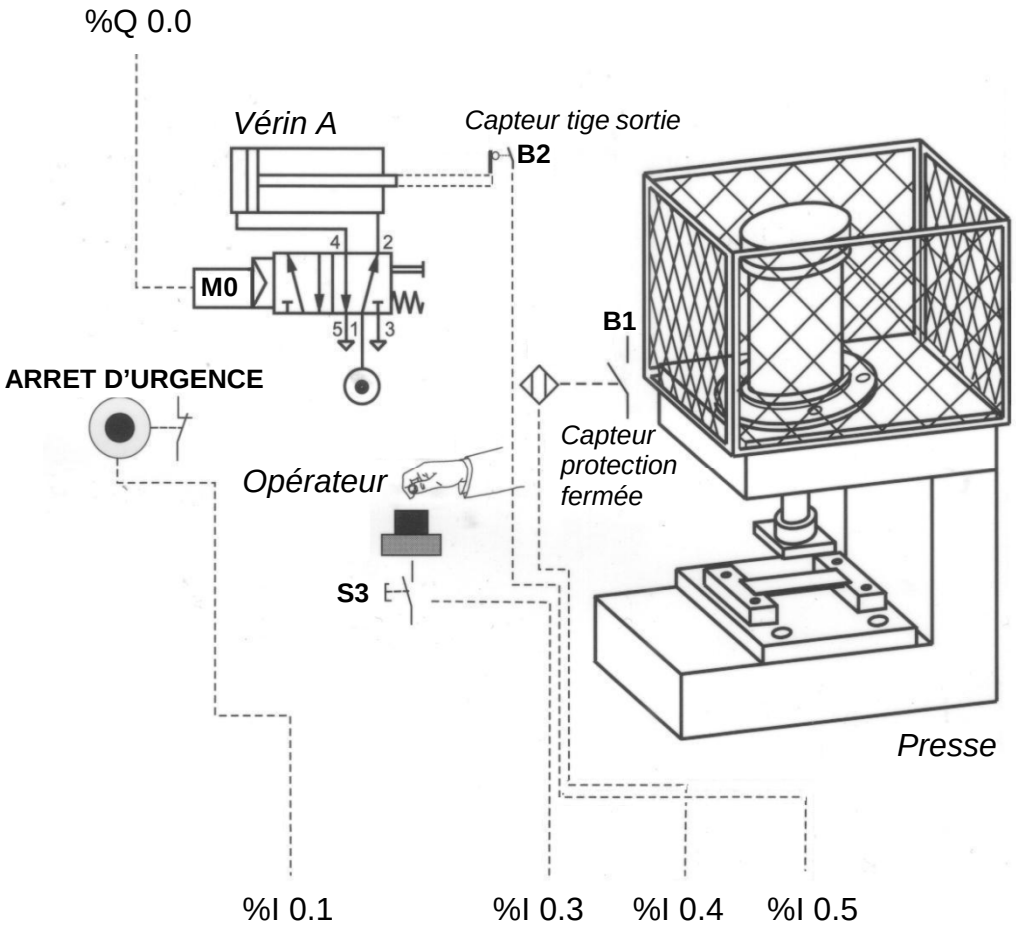
On va ajouter une composante temporelle au contrôle de la presse du TP 1 pour illustrer ce cours.

L'application à réaliser est la suivante :
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 ou quand le capteur **PROTECTION FERMEE B1** ne répond plus.

Si le capteur **VERIN TIGE SORTIE B2** répond, la presse doit se retirer après 5 secondes.
Un DB d'instance est utilisé comme mémoire pour la temporisation.

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



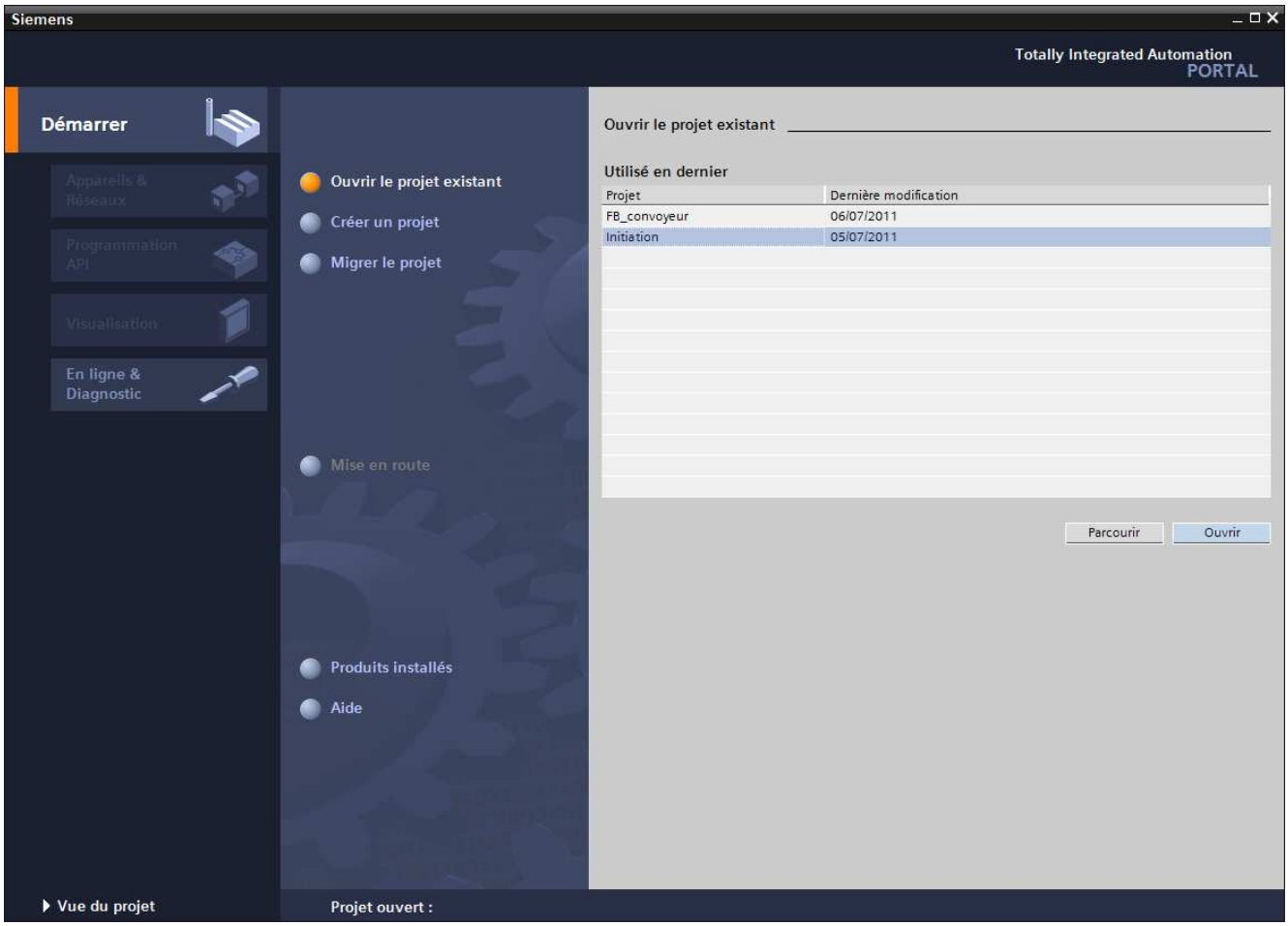
2.2. Programmation de la presse avec Retard à la montée avec le SIMATIC S7-1200

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.

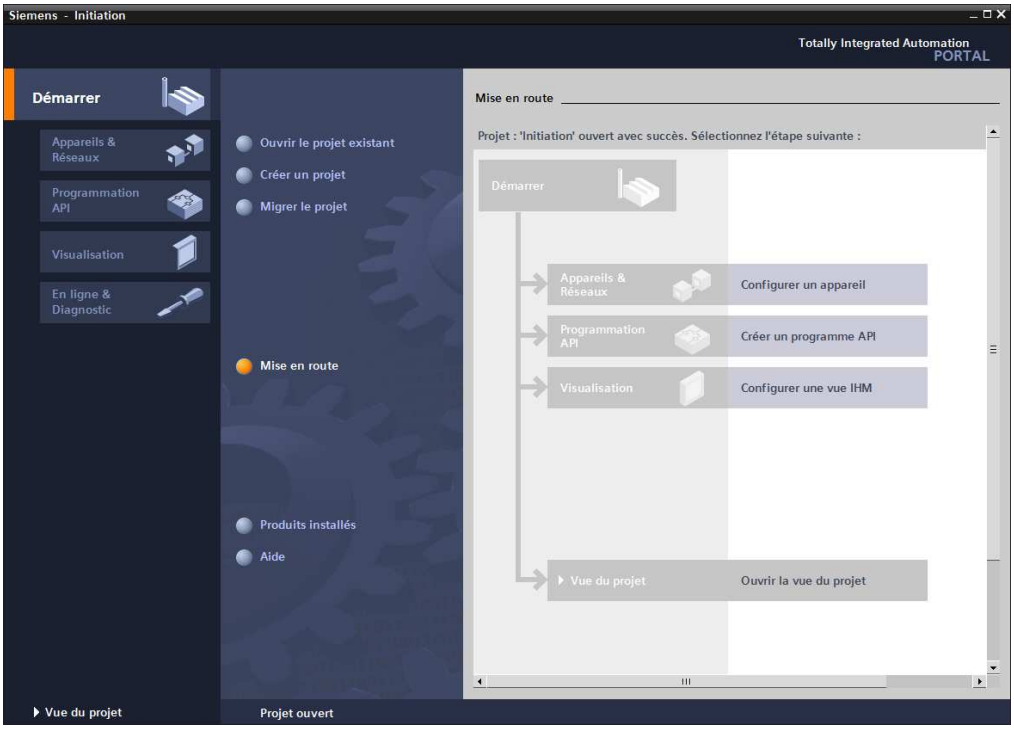
Dans les étapes suivantes, nous allons ouvrir un projet pour le SIMATIC S7-1200, l'enregistrer sous un nouveau nom, et le modifier pour qu'il réponde aux nouvelles exigences.

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

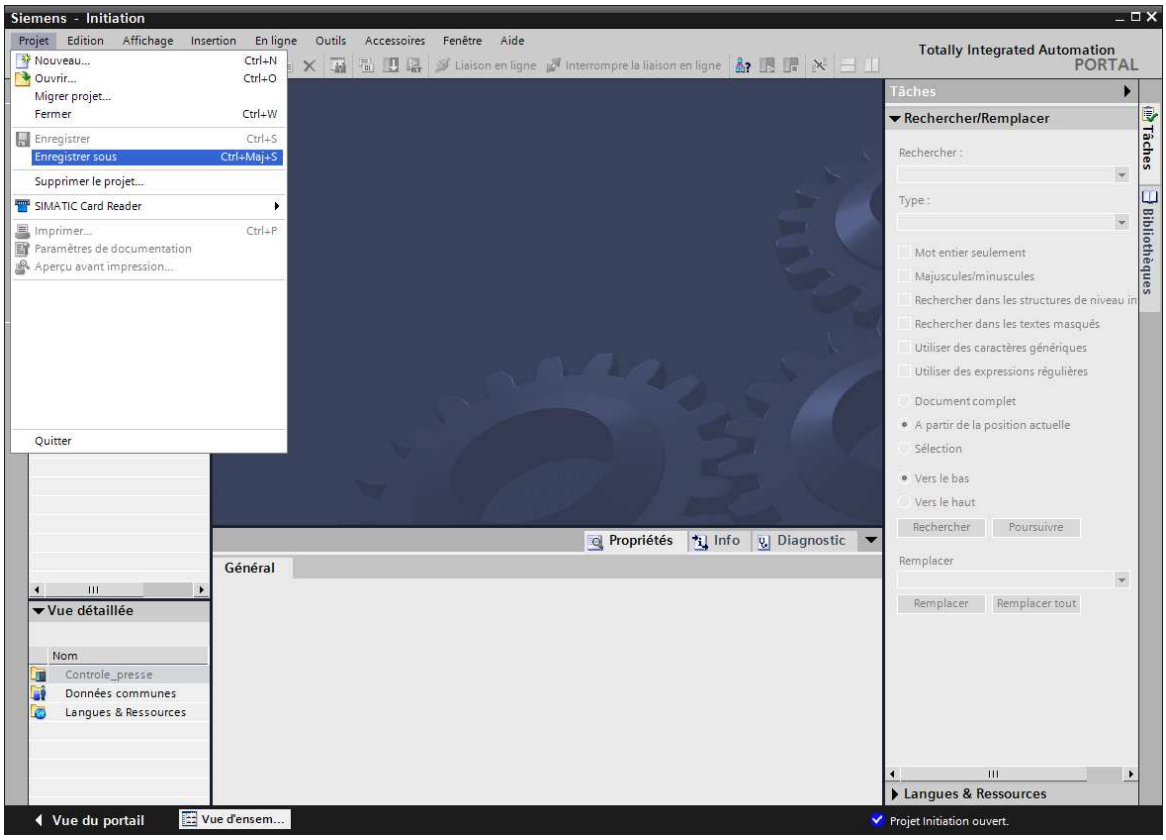
2. Nous allons maintenant ouvrir le projet « *Initiation* » du module M1 dans la vue du portail. Ce projet servira de base pour le programme.



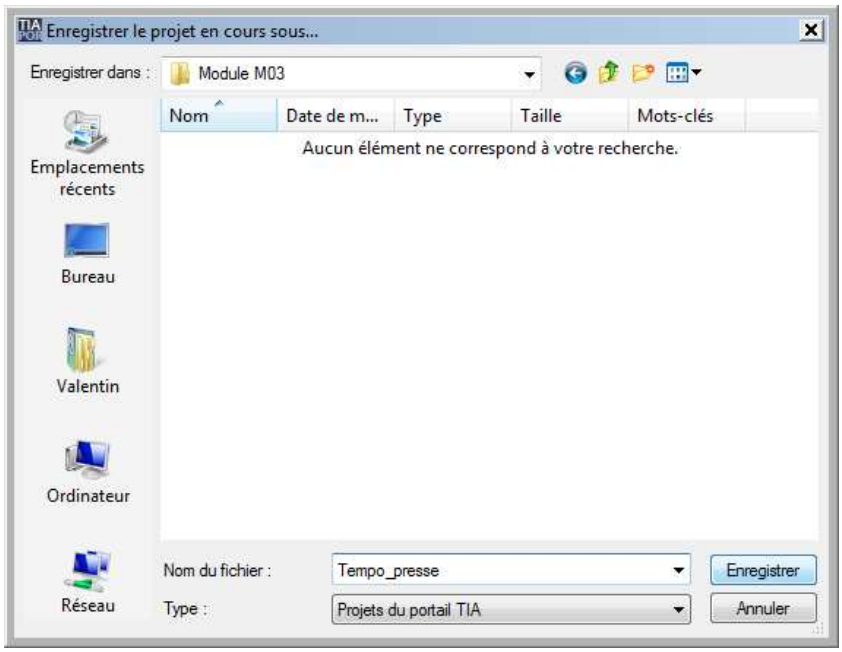
3. Maintenant, cliquez sur « **Mise en route** » et allez dans « **Ouvrir la vue du projet** ».
(vous pouvez également cliquer sur « **Vue du projet** » en bas à gauche de la fenêtre)



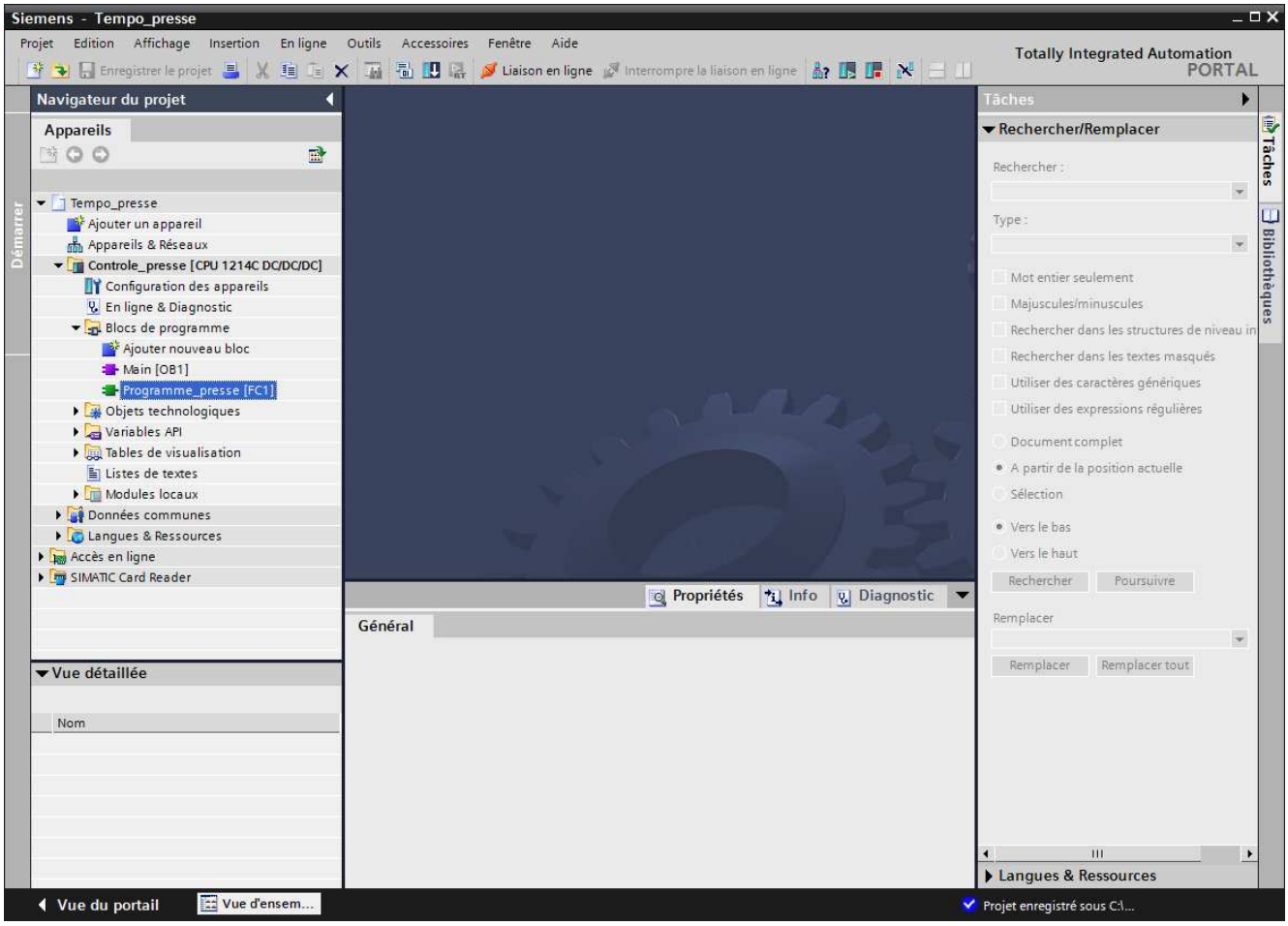
4. D'abord, sauvegardons le projet sous un nouveau nom. Dans la barre des menus, cliquez sur « **Projet** » puis « **Enregistrer sous** ».



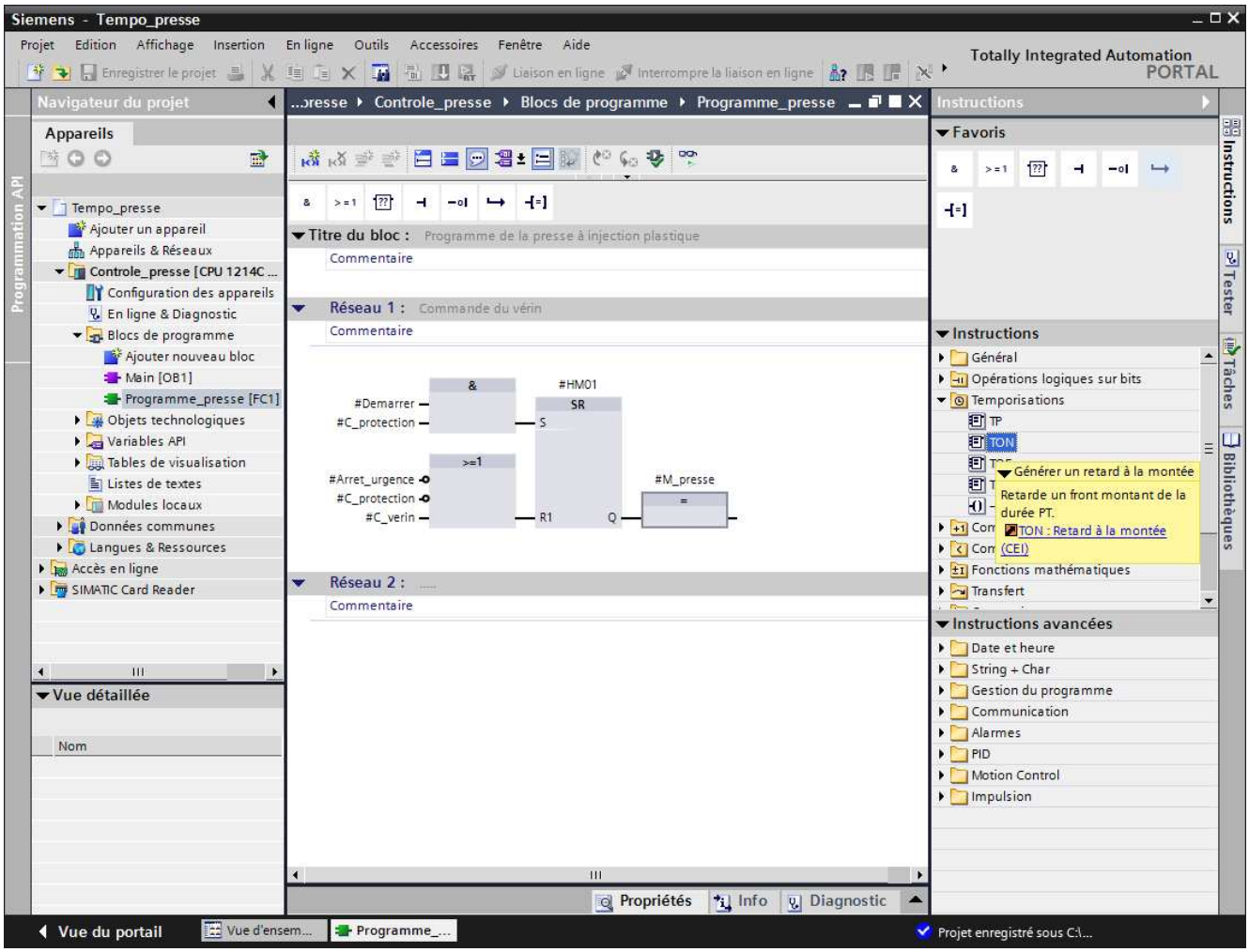
5. « **Enregistrer** » le projet sous le nom « Tempo_presse » par exemple.



6. Ouvrez le bloc « **Programme_presse [FC1]** » avec un double-clic pour commencer les modifications.



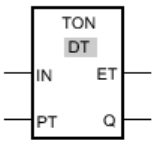
7. On peut dorénavant commencer à changer le programme.
   Pour générer un retard pour notre solution, on a besoin de l'opération « *Retard à la montée* **(TON)** ». Elle se situe dans la fenêtre de droite : « **Instructions** > **Temporisations** ».
   En laissant votre pointeur de souris sur un objet comme « TON », une description rapide s'affiche.



8. Si vous cliquez sur un objet pour le surligner et que vous appuyez sur « **F1** », une fenêtre d'aide s'affiche à droite et vous fournit des informations détaillées sur cet objet.

TON : Retard à la montée (CEI)

Représentation



Paramètre	Type de données	Zone de mémoire	Description
IN	BOOL	I, Q, M, D, L	Entrée de démarrage
PT	TIME	I, Q, M, D, L ou constante	Durée de laquelle le front montant est retardé à l'entrée IN.
ET	TIME	I, Q, M, D, L	Temps écoulé
Q	BOOL	I, Q, M, D, L	Sortie retardée de la durée PT.

Description

L'opération "Retard à la montée" permet de retarder un front montant du temps PT. L'opération "Retard à la montée" est uniquement exécutée lorsque le résultat logique (RLG) à l'entrée IN passe de "0" à "1" (front montant). Le temps PT commence au démarrage de l'opération. Lorsque la durée PT est écoulée, la sortie Q fournit l'état logique "1". La sortie Q reste mise à 1 tant que l'entrée de démarrage est à "1". Lorsque l'état logique passe de "1" à "0" à l'entrée de démarrage, la sortie Q est remise à 0. La fonction de temporisation est redémarrée lorsqu'un nouveau front positif est détecté à l'entrée de démarrage.

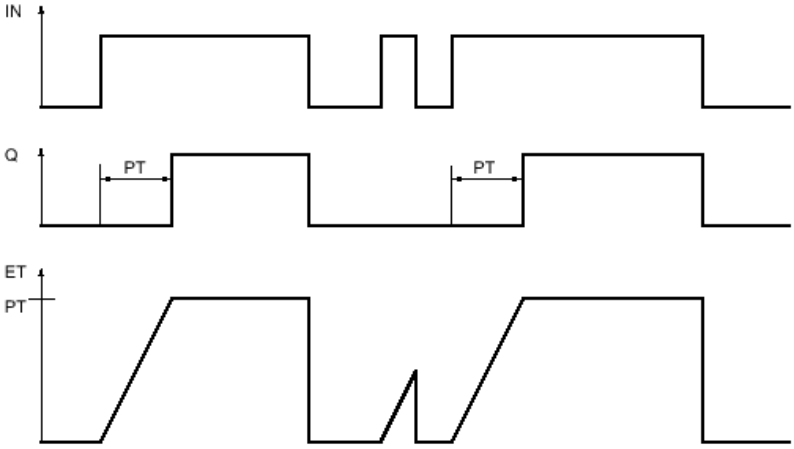
La sortie ET fournit le temps qui s'est écoulé depuis le dernier front montant à l'entrée IN. Ce temps commence à T#0s et se termine lorsque la valeur du temps PT est atteinte. Le temps écoulé peut être interrogé à la sortie ET tant que l'entrée IN fournit l'état logique "1". Lorsque l'entrée IN passe à "0", la sortie ET est remise à la valeur T#0.

L'insertion de l'opération "Retard à la montée" est accompagnée par la création d'un bloc de données d'instance dans lequel sont enregistrées les données de l'opération.

Emplacement

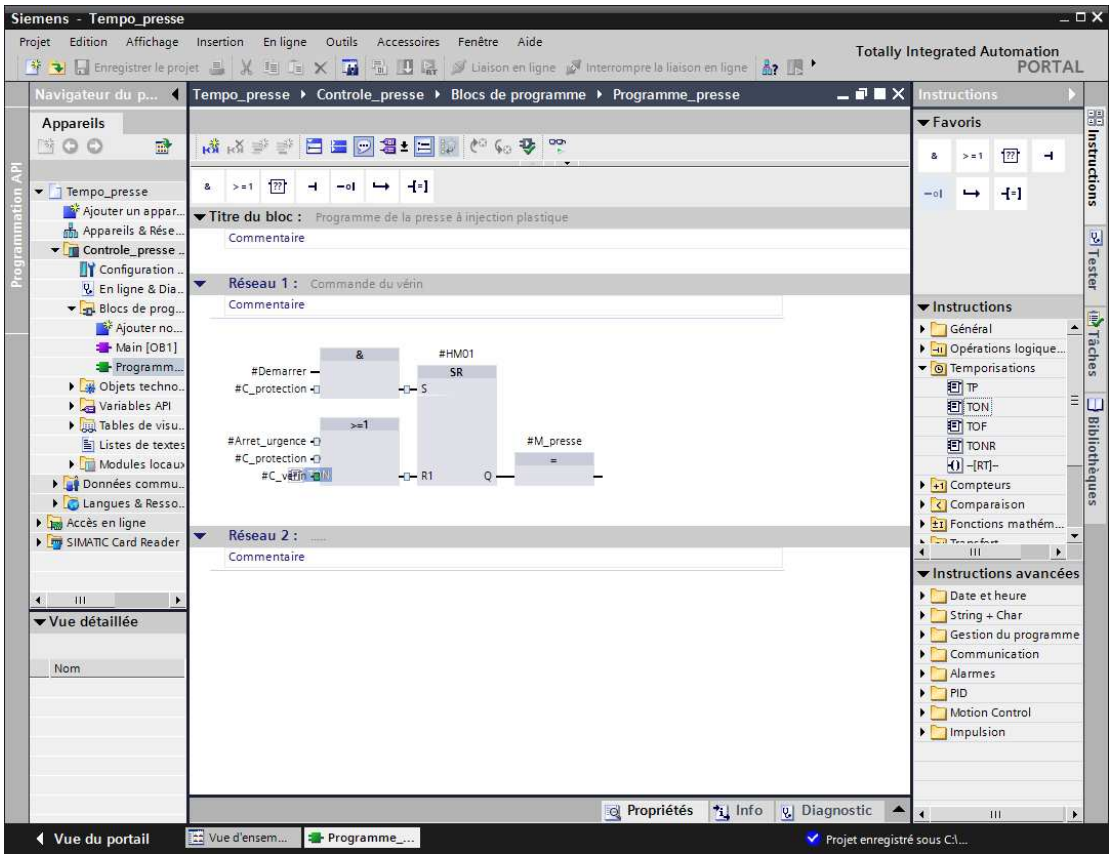
Pour l'évaluation du front, l'opération "Retard à la montée" requiert une opération amont. Elle peut être placée dans la chaîne opératoire ou à la fin de celle-ci.

Diagramme d'impulsions

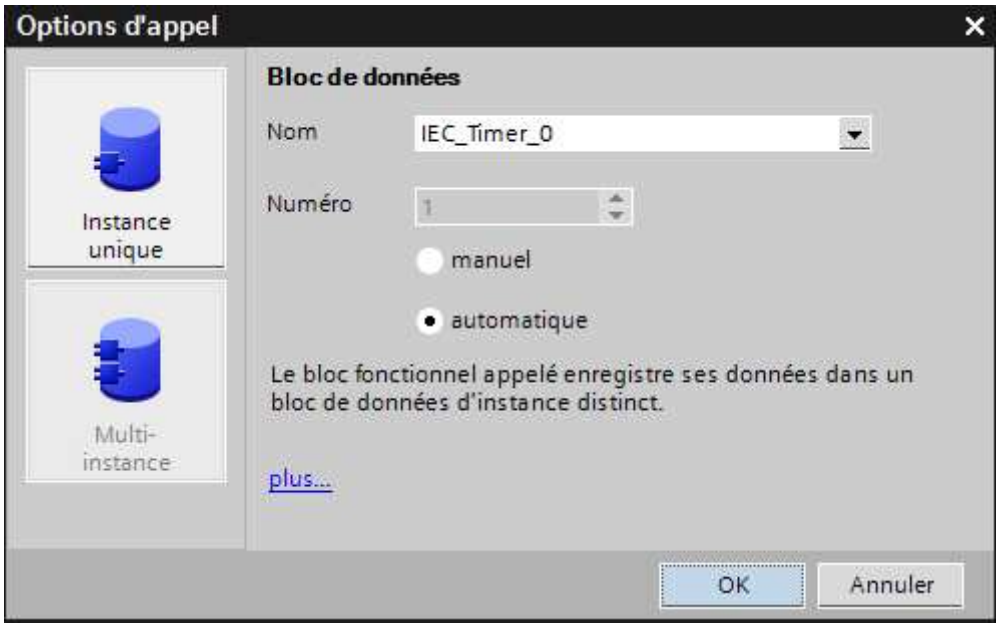


Indication : à partir d'ici, vous pouvez chercher vous-mêmes des informations sur les temporisations.

9. Ensuite, glissez-déposez le bloc temporisation « **TON** » sur la 3ème entrée du bloc OU, derrière « *#C_verin* ».



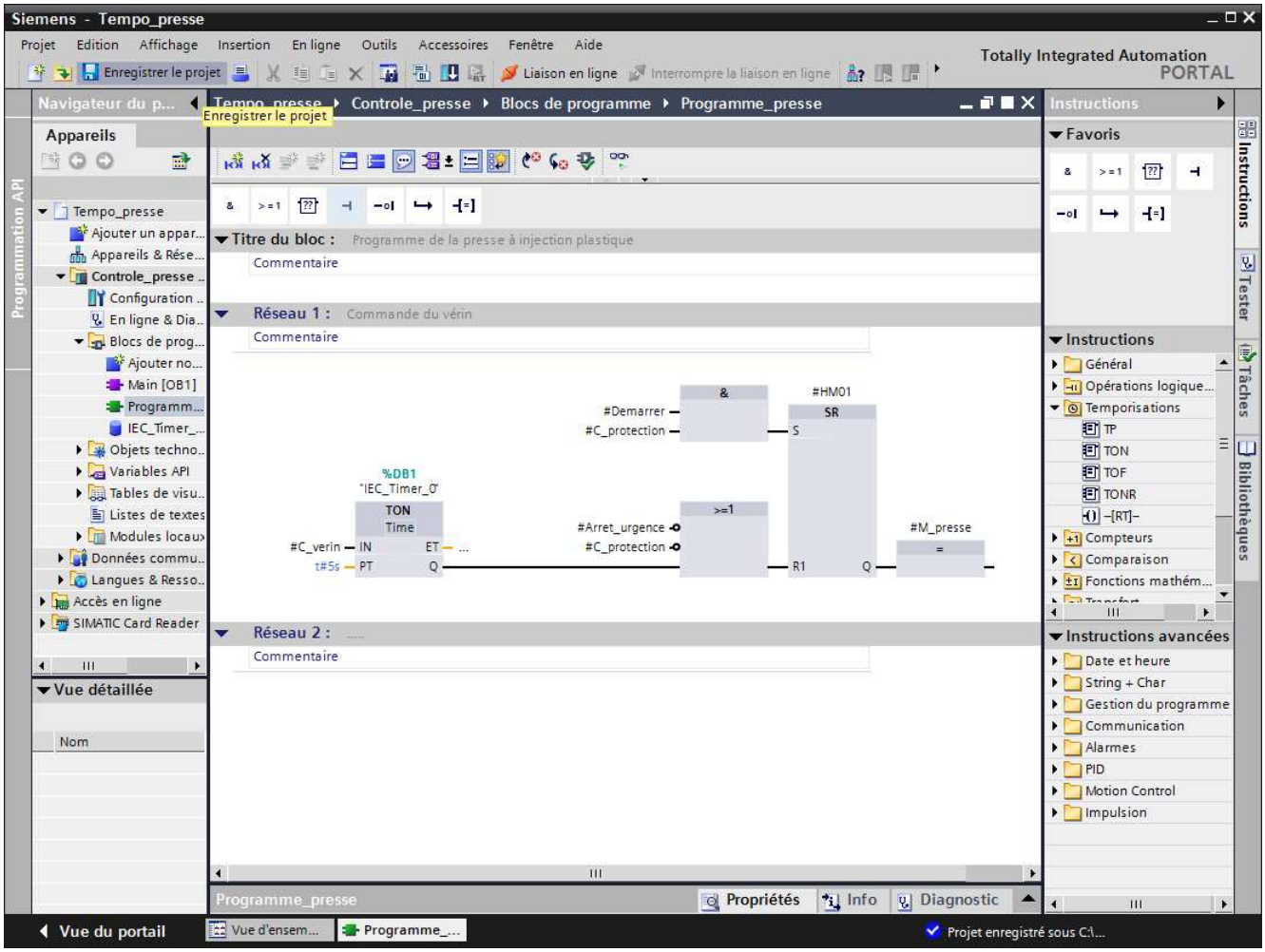
10. La fonction de temporisation requiert une mémoire. Elle lui est fournie en créant un nouveau bloc de données d'instance en **instance unique**. Cliquez simplement sur « **OK** ».



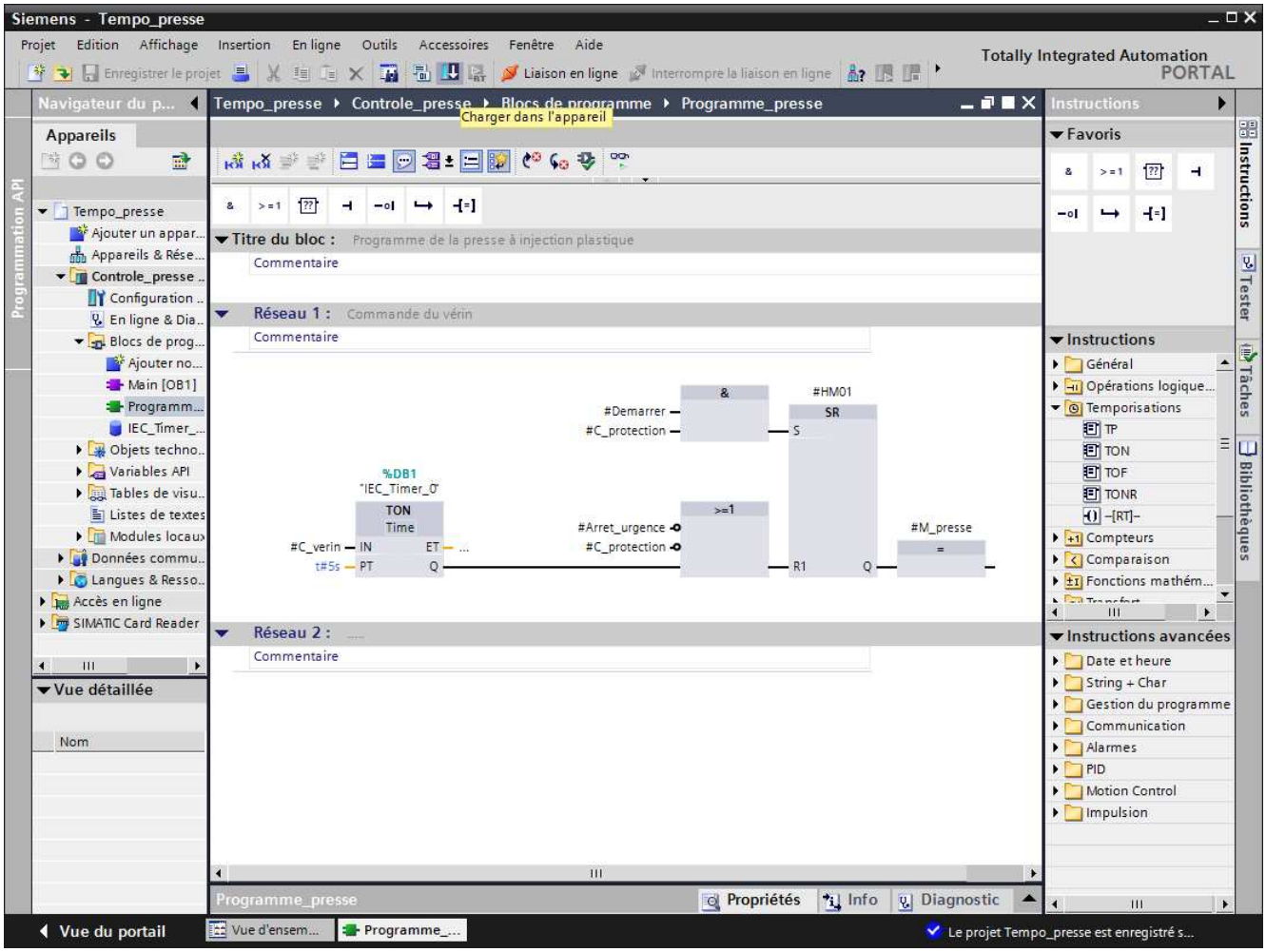
Indication :
Les multi-instances ne peuvent être utilisées que pour la programmation d'un bloc fonctionnel. Ceci sera illustré dans la suite avec le compteur CEI.

11. Entrez « **t#5s** » pour l'entrée « **PT** » du « **TON** » pour régler un retard de 5s pour l'entrée « *#C_verin* ».

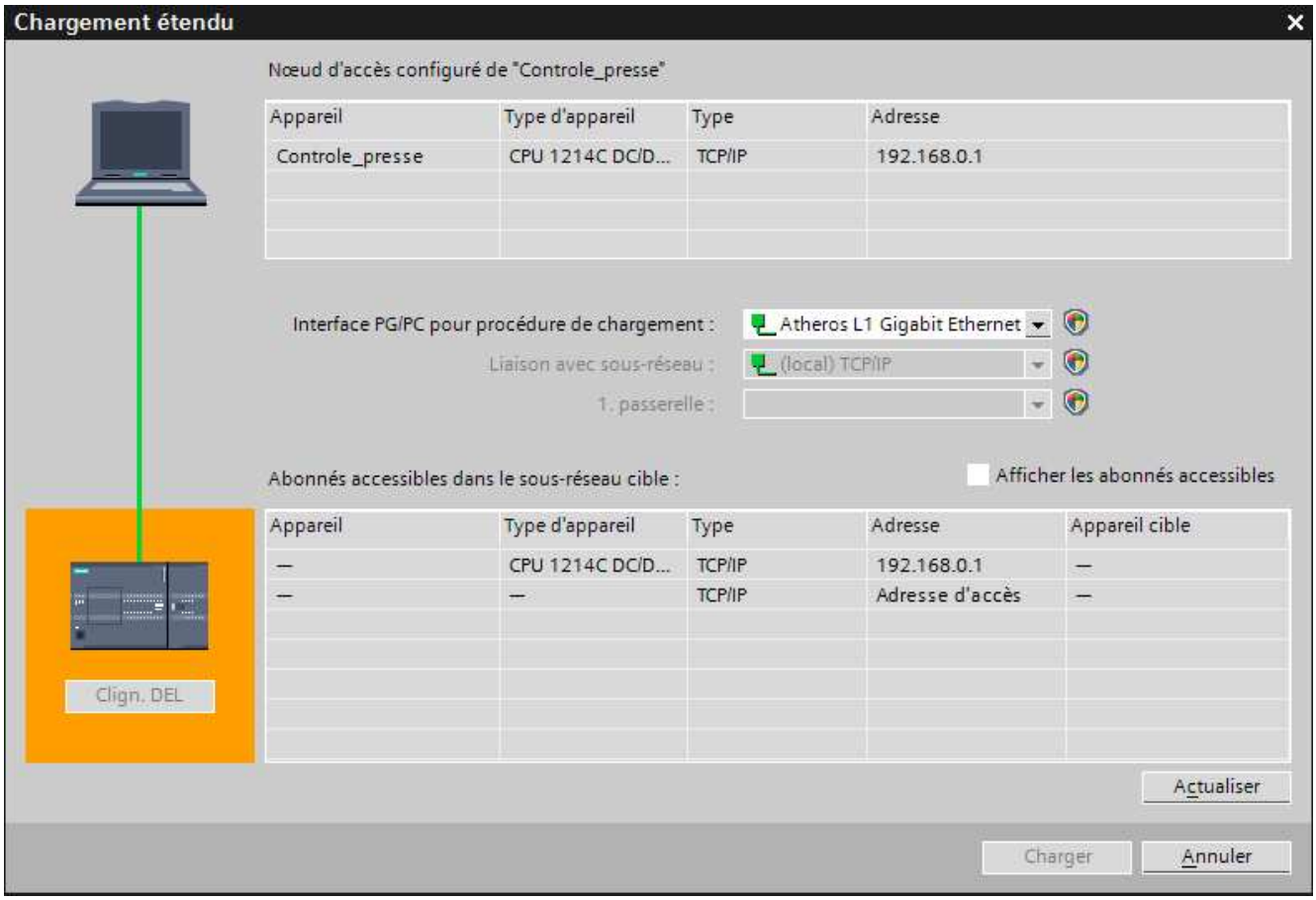
    Sauvegardez ensuite votre projet en cliquant sur Enregistrer le projet.



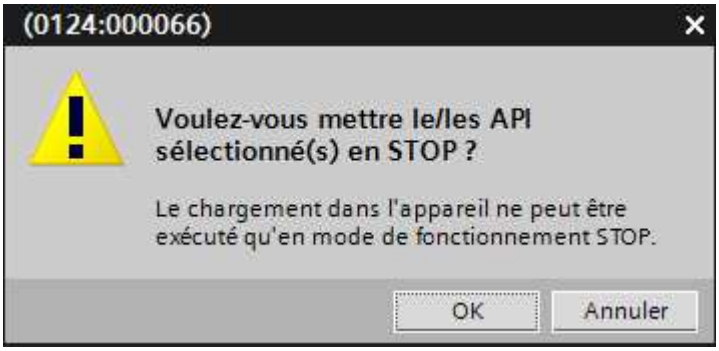
12. Pour charger votre programme entier dans la CPU, surlignez d'abord « **Contrôle_presse [CPU 1214C DC/DC/DC]** » en cliquant une fois dessus. Cliquez ensuite sur le symbole « **Charger dans l'appareil** ».



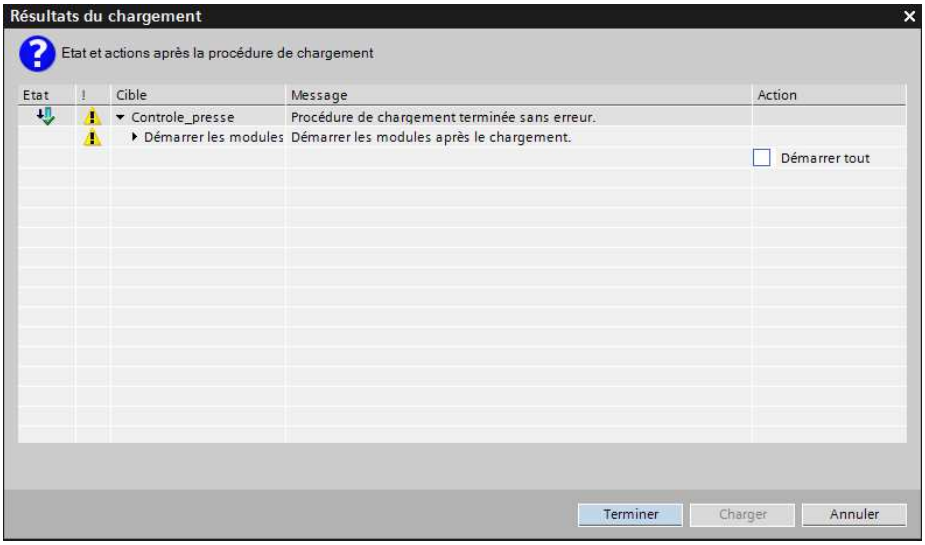
13. Dans le cas où vous auriez oublié de paramétrer l'interface PG/PC auparavant, une fenêtre où il est encore possible de le faire s'ouvre.
    Sélectionnez la CPU et cliquez sur « **Charger** ».



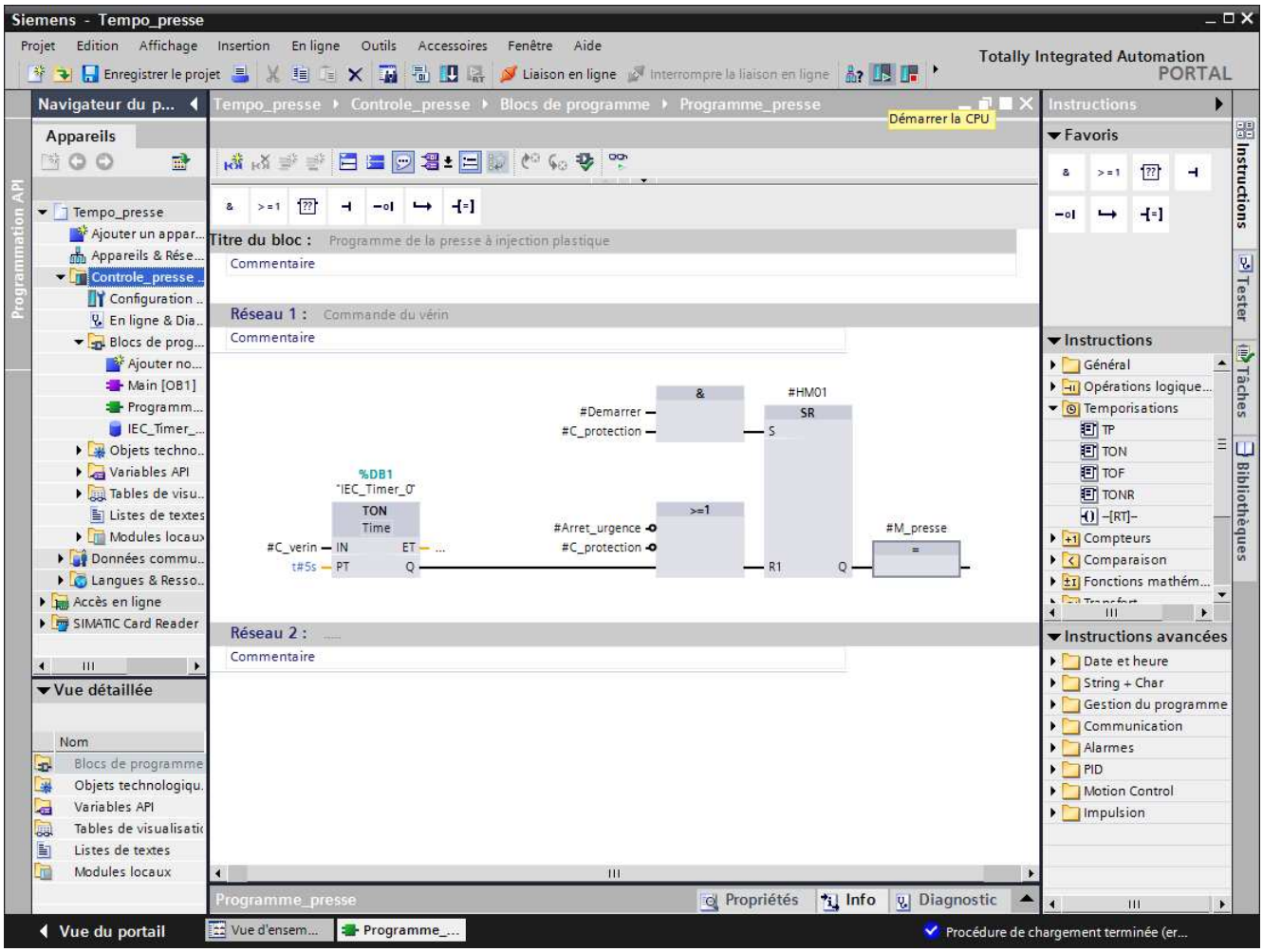
14. Si la CPU était en mode « **RUN** », un message s'affichera et vous demandera si vous voulez mettre la CPU en mode « **STOP** ». Confirmez le choix en cliquant sur « **OK** ».



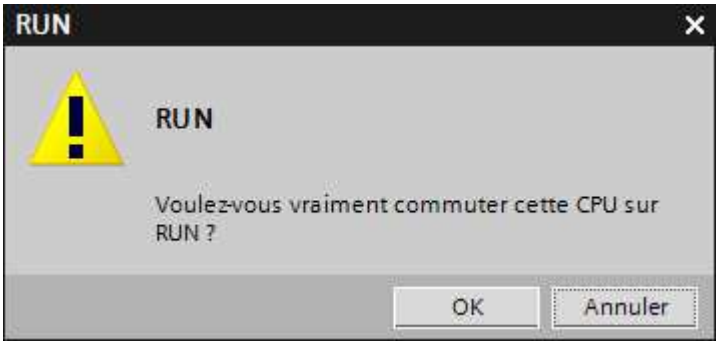
15. Une nouvelle fenêtre apparaît. Cliquez sur « **Charger** » une fois de plus. Pendant le chargement, l'état de progression est affiché dans la fenêtre. Si le chargement s'est correctement déroulé, le résultat s'affiche dans une nouvelle fenêtre. Cliquez finalement sur « **Terminer** ».



16. Ensuite, démarrez la CPU en cliquant sur le symbole « **Démarrer la CPU** ».

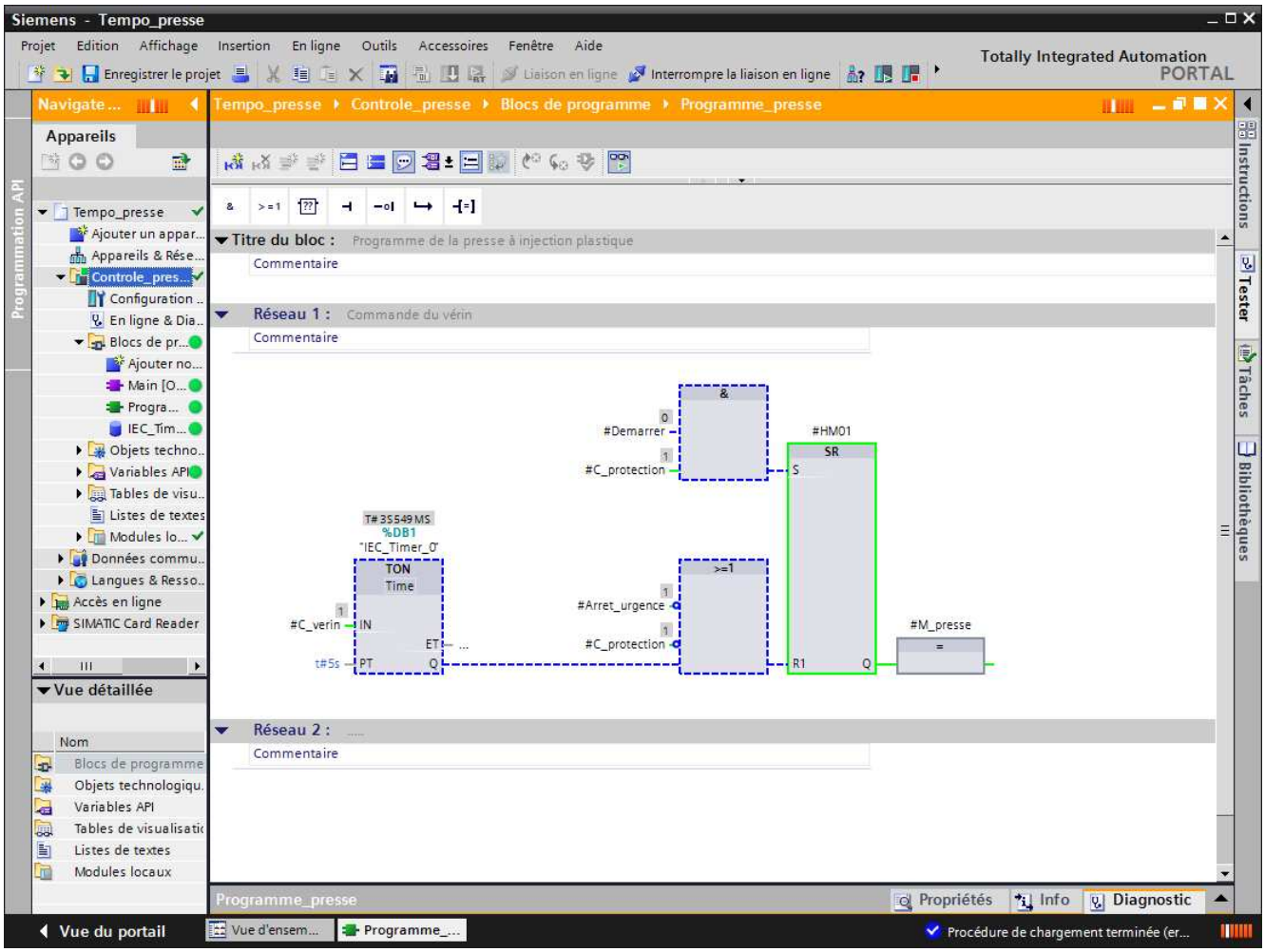


17. Confirmez le fait que vous vouliez vraiment commuter la CPU sur *RUN* en cliquant sur « **OK** ».



18. En cliquant sur l'icône « **Activer/désactiver visualisation du programme** », il est possible de surveiller l'état de la temporisation et le temps écoulé sur le bloc « *TON* » 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.



3 Exemple d'application : Commande d'un convoyeur avec Compteur et Multi-Instances

Quand les blocs sont créés, s'ils doivent travailler dans un programme quelconque qu'on pourrait appeler « boîte noire », ils doivent être programmés en **utilisant des variables**. Dans ce cas, la règle suivante s'applique : **dans ces blocs, seules les entrées/sorties à adresse non-absolue, les mnémoniques, etc… doivent être utilisées**. Dans ces blocs, seules les variables et les constantes sont utilisées.

Si des blocs fonctionnels secondaires (comme les tempos ou les compteurs) sont appelés à partir d'un bloc pouvant être utilisé une multitude de fois, il ne faut pas leur fournir leur propre bloc de données. La mémoire requise est fournie en multi-instance à l'intérieur du DB d'instance assigné au bloc fonctionnel effectuant l'appel.

Dans l'exemple ci-dessous, on ajoute un compteur de bouteilles au bloc fonctionnel contenant déjà la commande du convoyeur en fonction du mode de fonctionnement choisi.

Avec ce convoyeur, 20 bouteilles sont acheminés vers une caisse. Quand la caisse est pleine, le convoyeur doit s'arrêter et la caisse doit être changée.

Avec le bouton « S1 », on peut sélectionner le mode de fonctionnement « Manuel », et avec le bouton « S2 », on peut sélectionner le mode « Automatique ».
En mode « Manuel », le moteur est alimenté tant qu'on appuie sur le bouton « S3 » et que le bouton « S4 » n'est pas activé.
En mode « Automatique », le moteur du convoyeur est allumé avec le bouton « S3 » et éteint avec le bouton « S4 ».
Il y a maintenant en plus un capteur 'B0' qui compte le nombre de bouteilles dans la caisse. Quand 20 bouteilles sont comptées, le convoyeur s'arrête.
Quand une nouvelle caisse est amenée, il faut le confirmer au moyen du bouton 'S5'.

Tableau d'affectations

Adresses	Variables	Commentaires
%I 0.0	S1	Bouton mode manuel, S1 NO
%I 0.1	S2	Bouton mode automatique, S2 NO
%I 0.2	S3	Bouton marche, S3 NO
%I 0.3	S4	Bouton arrêt, S4 NF
%I 0.6	S5	Bouton reset compteur/nouvelle caisse, S5 NO
%I 0.7	B0	Capteur pour compter les bouteilles, B0 NO
%Q 0.2	M01	Moteur du convoyeur M01

3.1 Programmation du convoyeur avec le SIMATIC S7-1200

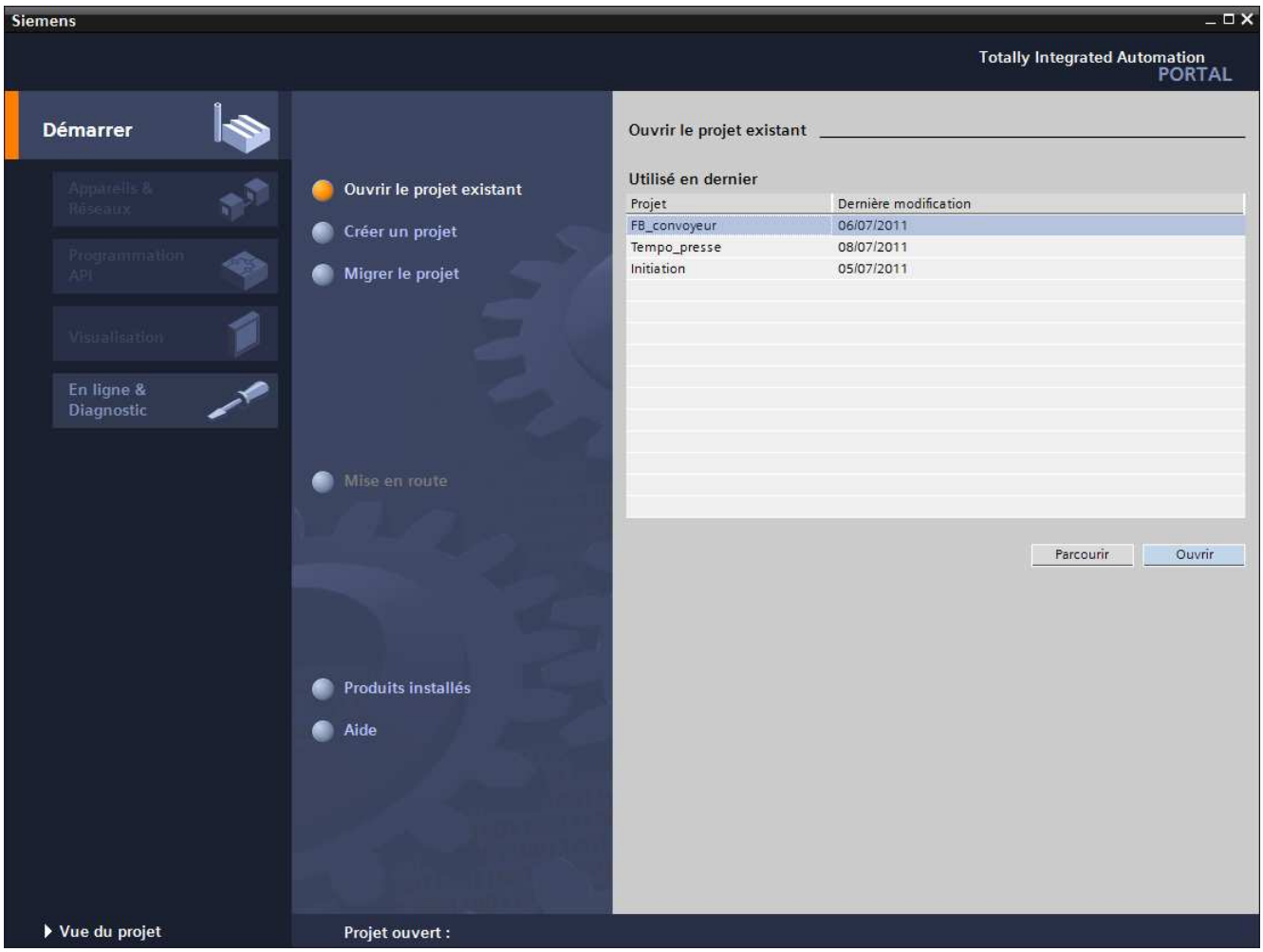
La gestion du projet et sa programmation se font grâce au logiciel « **Totally Integrated Automation Portal** ».

Là, sous une même interface, les éléments tels que le système de contrôle, 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.

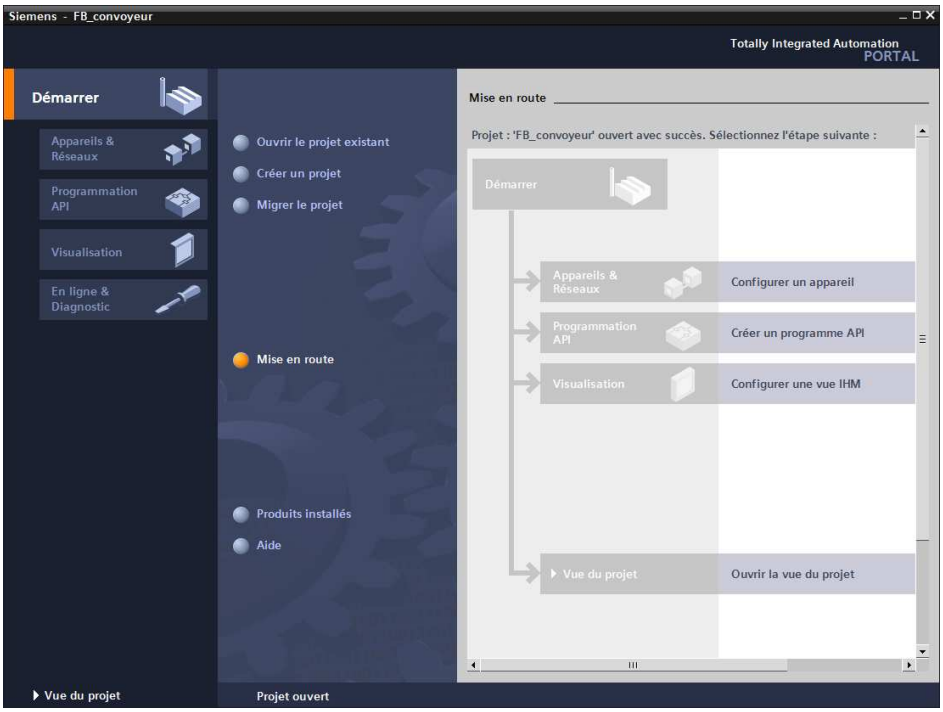
Dans les étapes suivantes, nous allons ouvrir un projet pour le SIMATIC S7-1200, l'enregistrer sous un nouveau nom, et le modifier pour qu'il réponde aux nouvelles exigences.

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

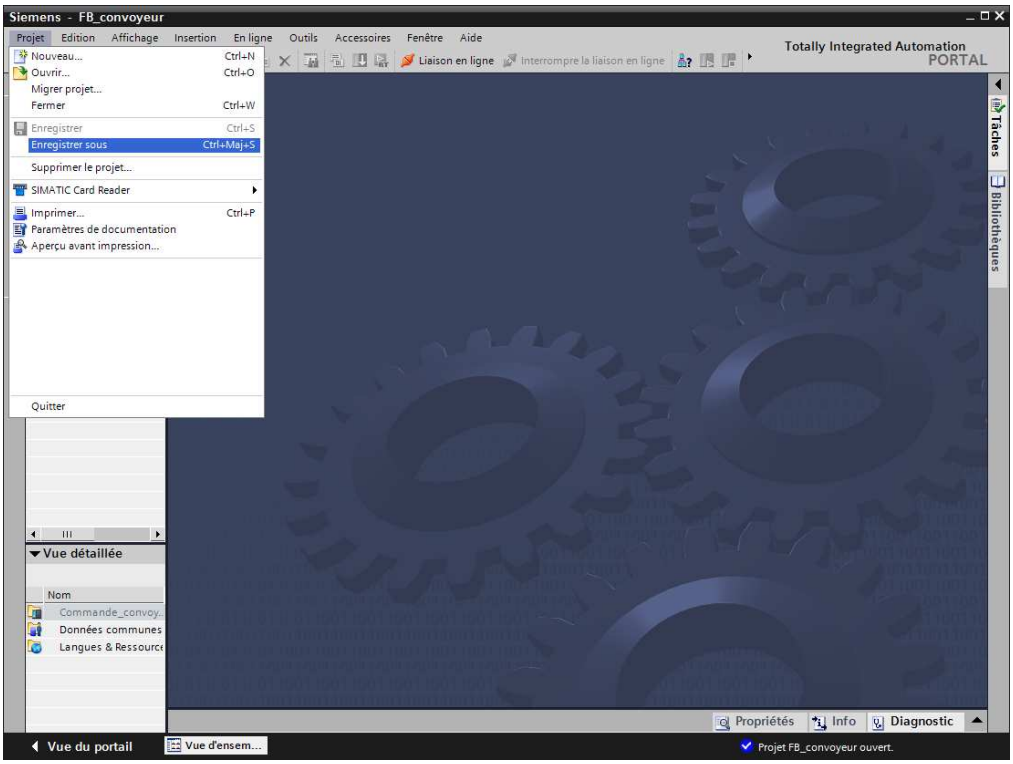
2. Nous allons maintenant ouvrir le projet « *FB_convoyeur* » du module M2 dans la vue du portail. Ce projet servira de base pour le programme.



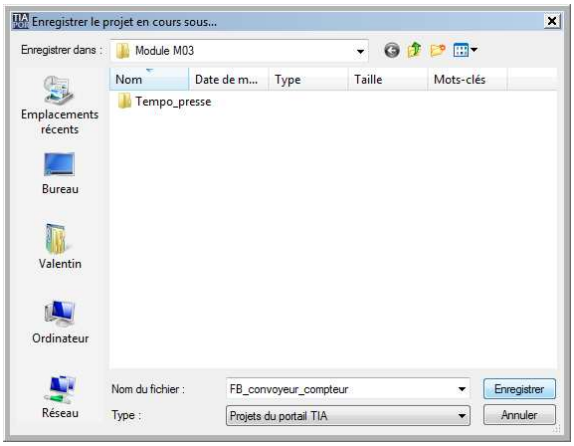
3. « **Mise en route** » est recommandée pour le début de la création du projet.
   Cliquez sur « **Ouvrir la vue projet** ».



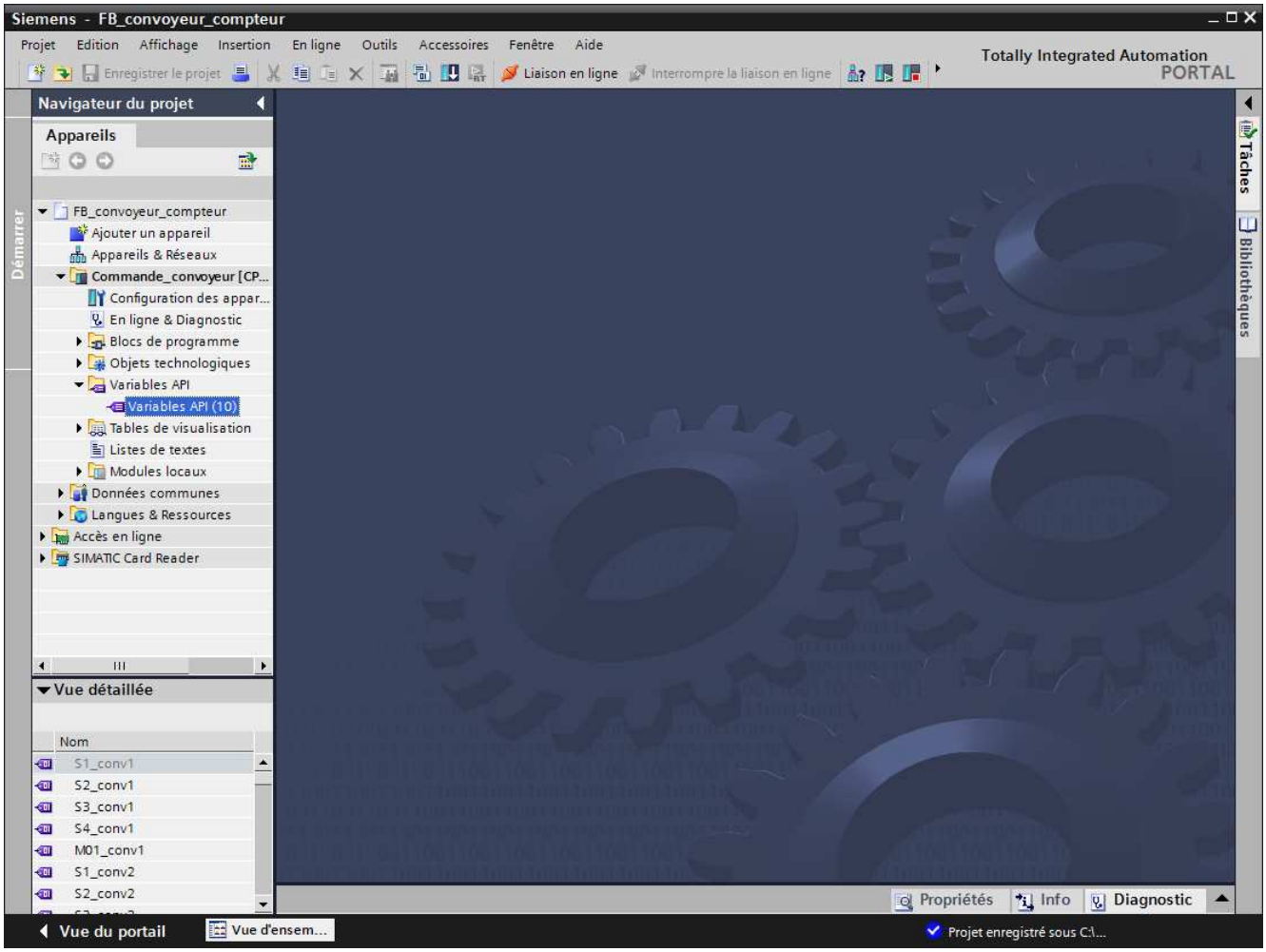
4. D'abord, sauvegardons le projet sous un nouveau nom. Dans la barre des menus, cliquez sur « **Projet** » puis « **Enregistrer sous** ».



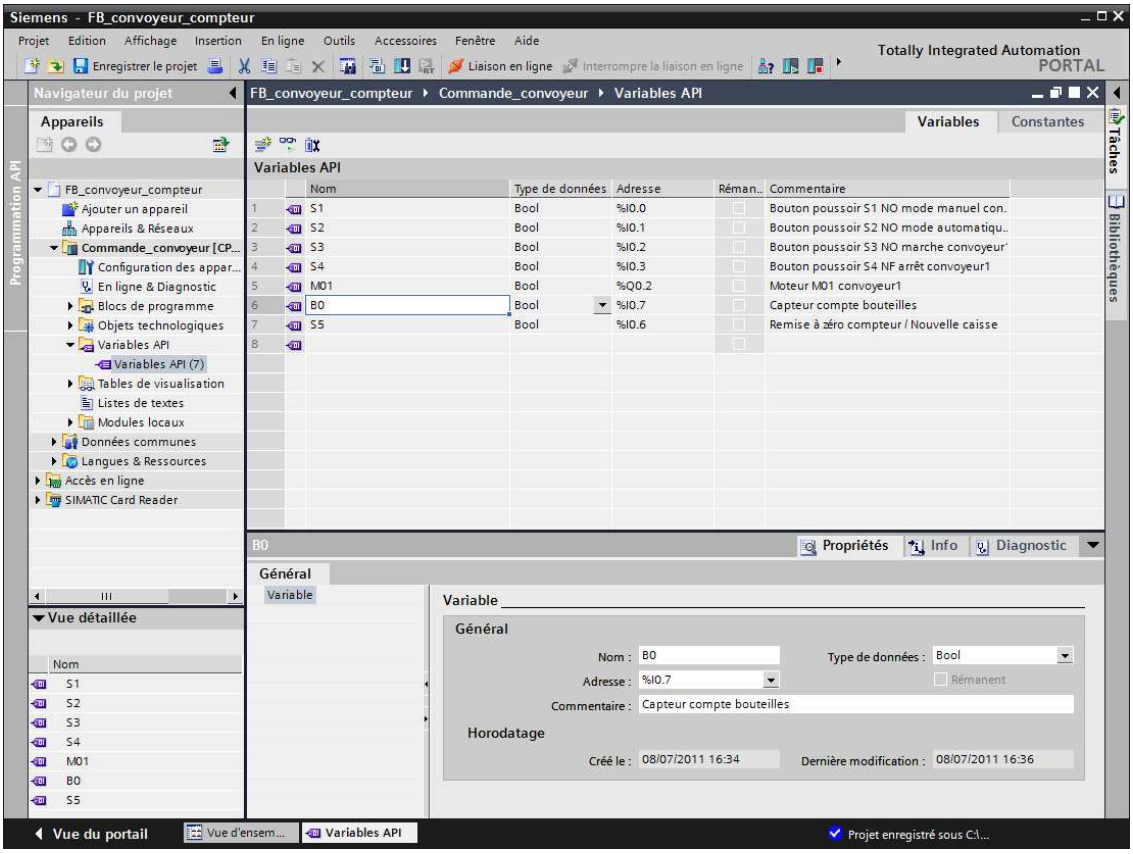
5. « **Enregistrer** » le projet sous le nom « **FB_convoyeur_compteur** » par exemple.



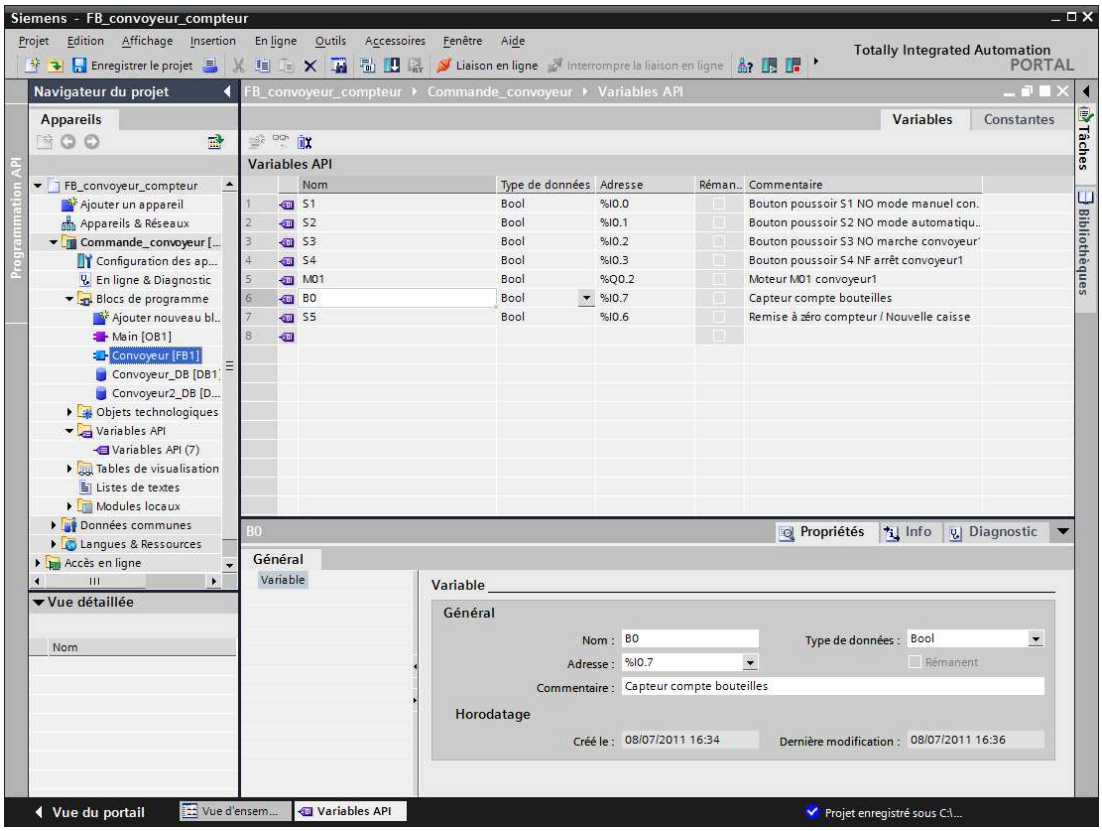
6. Pour établir de nouvelles variables globales, ouvrez d'un double-clic « **Variables API (5)** » dans « **Commande_convoyeur [CPU 1214C DC/DC/DC]** > **Variables API** ».



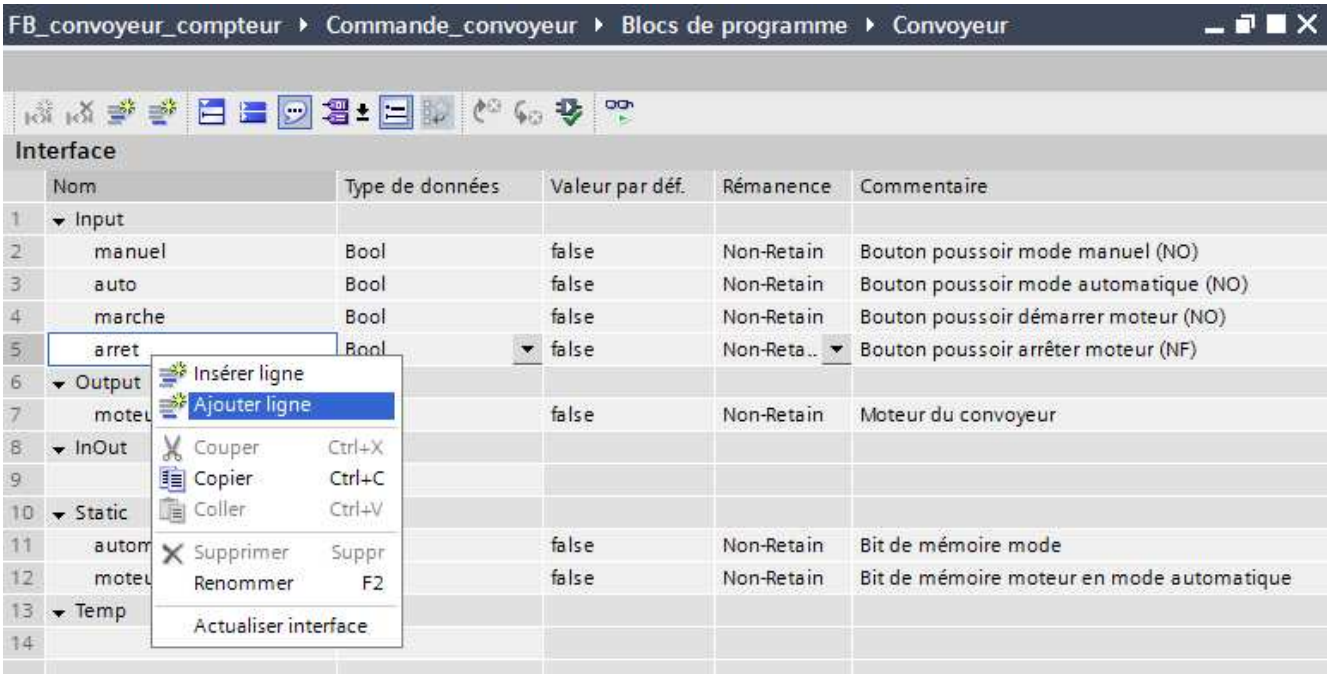
7. Ensuite, rajoutez les deux variables globales « **B0** » et « **S5** », comme indiqué ci-dessous.



8. Pour effectuer les changements, ouvrez le bloc « **Convoyeur [FB1]** » avec un double-clic.



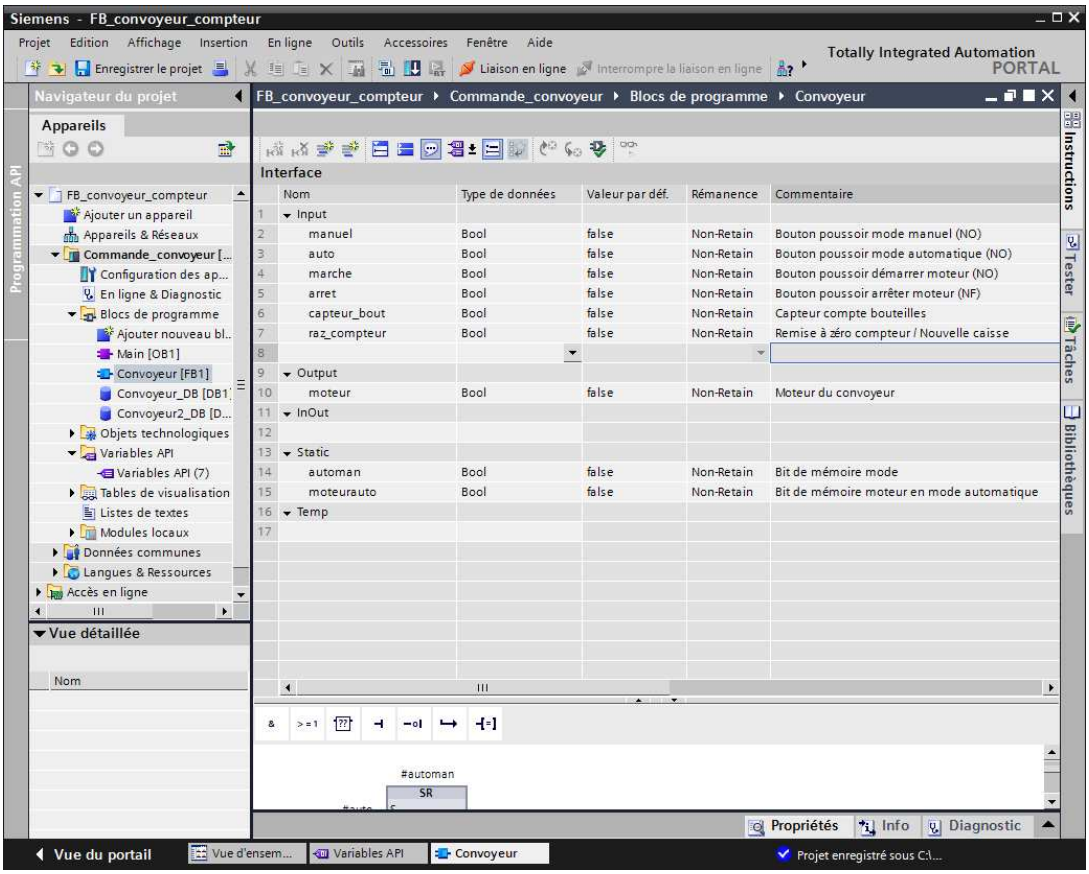
9. Dans un premier temps, ajoutez 2 lignes pour les variables d'entrée dans l'interface, à l'aide d'un clic-droit : « **Ajouter ligne** ».



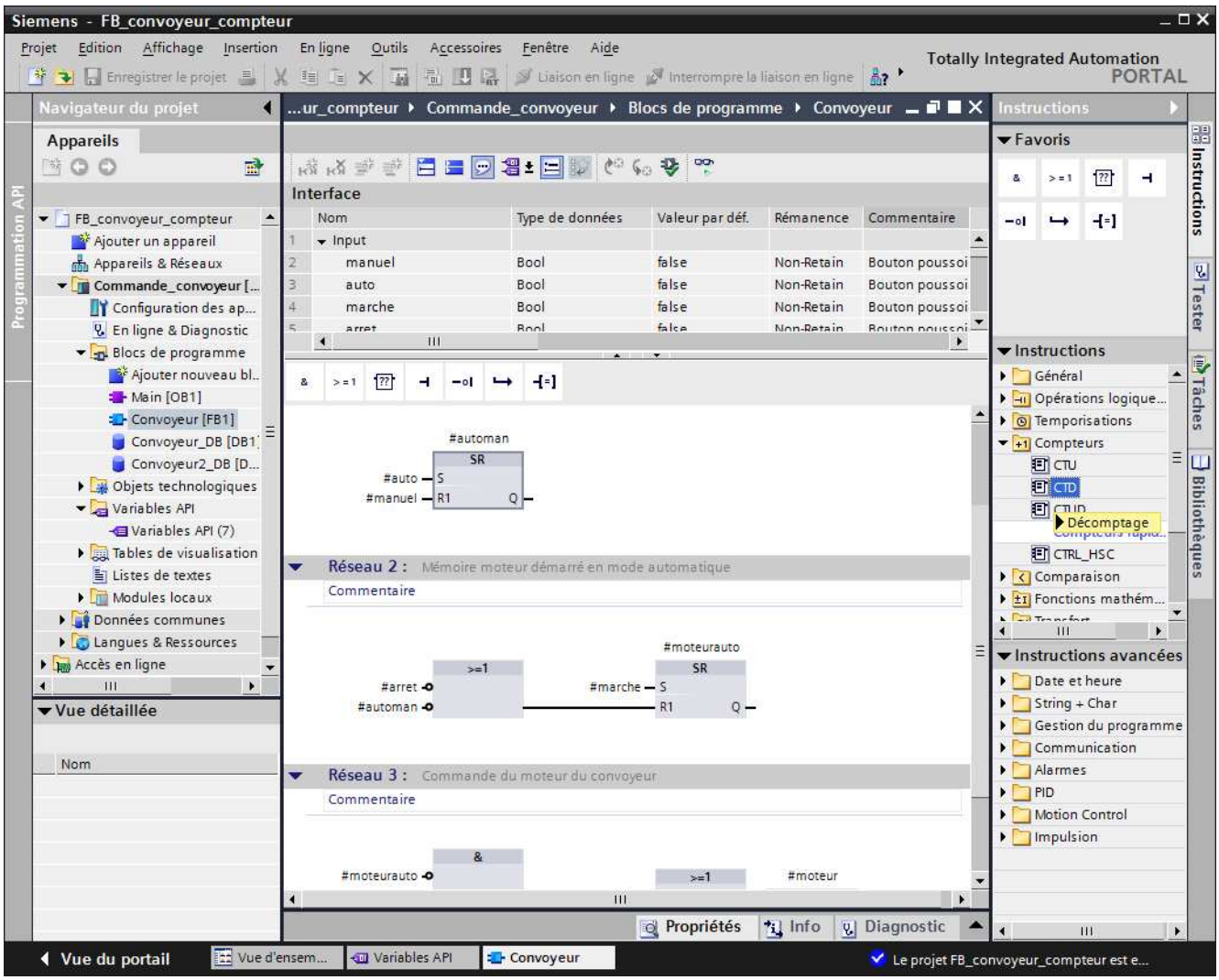
10. On peut maintenant rajouter les deux variables locales :

Input :

capteur_bout	Capteur permettant de compter les bouteilles
raz_compteur	Remise à zéro compteur / Nouvelle caisse



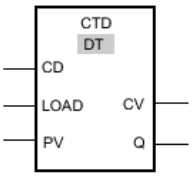
11. On peut maintenant commencer à modifier le programme.
    Le compteur dont on a besoin pour notre solution est un décompteur « **CTD** ». On le trouve dans la fenêtre de droite, « **Instructions** > **Compteurs** ».
    En laissant votre pointeur de souris sur un objet comme « CTD », une description rapide s'affiche.



12. Si vous cliquez sur un objet pour le surligner et que vous appuyez sur « **F1** », une fenêtre d'aide s'affiche à droite et vous fournit des informations détaillées sur cet objet.

CTD : Décomptage (CEI)

Représentation



Paramètre Anglais	Type de données	Zone de mémoire	Description
CD	BOOL	I, Q, M, D, L	Entrée du compteur
LOAD	BOOL	I, Q, M, D, L	Entrée de chargement
PV	SINT, UINT, DINT, USINT, UINT, UDINT	I, Q, M, D, L ou constante	Valeur prédéfinie du compteur
Q	BOOL	I, Q, M, D, L	Etat du compteur
CV	SINT, UINT, DINT, USINT, UINT, UDINT	I, Q, M, D, L	Valeur actuelle du compteur

Dans la liste déroulante "DT", vous pouvez sélectionner le type de données de l'opération.

Description

L'opération "Décomptage" permet de décrémenter la valeur à la sortie CV. Lorsque l'état logique passe de "0" à "1" à l'entrée CD (front positif), l'opération est exécutée et la valeur actuelle du compteur est décrémentée de un à la sortie CV. Lors de la première exécution de l'opération, la valeur actuelle du compteur est mise à zéro à la sortie CV. La valeur du compteur continue à être décrémentée à chaque détection de front positif, jusqu'à ce que la valeur limite inférieure du type de données spécifié soit atteinte. Lorsque la valeur limite inférieure est atteinte, l'état logique à l'entrée CD n'a plus d'influence sur l'opération.

L'état du compteur peut être interrogé à la sortie Q. Lorsque la valeur actuelle du compteur est inférieure ou égale à zéro, la sortie Q est mise à l'état logique "1". Dans tous les autres cas, l'état logique à la sortie Q est égal à "0".

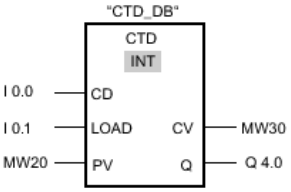
La valeur à la sortie CV prend la valeur du paramètre PV lorsque l'état logique à l'entrée LOAD passe à "1". Tant que l'état logique est égal à "1" à l'entrée LOAD, l'état logique à l'entrée CD n'a pas d'effet sur l'opération.

L'insertion de l'opération "Décomptage" est accompagnée par la création d'un bloc de données d'instance dans lequel sont enregistrées les données de l'opération.

Emplacement

Pour l'évaluation du front, l'opération "Décomptage" requiert une opération amont. Elle peut être placée dans la chaîne opératoire ou à la fin de celle-ci.

Exemple



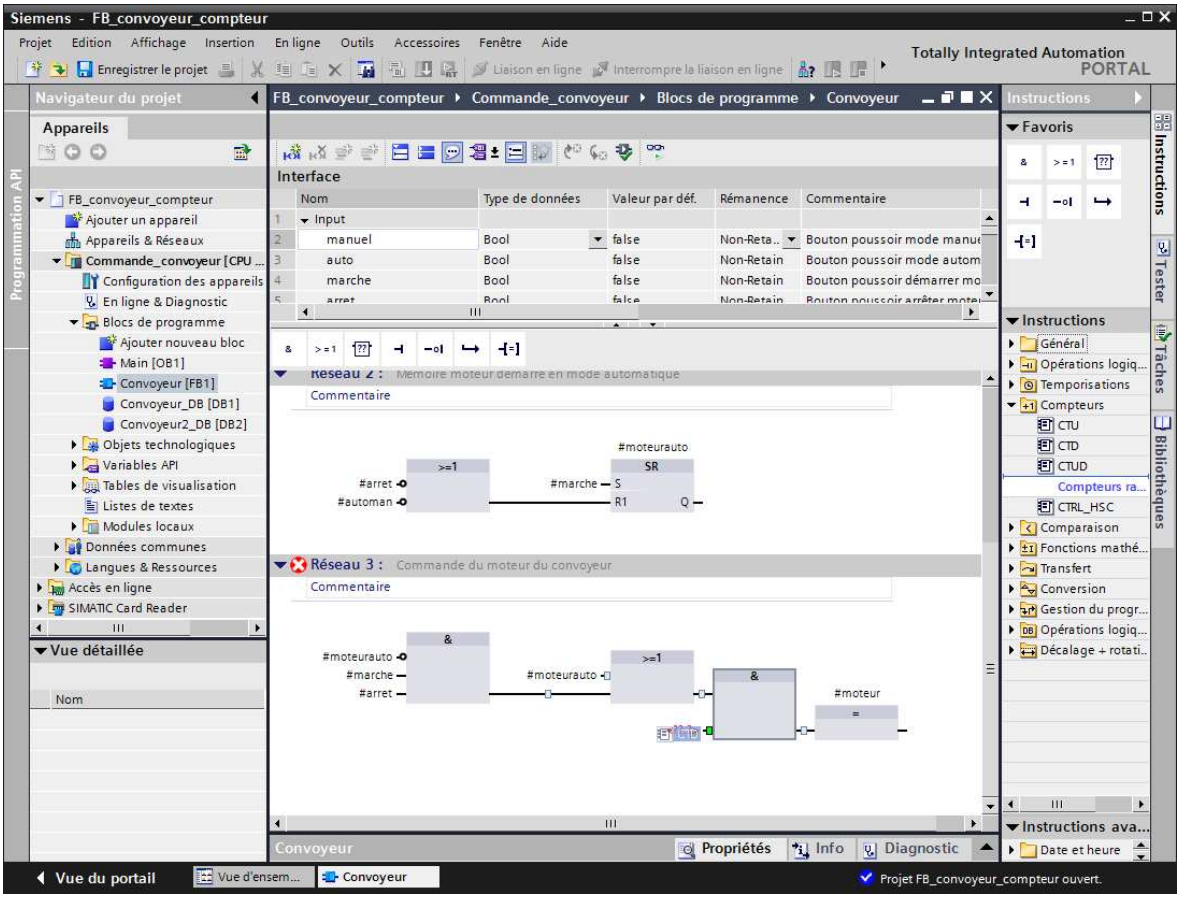
Lorsque l'état logique passe de "0" à "1" à l'entrée I 0.0, l'opération "décomptage" est exécutée et la valeur actuelle est décrémentée de un à l'entrée MW30. La valeur du compteur est décrémentée à chaque autre front positif, jusqu'à ce que la valeur limite inférieure du type de données spécifié (-32 768) soit atteinte.

La sortie Q 4.0 fournit l'état logique "1" tant que la valeur actuelle du compteur est inférieure ou égale à zéro. Dans tous les autres cas, la sortie Q 4.0 fournit l'état logique "0".

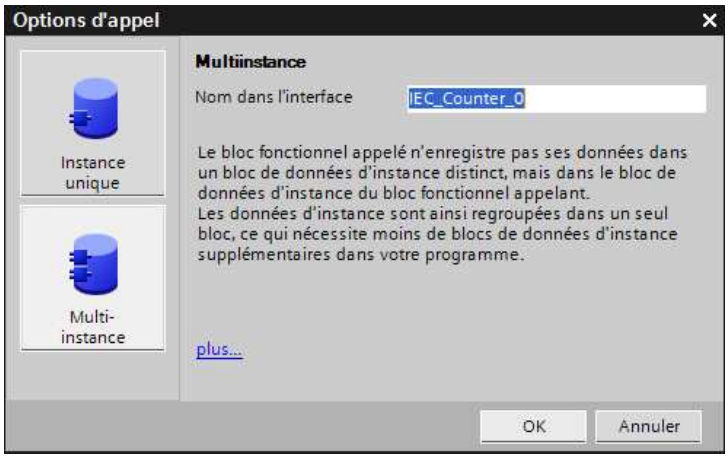
Indication :

À partir d'ici, vous pouvez chercher vous-mêmes des informations sur les compteurs.

13. Insérez dans un premier temps un bloc **ET** entre le OU et l'affectation, puis déposez le décompteur « **CTD** » à la seconde entrée du bloc **ET**.



14. La fonction compteur a besoin d'une mémoire. Elle lui est fournie dans un bloc de données d'instance en **multi-instance**, sans générer un nouveau DB d'instance. Choisissez donc « **Multi-instance** » et cliquez sur « **OK** ».

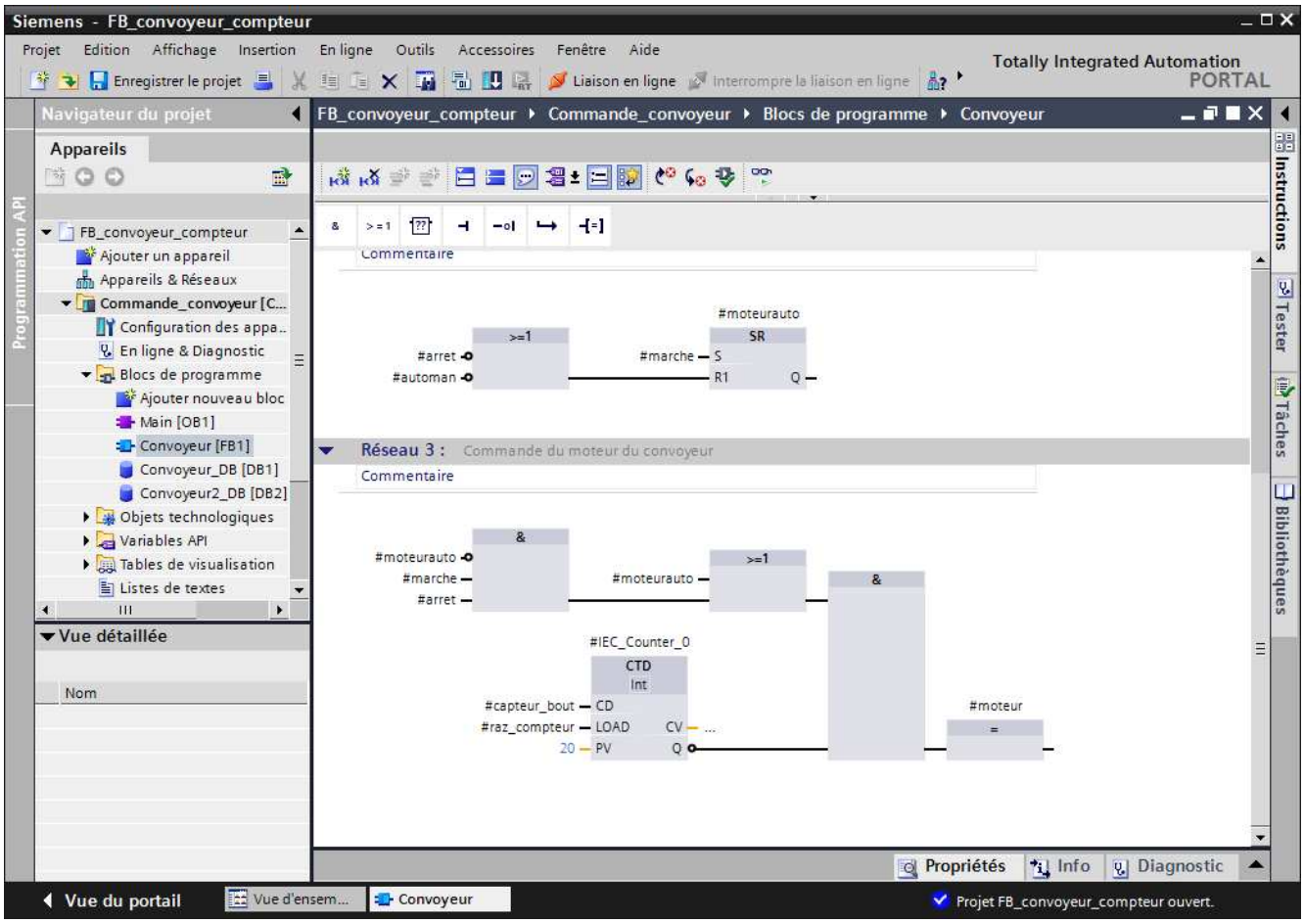


Indication :

Une multi-instance ne peut être utilisée que lors de la programmation d'un bloc fonctionnel.

15. Pour l'entrée **PV** du décompteur CTD, rentrez la valeur « **20** » qui correspond aux 20 bouteilles, pour l'entrée **CD** choisissez la variable « **#capteur_bout** », et pour l'entrée **LOAD** choisissez la variable « **raz_compteur** ». Ensuite, insérez une négation à la deuxième entrée du bloc ET.

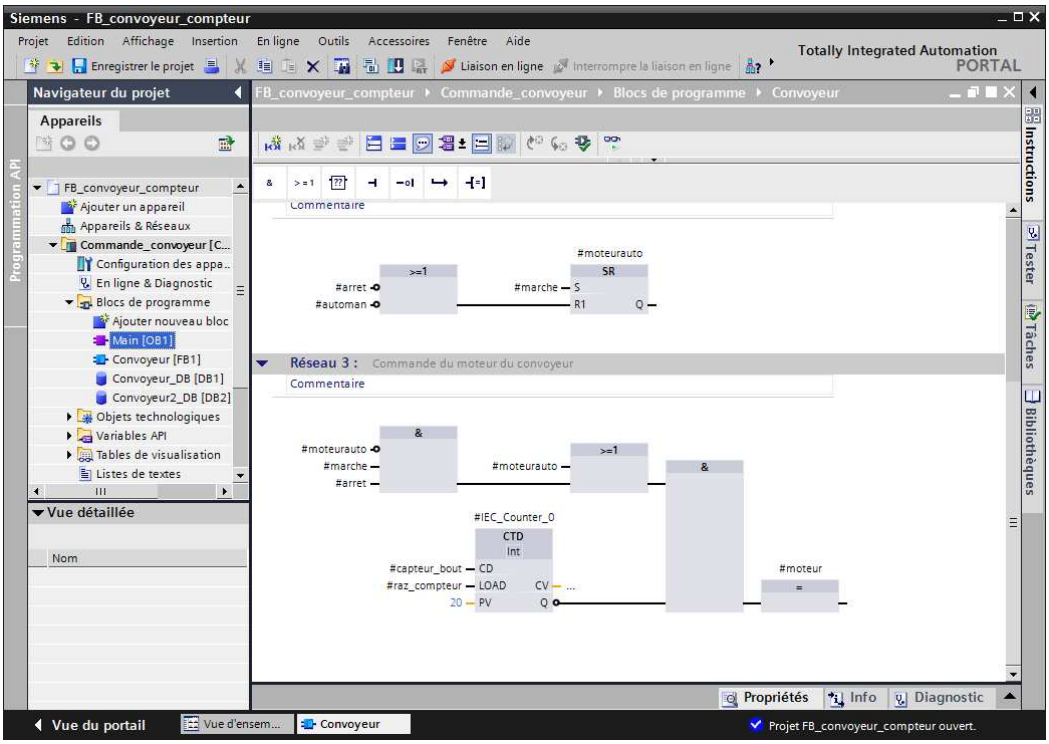
    Cliquez ensuite sur Enregistrer le projet pour sauvegarder votre projet.



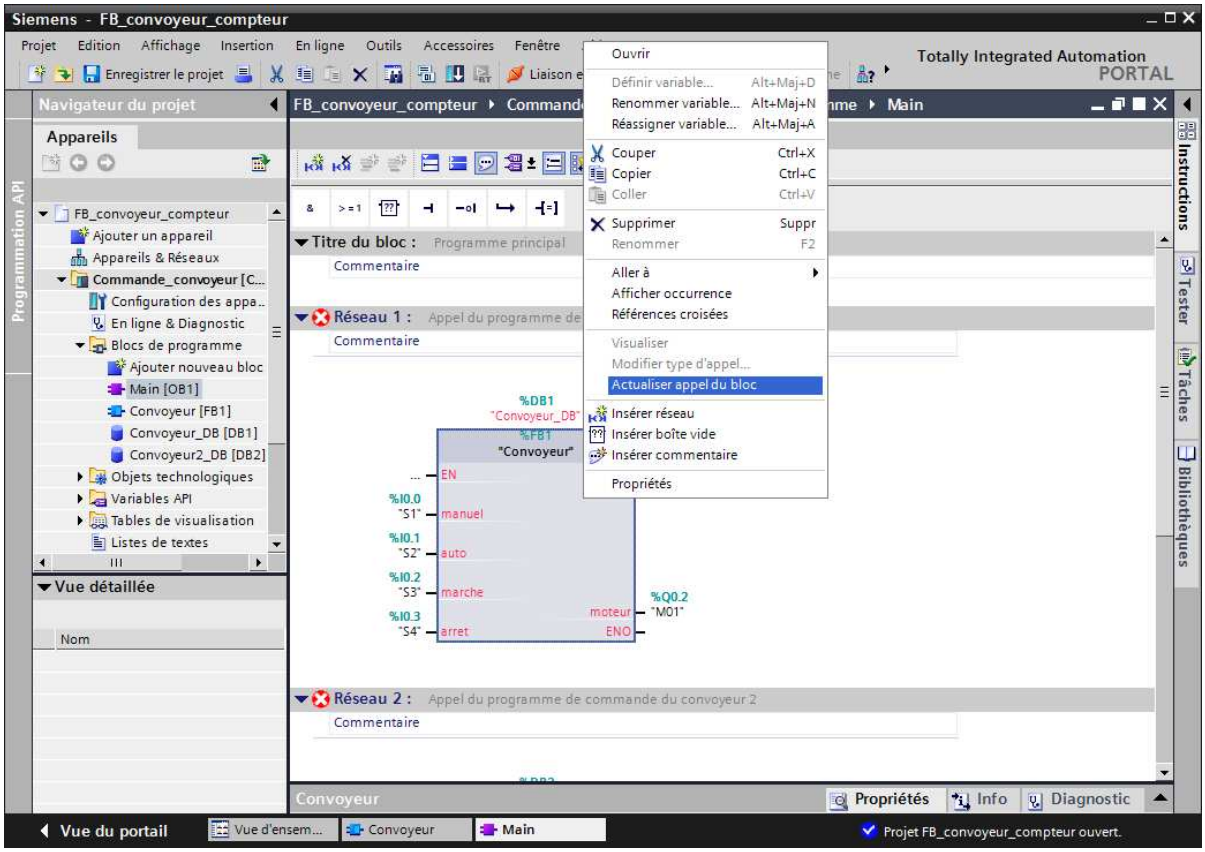
Indication :

Le décompteur est le plus approprié pour le comptage de quantités spécifiques, puisque la sortie binaire **Q** peut être utilisée pour d'autres connexions. Sinon, un comparateur devrait être programmé.

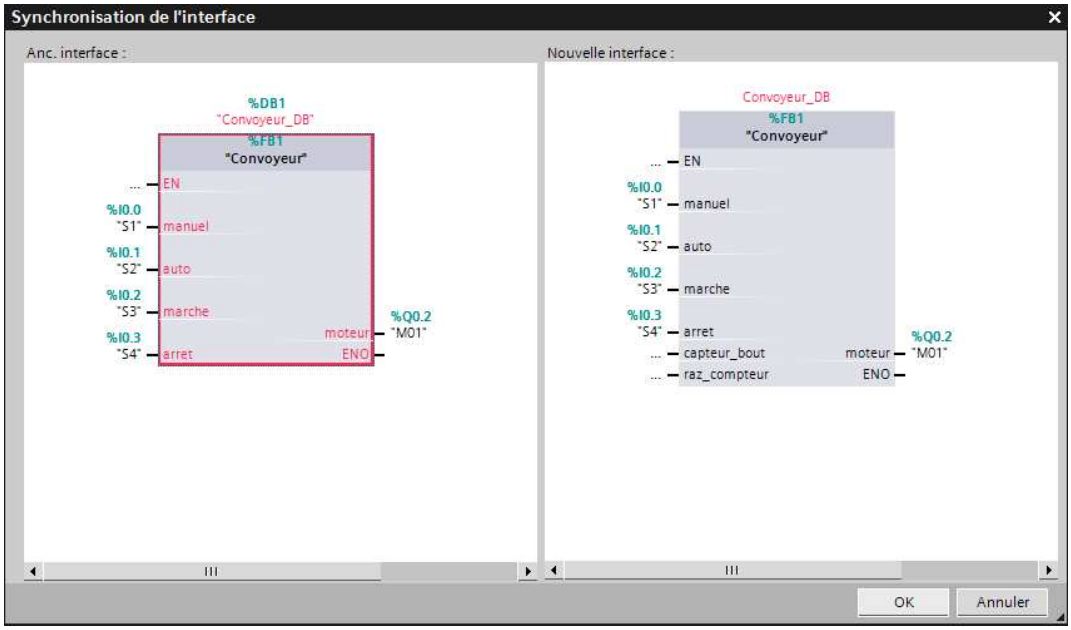
16. Ouvrez maintenant le bloc « **Main [OB1]** » pour mettre à jour l'appel du bloc « **Convoyeur [FB1]** ».



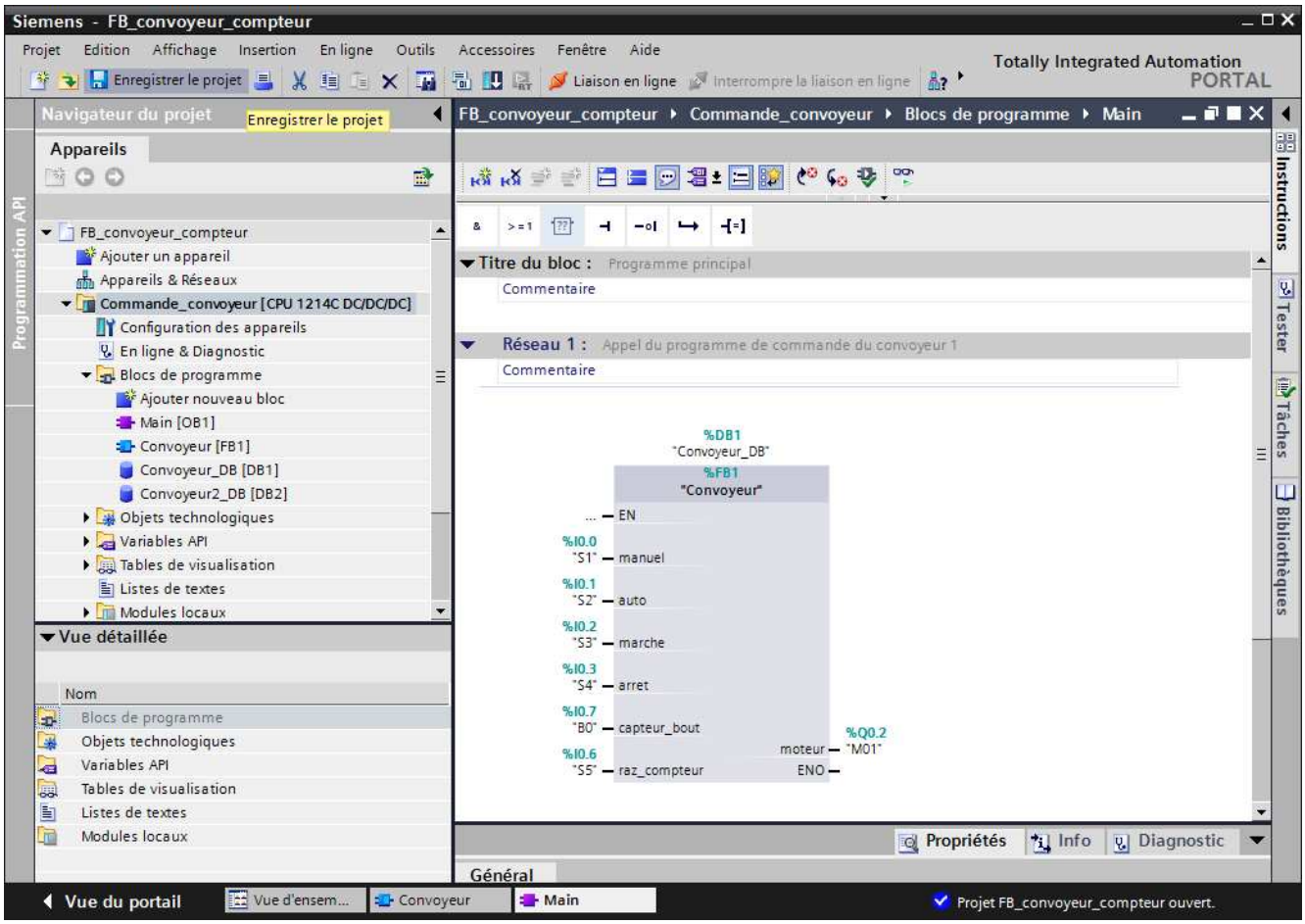
17. Dans le bloc « **Main [OB1]** », faites un clic droit sur « **Convoyeur** » et sélectionnez « **Actualiser appel du bloc** ».



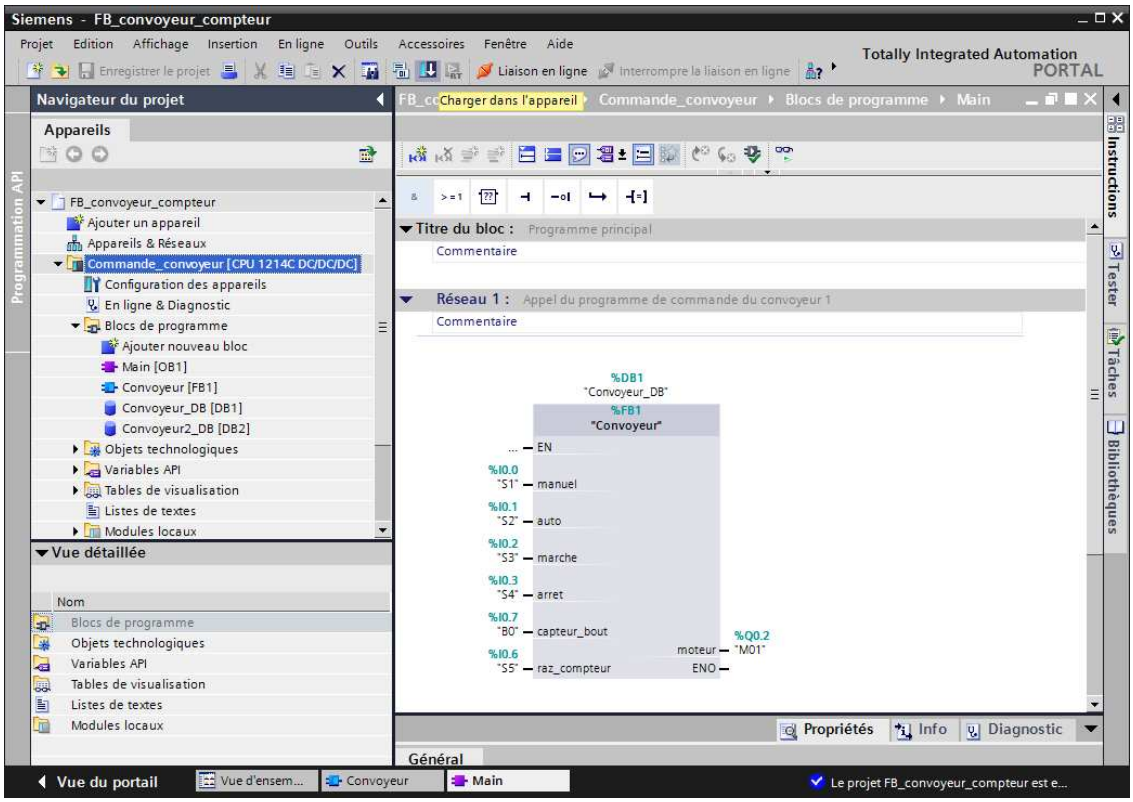
18. Vérifiez alors la « **Nouvelle interface** » et confirmez avec « **OK** ».



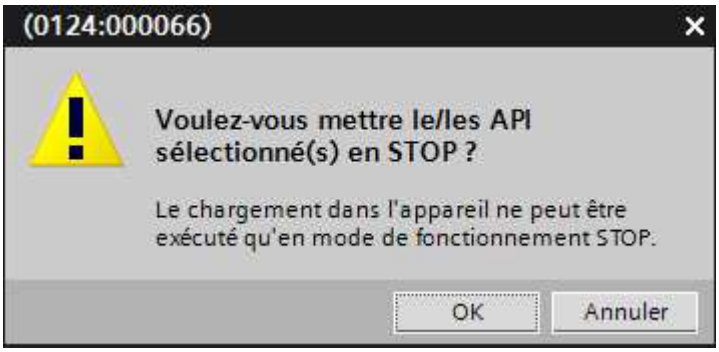
19. Assignez les nouvelles entrées aux bonnes variables et cliquez sur Enregistrer le projet pour sauvegarder votre projet.



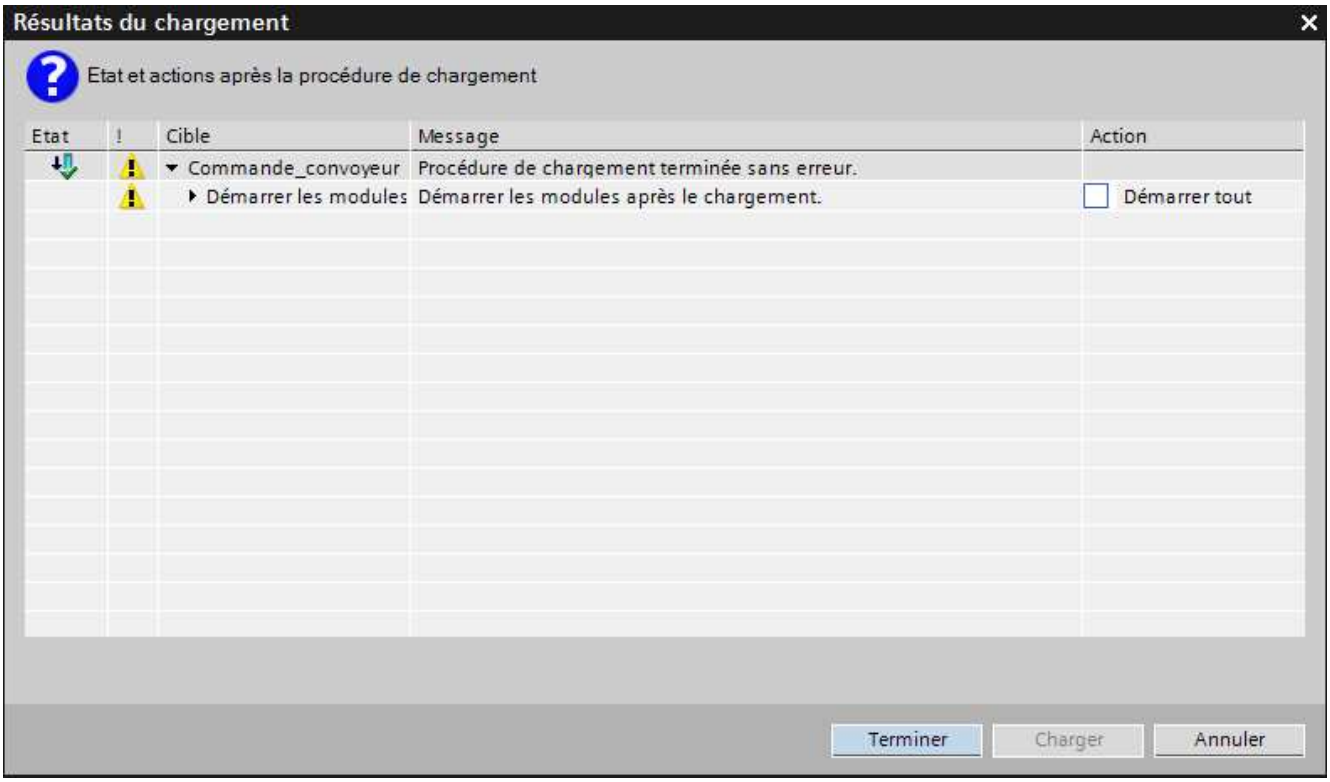
20. Pour charger votre programme entier dans la CPU, surlignez d'abord « **Commande_convoyeur [CPU 1214C DC/DC/DC]** » en cliquant une fois dessus. Cliquez ensuite sur le symbole « **Charger dans l'appareil** ».



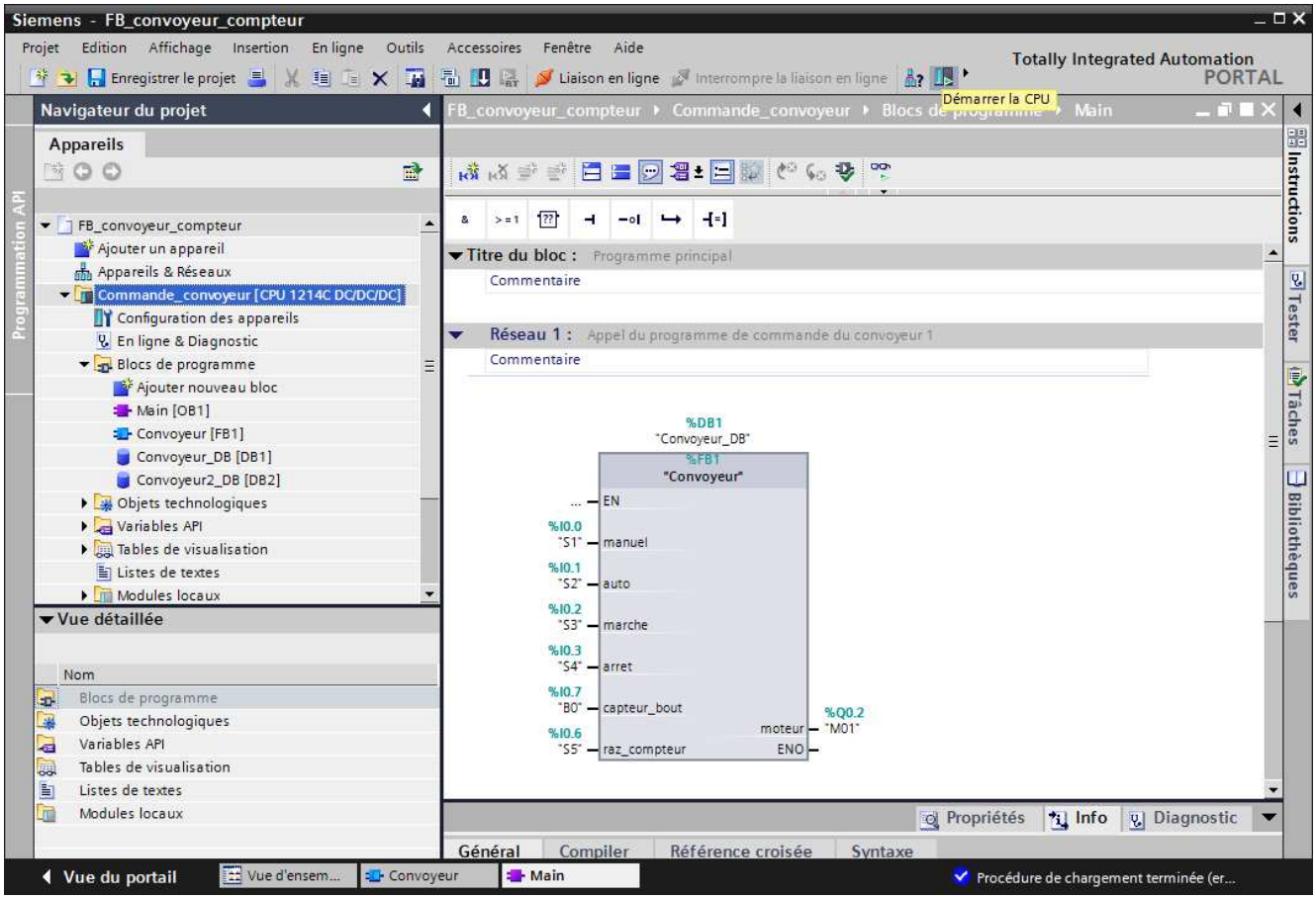
21. Si la CPU était en mode « **RUN** », un message s'affichera et vous demandera si vous voulez mettre la CPU en mode « **STOP** ». Confirmez le choix en cliquant sur « **OK** ».



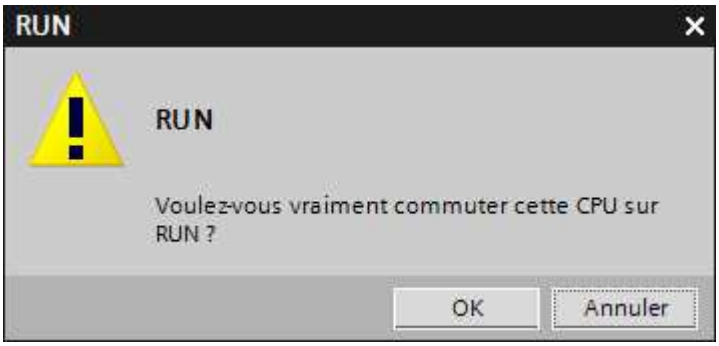
22. Une nouvelle fenêtre apparaît. Cliquez sur « **Charger** » une fois de plus. Pendant le chargement, l'état de progression est affiché dans la fenêtre. Si le chargement s'est correctement déroulé, le résultat s'affiche dans une nouvelle fenêtre. Cliquez finalement sur « **Terminer** ».



23. Ensuite, démarrez la CPU en cliquant sur le symbole  « **Démarrer la CPU** ».



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



25. En cliquant sur l'icône « **Activer/désactiver visualisation du programme** », il est possible de surveiller l'état du compteur pendant que vous testez le programme en commutant les interrupteurs de la maquette.