

# PROGRAMMATION ST/LD/SFC SOUS OPENPLC

JEAN-PHILIPPE ILARY  
 Département GEII – IUT Ville d'Avray  
 50 Rue de Sèvres, 92410 Ville d'Avray  
 jpilary@parisnanterre.fr

**Résumé :** Pendant le premier confinement, l'enseignement à distance du module automatisme nous ont amené à mettre en place des solutions pour palier à certaines difficultés matérielles et logicielles. Cette matière s'appuyait, en situation normale, sur l'emploi du logiciel Unity de Schneider associé à des Modicon M340. Malgré la mise à disposition, durant cette période de distanciel, d'une solution logicielle orientée simulation, certains étudiants ont eu des difficultés d'ordre matériel (ordinateur Apple, version de Windows etc.), c'est pourquoi nous avons orienté ces étudiants vers la solution logicielle OpenPLC. Ce document a pour ambition de fournir à un étudiant les bases pour débiter en autonomie avec ce logiciel.

OpenPLC Editor est une alternative logicielle Open Source qui répond à la norme CEI 61-131. Ce logiciel permet de développer des programmes API sur des plateformes Raspberry/Windows/Linux.

## I/ Introduction

Après avoir présenté les possibilités de la suite logicielle OpenPLC, ce document, adapté pour la revue, présente les démarches à suivre par l'étudiant afin qu'il puisse développer en autonomie.

La première partie permet la prise en main du logiciel OpenPLC Editor :

- Configurer le projet,
- Créer un programme LADDER simple et programmer un grafset en SFC,
- Utiliser le simulateur pour valider le fonctionnement du programme.

La seconde partie permet de valider les programmes du TD programmation LD/ST/SFC :

- Programmer un cahier des charges simple en langage LD/ST/SFC,
- Analyser les éventuelles erreurs de compilation,
- Valider les programmes en simulation,
- Rédiger un compte-rendu.

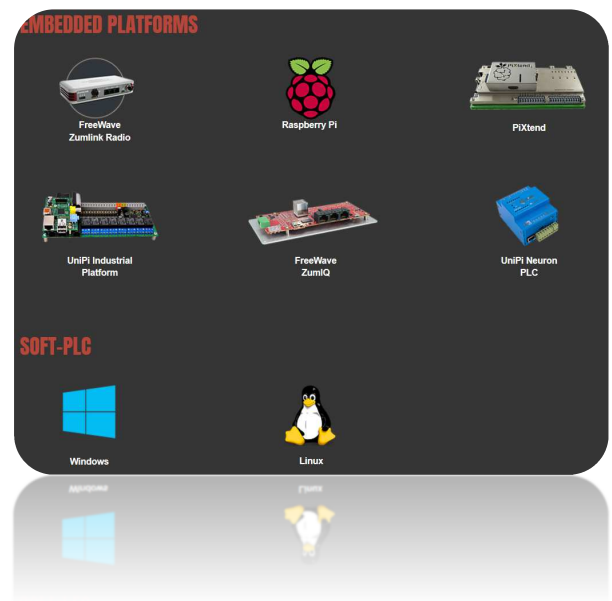
## II/ Installation & PRISE EN MAIN

Le projet OpenPLC propose en ensemble d'éléments afin de développer des automatismes en Open source. Il fournit trois parties : un Runtime, un Editeur et un outil de développement d'IHM. L'éditeur est le logiciel qui s'exécute sur votre ordinateur et est utilisé pour créer vos programmes PLC. Enfin, ScadaBR est le constructeur HMI. Avec ScadaBR, vous pouvez créer de belles animations web qui refléteront l'état de votre processus. ScadaBR communique avec OpenPLC Runtime sur Modbus/TCP.

### II.1/ L'Automate simulé (Runtime)

Le Runtime doit être installé sur votre appareil (il est responsable de l'exécution de votre programme PLC). Il permet de disposer d'un API à faible coût. Pour

l'installer, il suffit de se rendre sur le site <https://www.openplcproject.com/runtime> et de choisir la version souhaitée, soit pour carte Raspberry (ou équivalent), soit pour Windows, Linux. Exécutez, cela va permettre d'installer les logiciels nécessaires à la simulation d'un API. Des consoles DOS s'ouvriront pour télécharger les logiciels manquants.



L'installation prendra un certain temps !

Il est aussi possible de disposer d'Entrées/Sorties déportées à travers différents matériels comme une carte Arduino ou un ESP8266.

### II.2/ L'éditeur de programme

Il faut se rendre sur le site <https://www.openplcproject.com/plcopen-editor> et choisir la version à télécharger. Pour Mac OS/X voir <https://openplc.discussion.community/post/openplc-editor-on-mac-os-x-9905213>

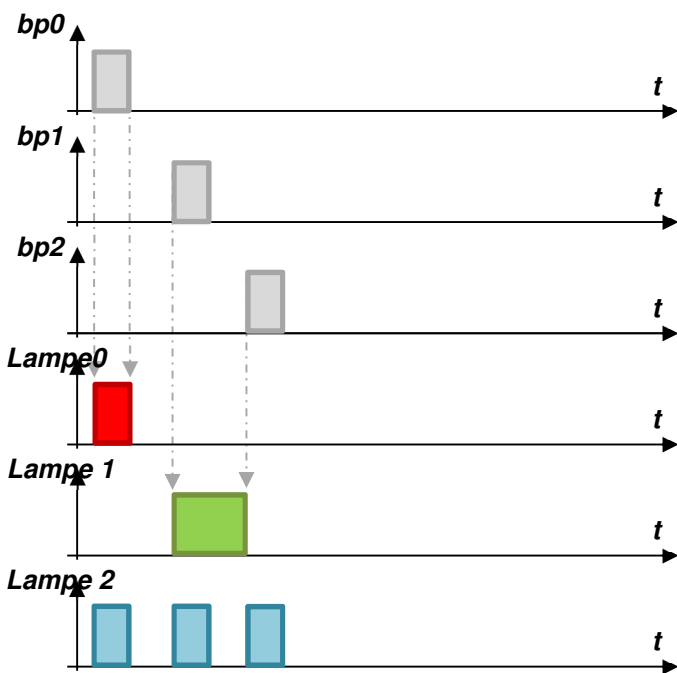
Pour la suite de l'article, nous travaillerons avec un API sur un système Windows.



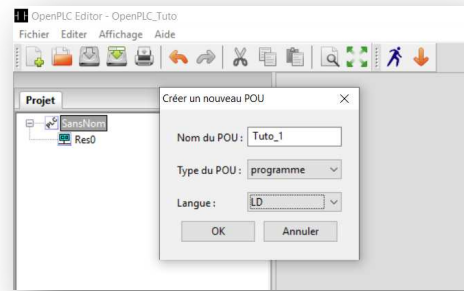
### III/ Programmation en LADDER

#### III.1/ Créer un projet et définir une architecture matérielle

Nous allons créer un programme répondant au chronogramme ci-après. Nous disposerons de trois boutons poussoirs (bp, bp1 et bp2) et de trois lampes (Lampe0, Lampe1 et Lampe2). Ce sera notre « Bonjour le monde » que tout programmeur connaît.



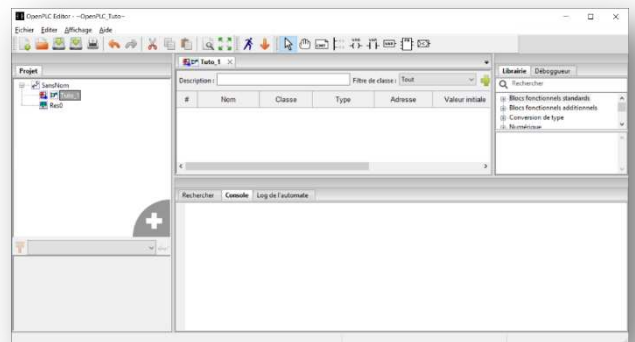
Créer un nouveau projet (CTRL+N) et choisir le langage Ladder (LD)



#### Sauvegarder le projet dans un répertoire vide !

(Remarque : attention, l'emplacement de sauvegarde peut générer des erreurs lors de la compilation, il suffit alors de le changer)

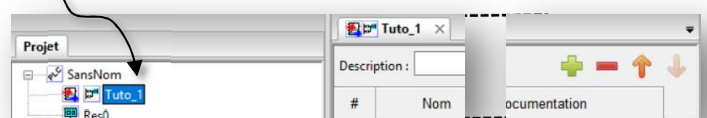
On obtient la fenêtre d'édition suivante qui va nous permettre d'écrire nos programmes API.



#### III.2/ Associer un symbole aux variables (E/S TOR)

Lors d'une programmation API, il est toujours préférable, comme en programmation informatique classique, de commencer par réfléchir à la structuration du programme, et en automatisme, à réaliser une table d'adressage avec les mnémoniques des variables associées à leur type.

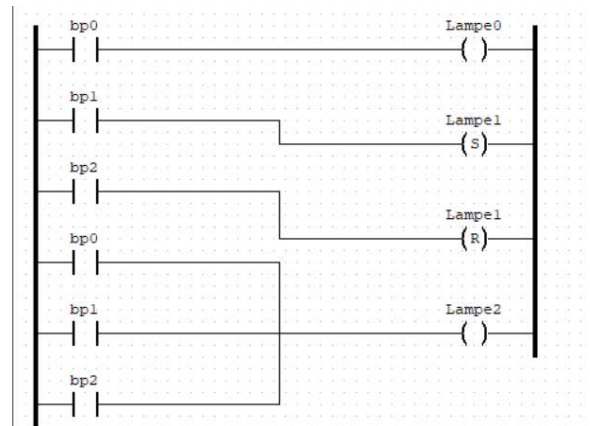
A partir du navigateur projet, double cliquer sur Tuto\_1 afin de pouvoir afficher la fenêtre d'édition.



Cliquer sur l'icône  afin de rajouter une variable. (Explication de l'Adressage en annexe)

Définir trois variables d'entrée TOR et trois variables de sortie TOR en suivant la table d'adressage ci-après. Les données sont définies en BOOL.

#	Nom	Classe	Type	Adresse
1	bp0	Locale	BOOL	%IX0.0
2	bp1	Locale	BOOL	%IX0.1
3	bp2	Locale	BOOL	%IX0.2
4	Lampe0	Locale	BOOL	%QX0.0
5	Lampe1	Locale	BOOL	%QX0.1
6	Lampe2	Locale	BOOL	%QX0.2

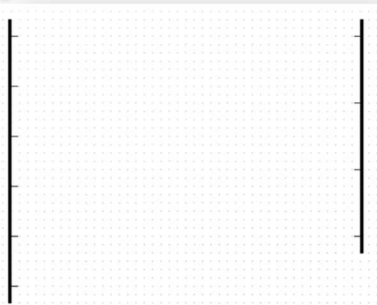


### III.3/ Créer un programme en LADDER

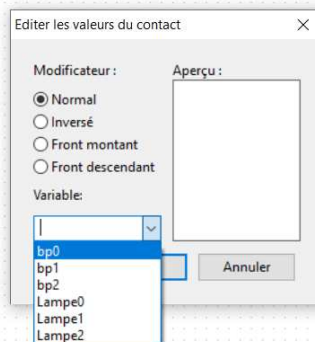
La zone de saisie du Ladder est sous le tableau des variables. Cela se fait grâce à la barre d'outils, saisir le programme suivant :



Commencez par placer les deux connexions verticales correspondant à l'alimentation du circuit. Côté gauche 6 départs et côté droit 4 départs. Pour les contacts ou bobines, faire de même.



Ensuite, pour les contacts (même démarche pour les bobines) saisir ses caractéristiques.



Ces outils vont permettre d'élaborer le programme API répondant au cahier des charges. Cela donne :

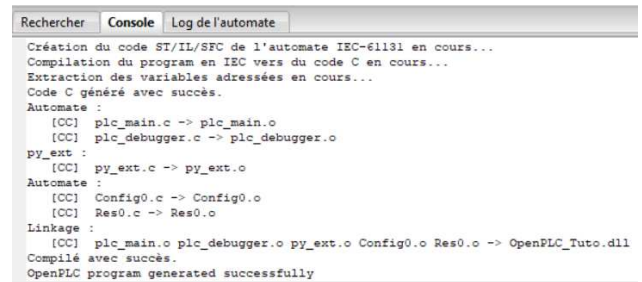
Il suffit alors d'enregistrer le projet

### III.4/ Tester et valider le programme

#### III.4.1/ Compiler



Pour compiler le projet, sélectionner l'icône . Le compilateur indique en bas de page les éventuels avertissements (non bloquant, à lire quand même) et erreurs (bloquant).



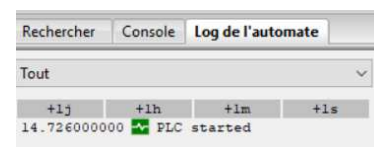
Il vous demande un nom de programme se terminant par .st, car quelque-soit le langage de programmation sélectionné, votre programme est traduit en langage ST. Vous pouvez ouvrir ce fichier avec le blocNote pour vous en rendre compte.

#### III.4.2/ Se connecter à l'API simulé, exécuter le programme et le valider

Pour se connecter à l'API simulé, sélectionner l'icône :



Le logiciel compile votre programme, le transfère au simulateur et bascule l'API en RUN. Vérifiez alors que l'API (PLC) est bien en RUN (started).



Pour simuler votre application, cliquer sur Déboguer l'instance dans le panneau gauche en cliquant sur la lunette à droite du texte Config0.Res0.instance0 :

Un nouvel onglet apparaît.




Les lignes en vert sont activées, celles en noir ne le sont pas.

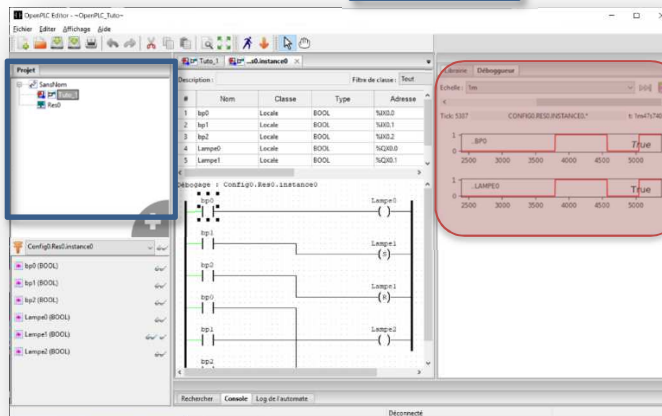
Pour revenir à l'édition du programme, sélectionnez l'onglet p0 sur la capture ci-dessus (pour vous l'onglet portera le nom de votre programme).

Tester le fonctionnement du programme avec la table d'animation et le programme LADDER animé :

- Lampe0 active si bp0 appuyé,

- Lampe1 active si appui sur bp1, l'ordre est mémorisé. Lampe1 est désactivée par l'appui sur bp2,
- Lampe2 active si bp0 appuyé ou bp1 appuyé ou bp3 appuyé.

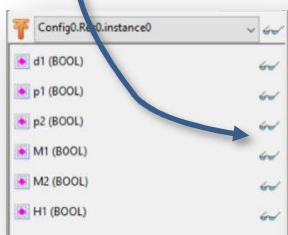
Lorsque vous avez terminé, déforcez les variables, passer l'automate en STOP (icône ) et revenez sur la fenêtre projet.



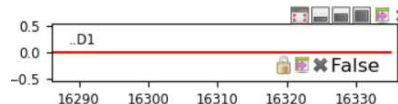
AIDE



**Pour visualiser une variable dans l'onglet Débugueur !**

Sélectionner les variables que vous souhaitez voir apparaître dans le chronogramme du débogueur en cliquant sur les lunettes.

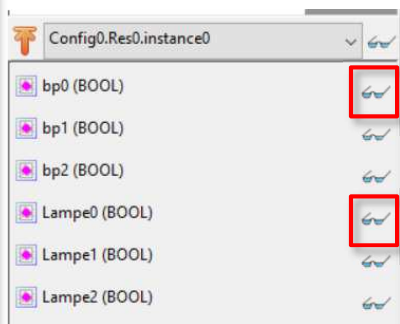


Double cliquer sur la variable dans la fenêtre « Débugueur » afin de faire apparaître son chronogramme, si celui-ci n'apparaît pas.

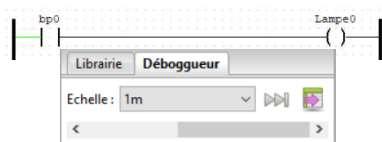


Survoler avec la souris, la valeur de la variable (ici FALSE) afin de faire apparaître   False. Il faut ensuite cliquer sur le cadenas, la fenêtre suivante apparaît :

Pour observer l'évolution des variables de sorties en fonction de l'état des variables d'entrées en LADDER, il est possible de suivre la démarche ci-après pour valider, par exemple, cette ligne de code :



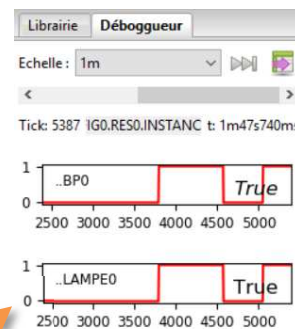
1 Sélectionner les variables à visualiser



2 Période du graphe 1 minute



3 Clic Droit pour sélectionner l'état souhaité de la variable

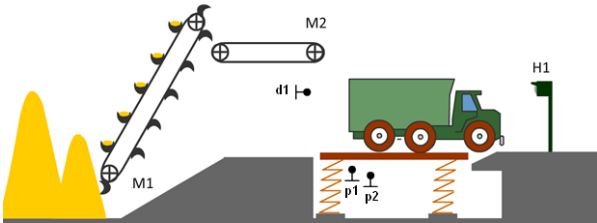


4 Observer les chronogrammes

### IV/Programmer en SFC (grafcet)

Afin de comprendre comment procéder pour réaliser un programme SFC, nous nous appuyons sur le cahier des charges suivant :

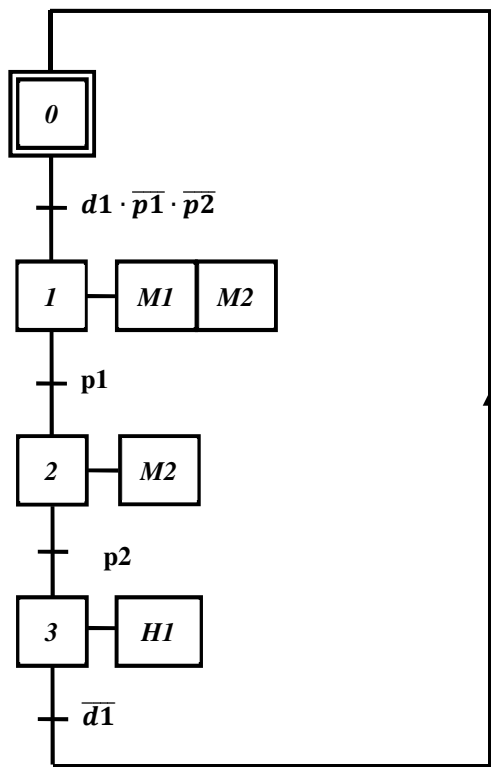
Un camion arrive en reculant, il actionne le détecteur de présence d1 ce qui entraîne le démarrage du tapis 1 (moteur M1) et du tapis 2 (moteur M2).



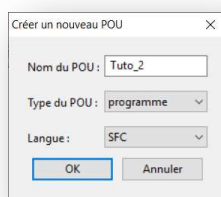
Le camion est placé sur un système de pesée (en contact avec d1). Quand le camion est en partie chargé, l'action de la plateforme sur le détecteur p1 doit stopper le tapis 1, et maintenir le tapis 2 en action.

Le tapis 2 continue de fonctionner jusqu'à l'activation du détecteur p2. Le voyant H1 s'allume pour signaler que le chargement est terminé.

Le grafcet suivant répond au cahier des charges :



Commencer par créer un nouveau projet, le nommer Tuto\_2. La programmation se fera en SFC.



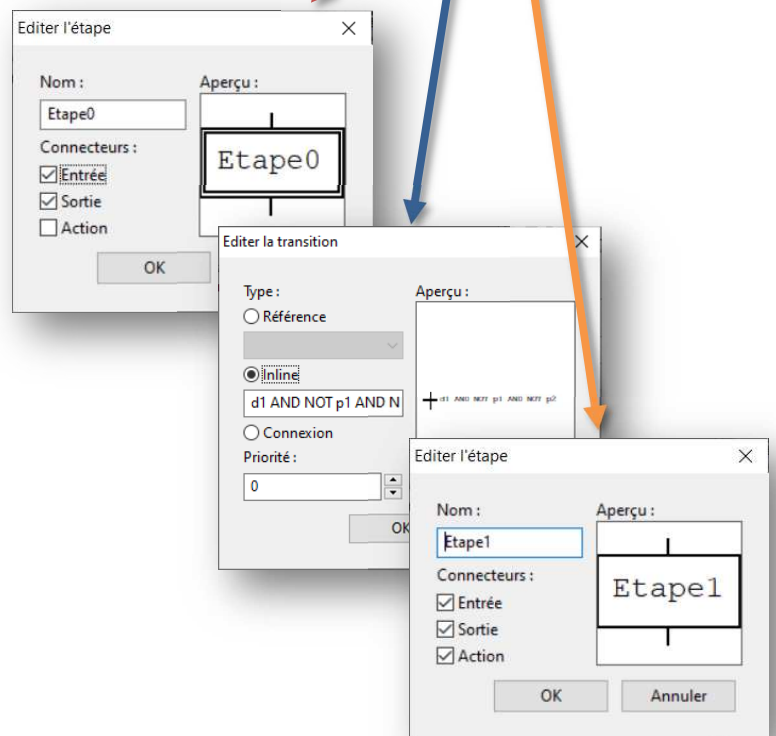
Ensuite, définir les symboles qui seront utilisés dans le programme. Il faut définir le type et l'adresse pour chacune des variables.

#	Nom	Classe	Type	Adresse
1	d1	Locale	BOOL	%IX0.0
2	p1	Locale	BOOL	%IX0.1
3	p2	Locale	BOOL	%IX0.2
4	M1	Locale	BOOL	%QX0.0
5	M2	Locale	BOOL	%QX0.1
6	H1	Locale	BOOL	%QX0.2

Il ne reste plus qu'à dessiner le grafcet !

#### Palette d'outils

Utilisez les icônes disponibles pour créer le grafcet demandé :



#### Mise en place des transitions et réceptivités

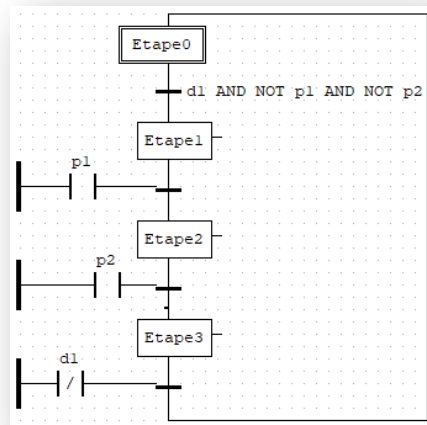
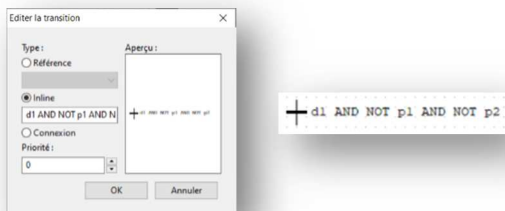
Pour réaliser la construction du squelette du grafcet, il est souhaitable de suivre la démarche ci-dessous :

Clic Gauche sur l'extrémité sous l'étape, et vous obtiendrez le menu proposant Transition/Divergence. En choisissant « Transition », la fenêtre « Editer la transition » apparaît.

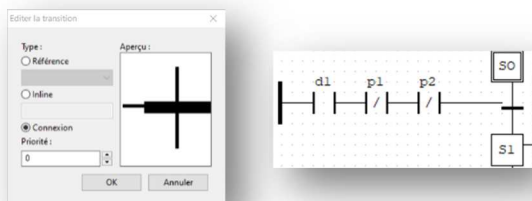


Il est alors possible de saisir la réceptivité de trois façons, on verra ici deux solutions.

En choisissant **Inline**, il sera possible de saisir l'équation logique de la réceptivité suivant la norme CEI 61-131 avec les opérateurs logiques AND, OR, XOR, NOT :



En choisissant **Connexion**, il sera possible d'écrire l'équation logique de la réceptivité en langage ladder en commençant à gauche par une barre d'alimentation et en rebouclant sur la transition.



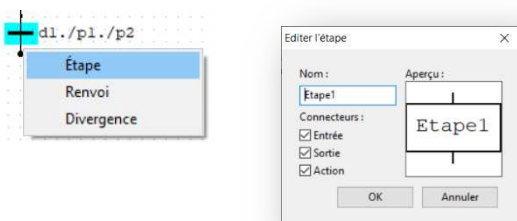
### Réalisation des Actions

Ensuite, il faut affecter les actions aux étapes. Pour cela, cliquez sur le départ de l'action. Le menu « Ajouter un bloc fonctionnel » apparaît alors :

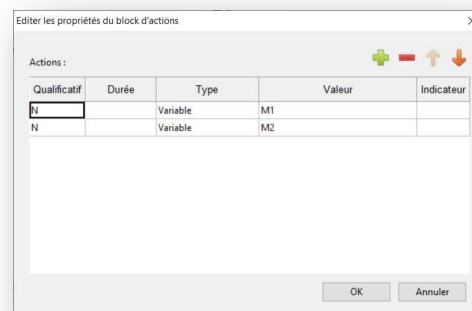


On procédera de même pour la réalisation d'une étape supplémentaire :

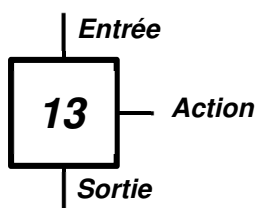
Cette sélection fait apparaître la fenêtre d'édition de l'étape.



Apparaît alors la fenêtre ci-dessous. Les qualificatifs sont ceux définis par la norme, comme indiqué ci-après.



Remarque : Une étape peut avoir 3 connexions (Connecteur)



Vous devez obtenir par exemple :

Comme on peut voir ci-dessous, le qualificatif N permet que l'action soit active tant que l'étape l'est.

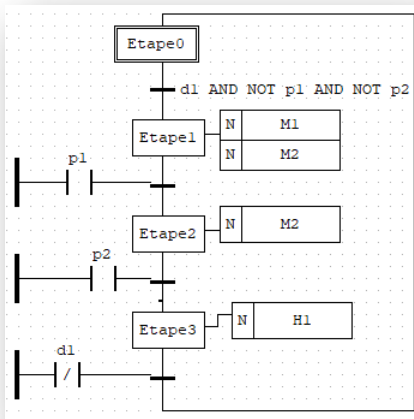
#### Qualificatif

Pour associer des actions à des étapes CEI, vous disposez des qualificatifs (qualifier) suivant:


Qualificatif	Description	Effet
N	Non-stored (non mémorisé)	l'action demeure active aussi longtemps que l'étape demeure active
R	overriding Reset (remise à zéro prioritaire)	désactivation de l'action
S	Set (Stored) (positionné (mémorisé))	activation de l'action, qui demeure ensuite active jusqu'au prochain reset
L	time Limited (limite dans le temps)	activation de l'action pendant une durée déterminée
D	time Delayed (temporisé)	activation de l'action après un certain temps, pour autant que l'étape demeure active
P	Pulse (impulsion)	l'action est exécutée exactement une fois, lorsque l'étape est activée
SD	Stored and time Delayed (mémorisé et temporisé)	activation de l'action après un certain temps; l'action demeure ensuite active jusqu'au prochain reset
DS	Delayed and Stored (temporisé et mémorisé)	activation de l'action après un certain temps, pour autant que l'étape demeure active; l'action demeure alors active jusqu'au prochain reset
SL	Stored and time Limited (mémorisé et limité dans le temps)	activation de l'action pendant une durée déterminée

Les qualificatifs L, D, SD, DS et SL doivent être accompagnés d'une spécification de temps sous format de constante TIME, p.ex. L T#5s.

On doit obtenir au final une écriture équivalente à :





La programmation du grafcet est terminée.


**Rappel :** Pour revenir à l'édition du programme, appuyer sur Stop  et sélectionner l'onglet du programme :



En programmeur, il est de notre devoir de commenter le programme. Pour ajouter des commentaires sur la page Chart, utiliser le raccourci en pressant la touche F8.

## V/ Validation du fonctionnement

Maintenant que le programme est complet, on va commencer par le compiler  (rappel : le dossier du projet doit être dans un répertoire bien accessible (pas sur un lecteur réseau par exemple). Zéro erreur, vous pouvez alors l'exécuter .

Ensuite, comme pour la programmation LADDER, il est possible de superviser les variables (voir Aide plus haut). Cela se fait par  et le résultat apparaît à droite dans la fenêtre Débogueur.

Si vous avez un API compatible OpenPLC (Raspberry par exemple), il vous est possible d'y transférer le programme et de l'exécuter (voir le tutorial en ligne :

<https://www.openplcproject.com/reference/basics/upload>

## VI/ Création d'une supervision

Il s'agit ici d'un contrôle de surveillance et d'acquisition de données (type SCADA) qui vous permet de créer des écrans interactifs, également appelés

IHM pour une supervision locale), pour vos projets d'automatisation. ScadaBR peut interagir avec plusieurs PLCs, y compris un API OpenPLC.



La façon la plus simple d'installer ScadaBR est d'utiliser une machine virtuelle. Dans la machine virtuelle, tout ce que vous avez à faire est de charger le fichier d'image ScadaBR dans VirtualBox par exemple pour avoir un environnement ScadaBR prêt à l'emploi sur votre système (voir le site pour le détail de l'installation :

<https://www.openplcproject.com/reference/scadabr/>).

## VII/ Conclusion

Les sept pages annexes qui suivent sont des outils qui serviront aux étudiants, à qui ce tutoriel est destiné.

Je remercie, pour leur aide Claire BASSET (IUT GEII - Ville d'Avray) et Jean-Michel GAY (BTS ET - Versailles)

Ce travail n'est pas terminé, mais vu les circonstances, il m'a semblé intéressant de le partager en l'état. Ce document et les fichiers devraient être disponibles sur Eduscol.

Prenez soin de vous !

# ANNEXE 1

## Syntaxe des adresses I/O TOR sous OpenPLC

### %IX & %QX

<https://www.openplcproject.com/reference/plc-addressing/>

#### Entrées TOR :

L'adressage des entrées TOR se fait de façon classique. La plage va de %IX0.0 à %IX99.7 (soit 100×8 entrées TOR possibles) :

Description de la syntaxe :

- %IX0.1 : Input
- %IX0.1 : « Carte » n°0, il y a 100 « cartes » possible à 8 entrées
- %IX0.1 : Entrée n°1 de la carte n°0, il y a 8 entrées/carte

#	Nom	Classe	Type	Adresse
1	Entree_TOR	Locale	BOOL	%IX0.1
2	Sortie_TOR	Locale	BOOL	%QX0.2

#### Sorties TOR :

L'adressage des sorties TOR se fait aussi de façon classique. La plage va de %QX0.0 à %QX99.7 (soit 100×8 sorties TOR possibles) :

Description de la syntaxe :

- %QX0.2 : Sortie
- %QX0.2 : « Carte » n°0, il y a 100 « cartes » possible à 8 sorties
- %QX0.2 : Sortie n°2 de la carte n°0, il y a 8 sorties/carte

Extrait du lien ci-dessus :

<data-size>	Common Name	Number of Bits	Elementary Data Types
X	Bit	1	BOOL
B	Byte	8	BYTE, , SINTUSINT
W	Word	16	WORD, , INTUINT
D	Double word	32	DWORD, , DINTUDINTFLOAT
L	Long word	64	LWORD, , LINTULINTDOUBLE

### A titre de comparaison Sous l'environnement d'Unity Pro

#### Entrées TOR : (%IRack.Module.Voie.[rang])

L'adressage des entrées TOR se fait de façon classique :

Description de la syntaxe :

- %I0.3.1.0 : Input
- %I0.3.1.0 : Rack n°0,
- %I0.3.1.0 : Carte n°3,
- %I0.3.1.0 : Bit n°1 de la carte n°3,
- %I0.3.1.0 : Rang =0 pas de propriété pour cette entrée.

#### Sorties TOR : (%QRack.Module.Voie.[rang])

L'adressage des sorties TOR se fait aussi de façon classique :

Description de la syntaxe :

- %Q0.2.1.0 : Sortie
- %Q0.2.1.0 : Rack n°0,
- %Q0.2.1.0 : Carte n°2,
- %Q0.2.1.0 : Bit n°1 de la carte n°2,
- %Q0.2.1.0 : Rang =0 pas de propriété pour cette entrée.



## ANNEXE 2

### Syntaxe des adresses I/O Analogiques sous OpenPLC

#### %IW & %QW

##### Entrées ANALOGIQUES :

L'adressage des entrées ANALOGIQUE suit la représentation suivante et la plage va de %IW0 à %IW99 :

##### Description de la syntaxe :

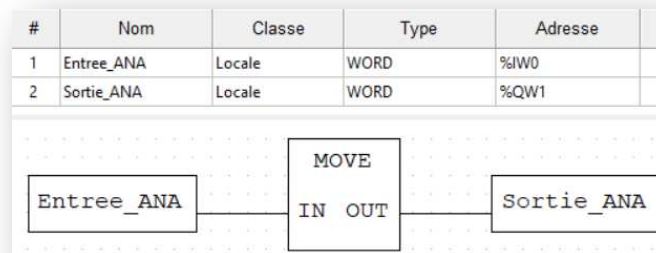
- %IW0 : Input,
- %IW0 : Mot de 16 bits (donc de 0 à 65535),
- %IW0 : Entrée analogique n°0.

##### Sorties ANALOGIQUES :

L'adressage des sorties ANALOGIQUE suit la représentation suivante et la plage va de %QW0 à %QW99 :

##### Description de la syntaxe :

- %QW1 : Sortie,
- %QW1 : Mot de 16 bits (donc de 0 à 65535),
- %QW1 : Sortie analogique n°1.



#### ***A titre de comparaison Sous l'environnement d'Unity Pro***

##### Entrées ANA : (%IWRack.Module.Voie.[rang])

L'adressage des entrées ANA se fait de façon classique :

##### Description de la syntaxe :

- %IW0.3.1.0 : Input
- %IW0.3.1.0 : Rack n°0,
- %IW0.3.1.0 : Carte n°3,
- %IW0.3.1.0 : mot n°1 de la carte n°3,
- %IW0.3.1.0 : Rang =0 pas de propriété pour cette entrée (on peut s'en passer)

##### Sorties ANA : (%QWRack.Module.Voie.[rang])

L'adressage des sorties ANA se fait aussi de façon normalisée :

##### Description de la syntaxe :

- %QW0.2.1 : **Sortie**
- %QW0.2.1 : Rack n°0,
- %QW0.2.1 : Carte n°2,
- %QW0.2.1 : Voie n°1 de la carte n°2.

## ANNEXE 3

### Syntaxe des adresses mémoire interne (Registres) sous OpenPLC

#### %M

##### Registres sur 8 bits :

La plage va de %MB0 à %MB1023 :

Description de la syntaxe :

- %**M**B1 : **M**émoire - Registre
- %**M**B1 : Byte de 8 bits pouvant être rangé dans ce registre
- %**M**B1 : Registre n°1

##### Registres sur 16 bits :

La plage va de %MW0 à %MW1023 :

Description de la syntaxe :

- %**M**W1 : **M**émoire - Registre
- %**M**W1 : Mot de 16 bits pouvant être rangé dans ce registre
- %**M**W1 : Registre n°1

##### Registres sur 32 bits :

La plage va de %MD0 à %MD1023 :

Description de la syntaxe :

- %**M**D2 : **M**émoire - Registre
- %**M**D2 : Double Mot de 32 bits pouvant être rangé dans ce registre
- %**M**D2 : Registre n°2

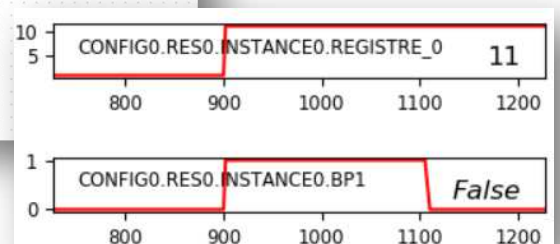
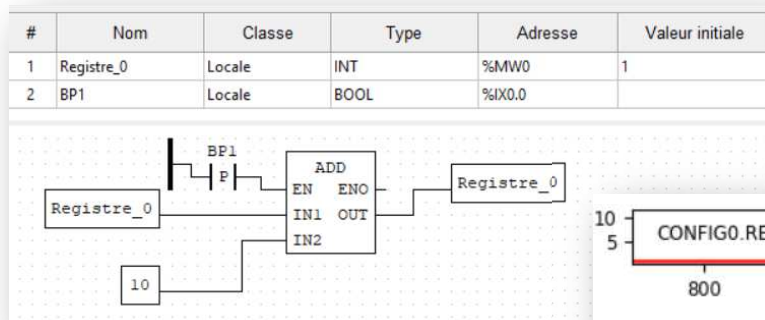
##### Registres sur 64 bits :

La plage va de %ML0 à %ML1023 :

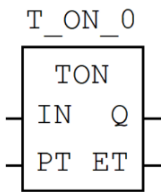
Description de la syntaxe :

- %**M**L3 : **M**émoire - Registre
- %**M**L3 : Long Mot de 64 bits pouvant être rangé dans ce registre
- %**M**L3 : Registre n°3

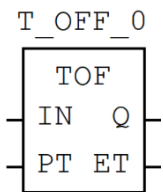
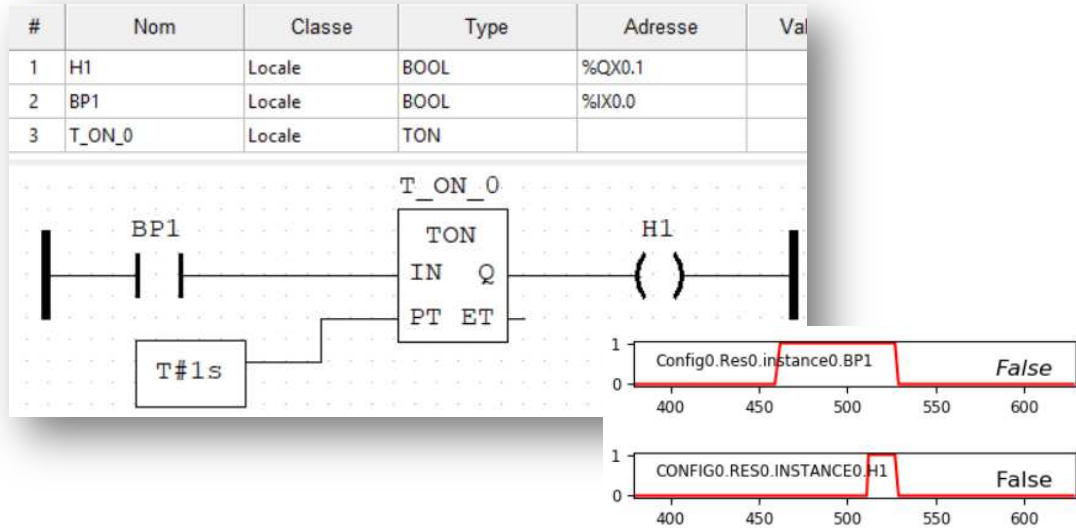
Registre\_0=Registre\_0+10 Sur front montant de BP1



### ANNEXE 3 Temporisations (LADDER)

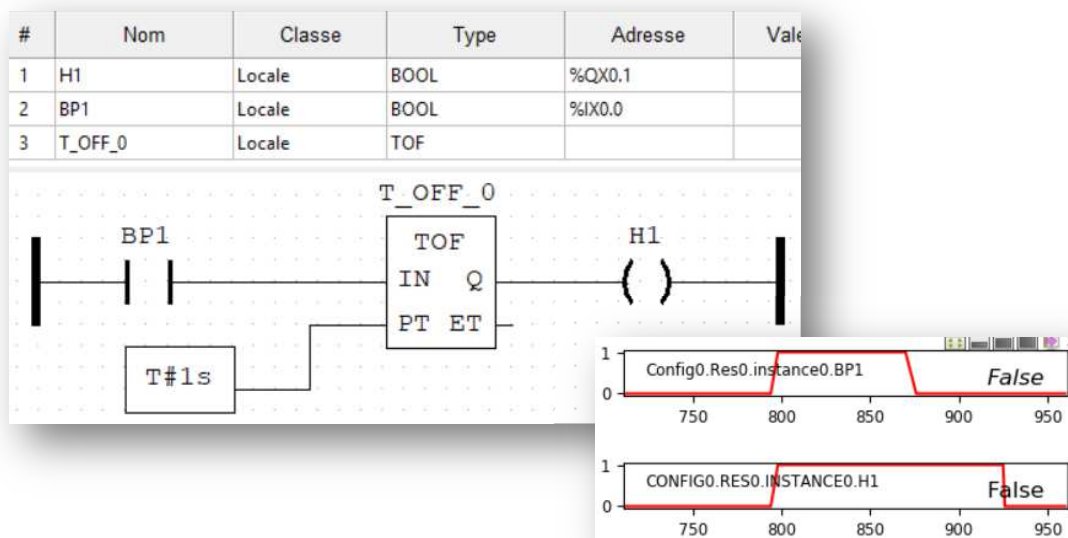


La temporisation à retard peut être utilisée pour retarder un front montant d'une durée donnée. Dans l'exemple ci-dessous, la temporisation au travail est mise à 1s



La temporisation de retard peut être utilisée pour retarder un front descendant d'une durée donnée.

Dans l'exemple ci-dessous, la temporisation est au repos est mise à 1s



## ANNEXE 3

### Temporisations (SFC)

Les temporisations en SFC sont possible grâce aux Qualificatifs suivants :

#### Qualificatif

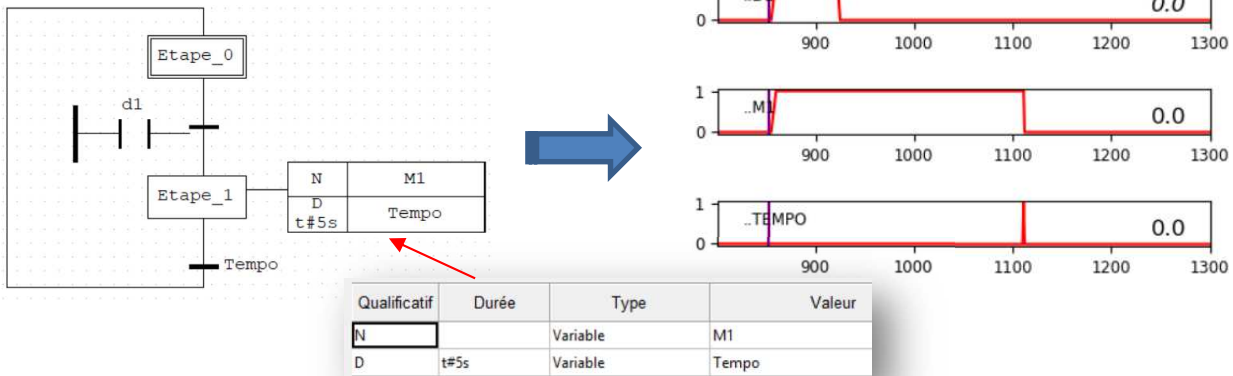
Pour associer des actions à des étapes CEI, vous disposez des qualificatifs (qualifier) suivant:

<b>N</b>	Non-stored (non mémorisé)	l'action demeure active aussi longtemps que l'étape demeure active
<b>R</b>	overriding Reset (remise à zéro prioritaire)	désactivation de l'action
<b>S</b>	Set (Stored) (positionné (mémorisé)	activation de l'action, qui demeure ensuite active jusqu'au prochain reset
<b>L</b>	time Limited (limite dans le temps)	activation de l'action pendant une durée déterminée
<b>D</b>	time Delayed (temporisé)	activation de l'action après un certain temps, pour autant que l'étape demeure active
<b>P</b>	Pulse (impulsion)	l'action est exécutée exactement une fois, lorsque l'étape est activée
<b>SD</b>	Stored and time Delayed (mémorisé et temporisé)	activation de l'action après un certain temps; l'action demeure ensuite active jusqu'au prochain reset
<b>DS</b>	Delayed and Stored (temporisé et mémorisé)	activation de l'action après un certain temps, pour autant que l'étape demeure active; l'action demeure alors active jusqu'au prochain reset
<b>SL</b>	Stored and time Limited (mémorisé et limité dans le temps)	activation de l'action pendant une durée déterminée

Les qualificatifs L, D, SD, DS et SL doivent être accompagnés d'une spécification de temps sous format de constante TIME, p.ex. L T#5s.

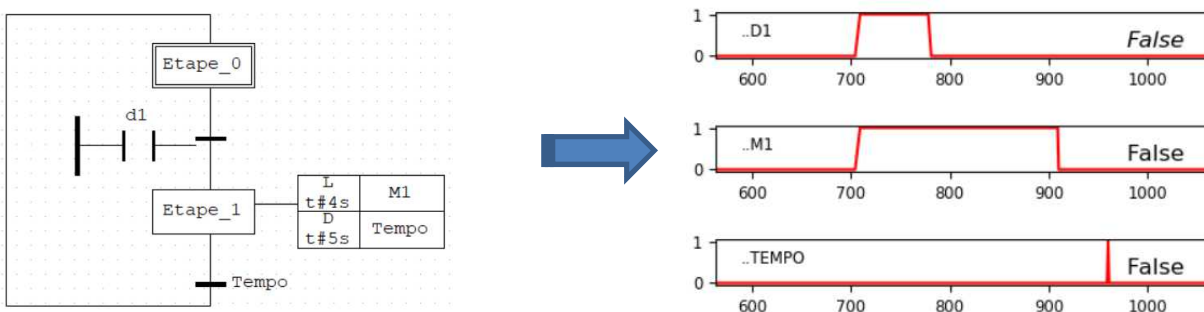
#### D : Time Delayed

La variable Tempo passe à 1 après 5s.



#### L : Time limited

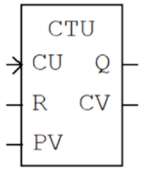
L'action M1 est réalisé pendant 4s dès que l'étape\_1 est active. L'étape\_1 sera restée active 5s grâce à tempo.



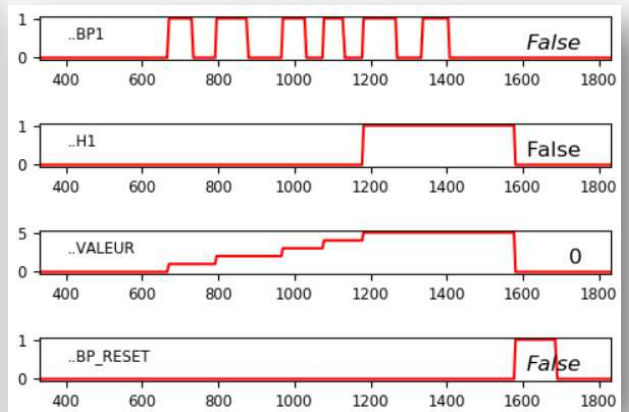
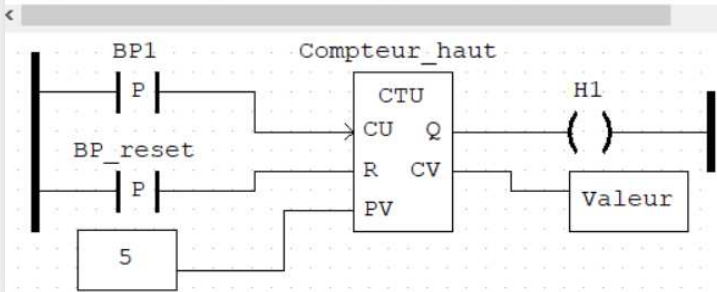
## ANNEXE 4 Compteurs (LADDER)

### Compteur\_haut

Le compteur peut être utilisé pour signaler qu'un compte a atteint une valeur maximale. Dans l'exemple ci-dessous, le compte (PV) est mis à 5 et chaque front montant de BP1 incrémente de 1 le compteur. La sortie Q passe à '1' lorsque PV est atteint. L'appui sur le BP\_reset remet tout à 0.

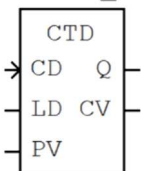


#	Nom	Classe	Type	Adresse	Val
1	BP_reset	Locale	BOOL	%IX0.1	
2	BP1	Locale	BOOL	%IX0.0	
3	H1	Locale	BOOL	%QX0.1	
4	Compteur_haut	Locale	CTU		
5	Valeur	Locale	INT	%MW0	

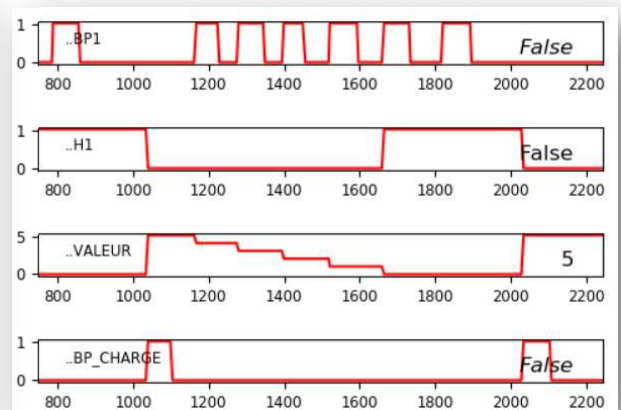
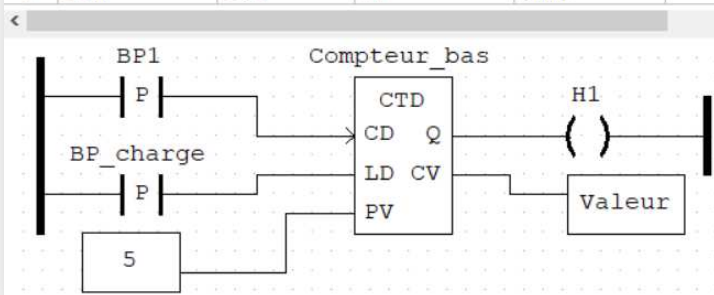


### Compteur\_bas

Le décompteur peut être utilisé pour signaler qu'un compte a atteint zéro, lors du décompte d'une valeur prédéfinie. Lorsque LD passe à 1, le compteur est initialisé à la valeur présente sur PV (ici 5). Ensuite, à chaque impulsion sur CD, cette valeur est décrétementée jusqu'à 0.



#	Nom	Classe	Type	Adresse	Val
1	BP_charge	Locale	BOOL	%IX0.1	
2	BP1	Locale	BOOL	%IX0.0	
3	H1	Locale	BOOL	%QX0.1	
4	Compteur_bas	Locale	CTD		
5	Valeur	Locale	INT	%MW0	



## ANNEXE 4'

### Compteurs (SFC)

Un exemple pour présenter cette fonctionnalité. On souhaite que le moteur M1 soit soumis cinq fois de suite au cycle suivant dès l'activation de d1 :

- activé 2s,
- en pause 1s.

Après ce cycle de 5 répétitions, le système s'arrête

