NEWEL 3

# FONCTIONS AUTOMATE PROGRAMMABLE (PLC)

AIDE

Digitel se réserve le droit de modifier les informations contenues dans ce document sans préavis.

Document non contractuel

*help-plc-function-fr.docx*
*04.02.2021*

Digitel SA

# *Table des matières*

## 1. DESCRIPTION DES FONCTIONS AUTOMATE PROGRAMMABLE (PLC)

Les fonctions « automate programmable » permettent d'automatiser certaines tâches de façon similaire à celle d'un automate programmable de type PLC (*Programmable Logic Controller*). Ces fonctions sont programmées dans TelesWin avec un langage simple à comprendre et à rédiger et elles sont stockées dans l'unité centrale DC58 et sont exécutées toutes les deux secondes. Ces fonctions permettent d'adapter facilement le fonctionnement de l'installation en fonction des besoins.

La solution Digitel permet d'utiliser toutes les entrées et sorties connectées sur le réseau, quel que soit leur usage (régulation des postes de froid, compresseurs, etc.) En comparaison, les automates programmables traditionnels ne peuvent agir que sur leurs propres entrées et sorties et non celles d'autres éléments du réseau.

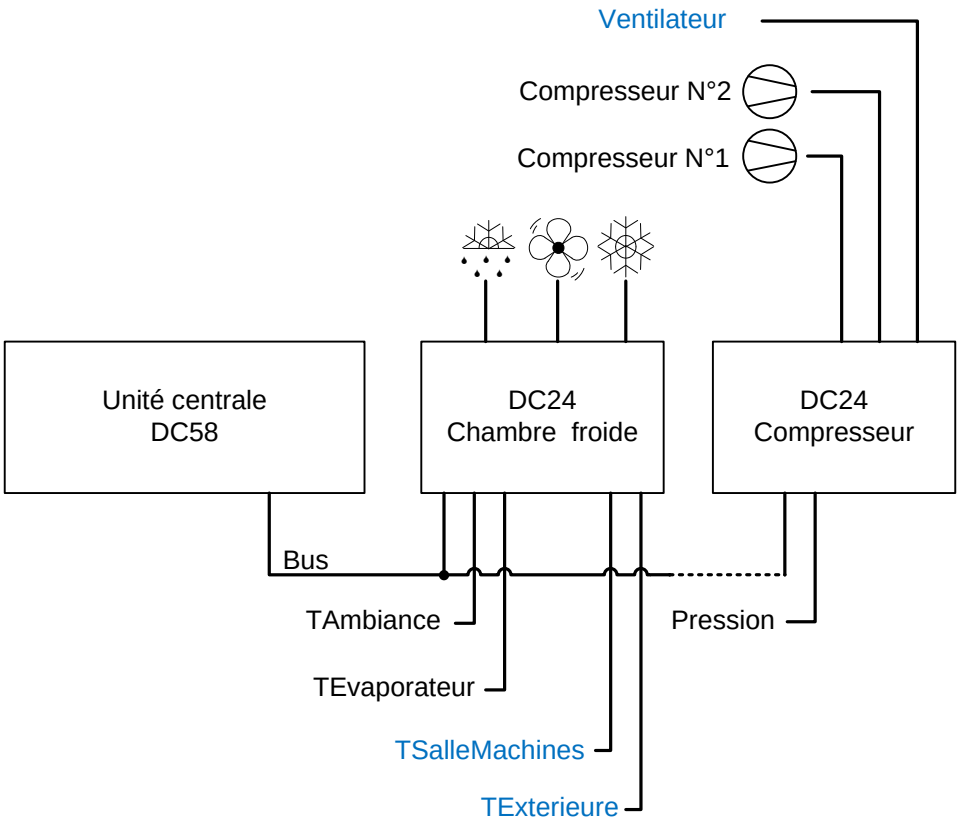
Les fonctions PLC permettent de créer de nouvelles fonctionnalités ou d'étendre les fonctionnalités existantes, même après la mise en service et pendant la marche normale de l'installation, et ceci sans intervenir sur les infrastructures comme le câblage. Une riche bibliothèque de fonctions préprogrammées permet de réaliser des tâches diverses et complexes.

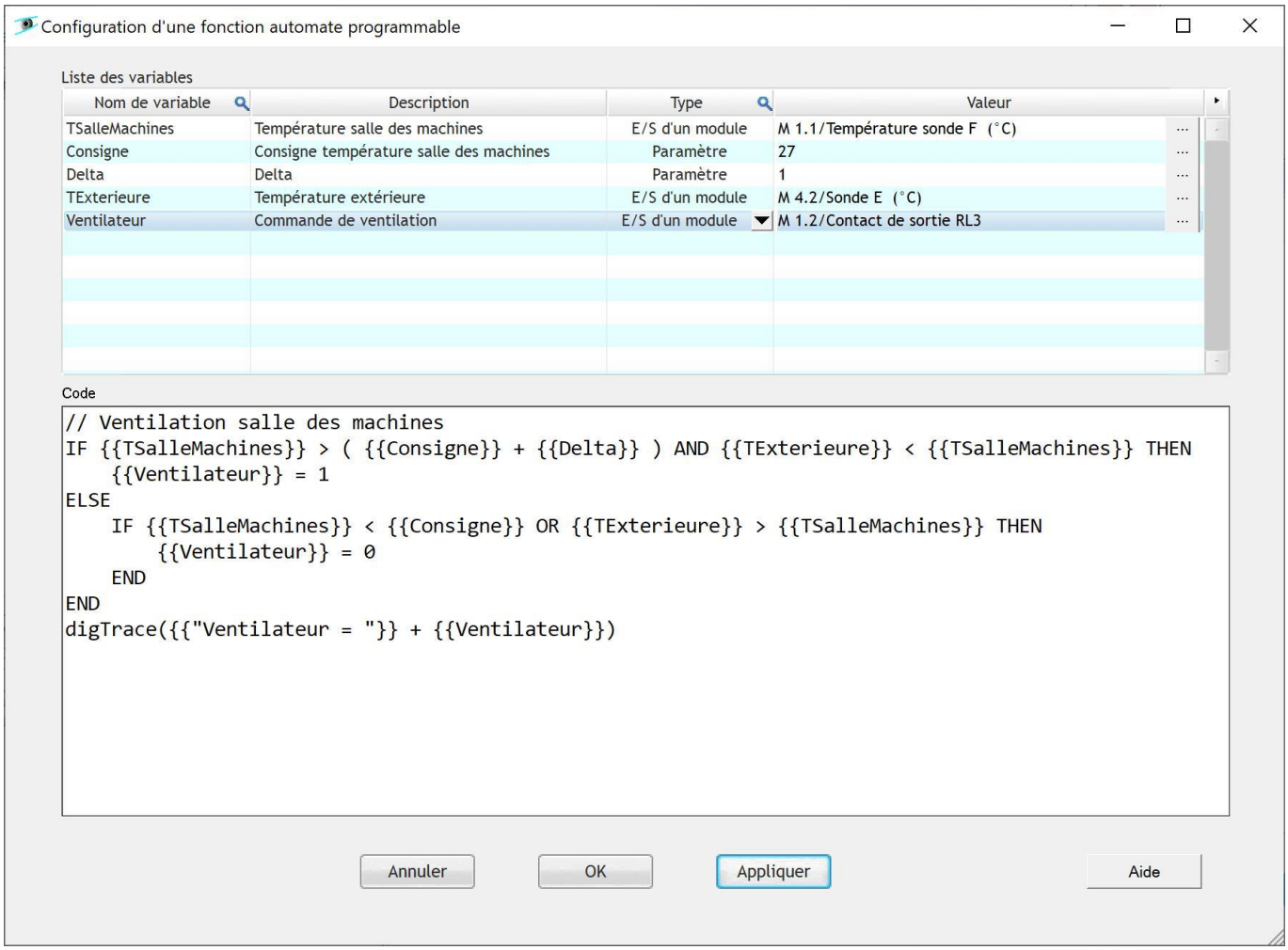
## 2. MATÉRIEL ET LOGICIELS COMPATIBLES

- TelesWin avec version du logiciel 22.73-19.45.1 ou supérieur
- Unité centrale DC58 avec version du logiciel 19451 ou supérieure.
- Régulateur DC24 D/DE/E/EE avec version du logiciel 19451 ou supérieure.

## 3. EXEMPLE D'UTILISATION

- Commande de la ventilation de la salle des machines en utilisant 2 entrées libres du régulateur de la chambre froide et une sortie libre du régulateur des compresseurs.




Configuration d'une fonction automate programmable
Liste des variables
Nom de variable
Description
Type
Valeur
TSalleMachines
Température salle des machines
E/S d'un module
M 1.1/Température sonde F (°C)
Consigne
Consigne température salle des machines
Paramètre
27
Delta
Delta
Paramètre
1
TExterieure
Température extérieure
E/S d'un module
M 4.2/Sonde E (°C)
Ventilateur
Commande de ventilation
E/S d'un module
M 1.2/Contact de sortie RL3
Code
// Ventilation salle des machines
IF {{TSalleMachines}} > ( {{Consigne}} + {{Delta}} ) AND {{TExterieure}} < {{TSalleMachines}} THEN
{{Ventilateur}} = 1
ELSE
IF {{TSalleMachines}} < {{Consigne}} OR {{TExterieure}} > {{TSalleMachines}} THEN
{{Ventilateur}} = 0
END
END
digTrace({{"Ventilateur = "}} + {{Ventilateur}})
Annuler
OK
Appliquer
Aide

# 4. VALEUR D'ENTRÉES ET DE SORTIES

- Les fonctions PLC permettent d'effectuer des actions sur des sorties (S) en fonction de valeurs d'entrées (E). Toutes les E/S du réseau peuvent être utilisées.

## 4.1. VALEURS D'ENTRÉES

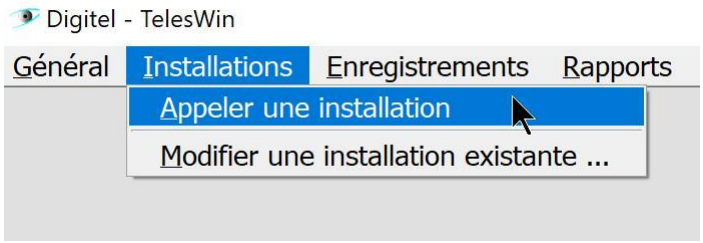
- Mesures de paramètres physiques (température, pression, hygrométrie, luminosité).
- Timers définis dans l'unité centrale.
- Paramètres fixes, déterminés par les programmeurs.
- Paramètres modifiables par les utilisateurs.

## 4.2. VALEURS DE SORTIES

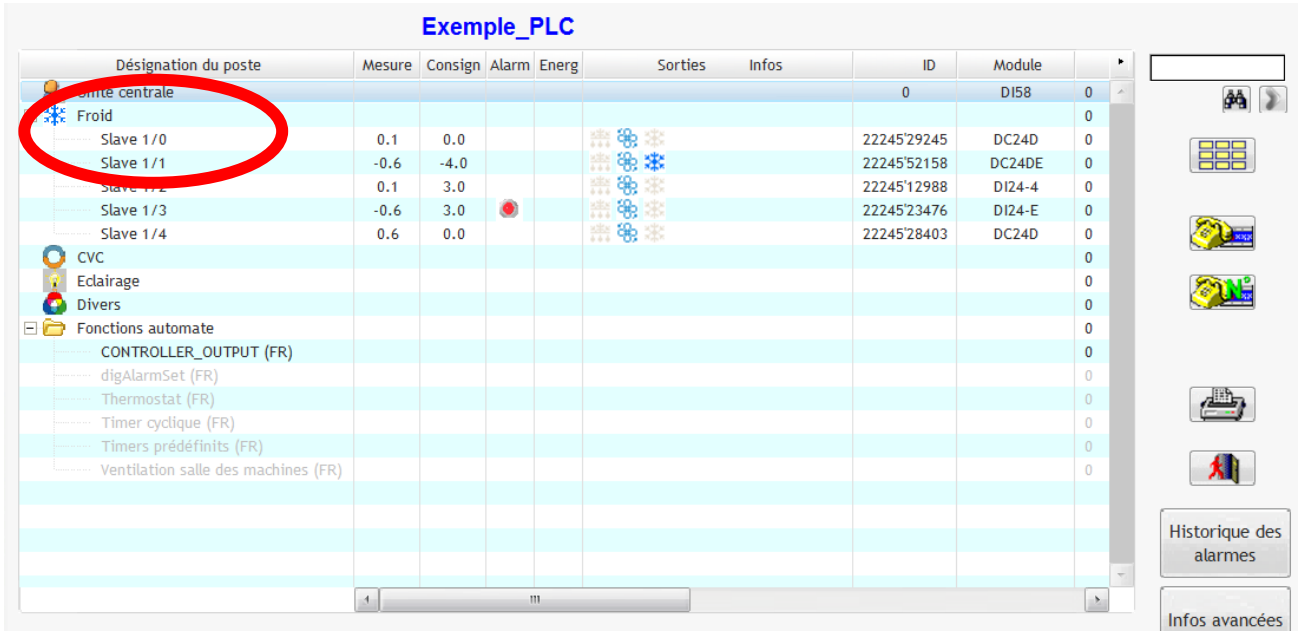
- Action sur les sorties digitales des satellites (enclenchement / déclenchement de ventilateurs, de compresseurs ou de relais).
- Commande de sorties analogiques.
- Activation d'alarmes définies dans l'unité centrale.
- Envoi de SMS.
- Envoi de mails.
- Affichage de messages et de valeurs dans une console sur TelesWin.

## 5. COMMENT CRÉER UNE FONCTION PLC

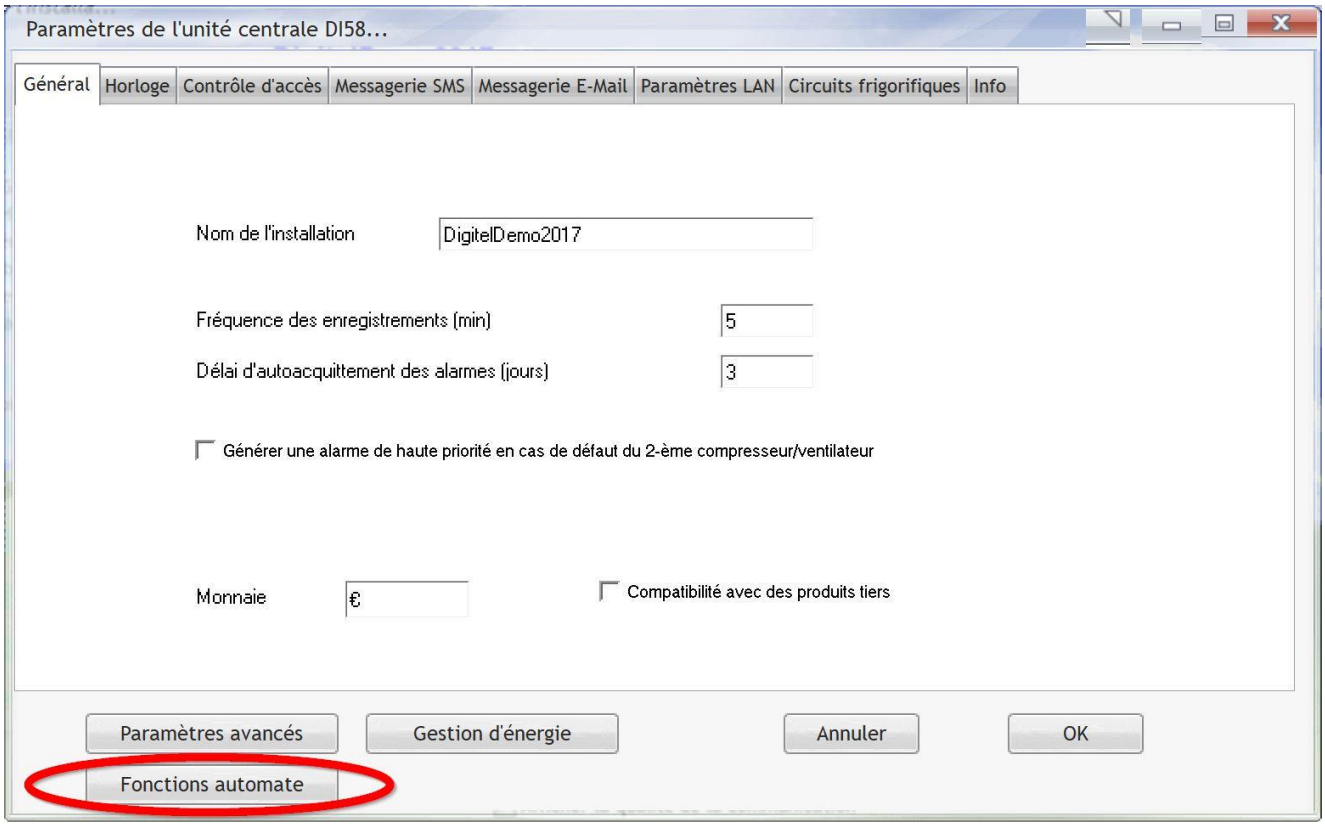
- Seule la personne ayant l'autorisation de « Configuration » de l'installation (voir sur unité centrale « Contrôle d'accès ») peut créer et modifier les fonctions PLC. Elle doit également disposer un dongle avec l'option « Configuration des installations » cochée.
- Ouvrez la fenêtre **`Configuration de l'installation`** (menu **`Installations / Appeler une installation`**).



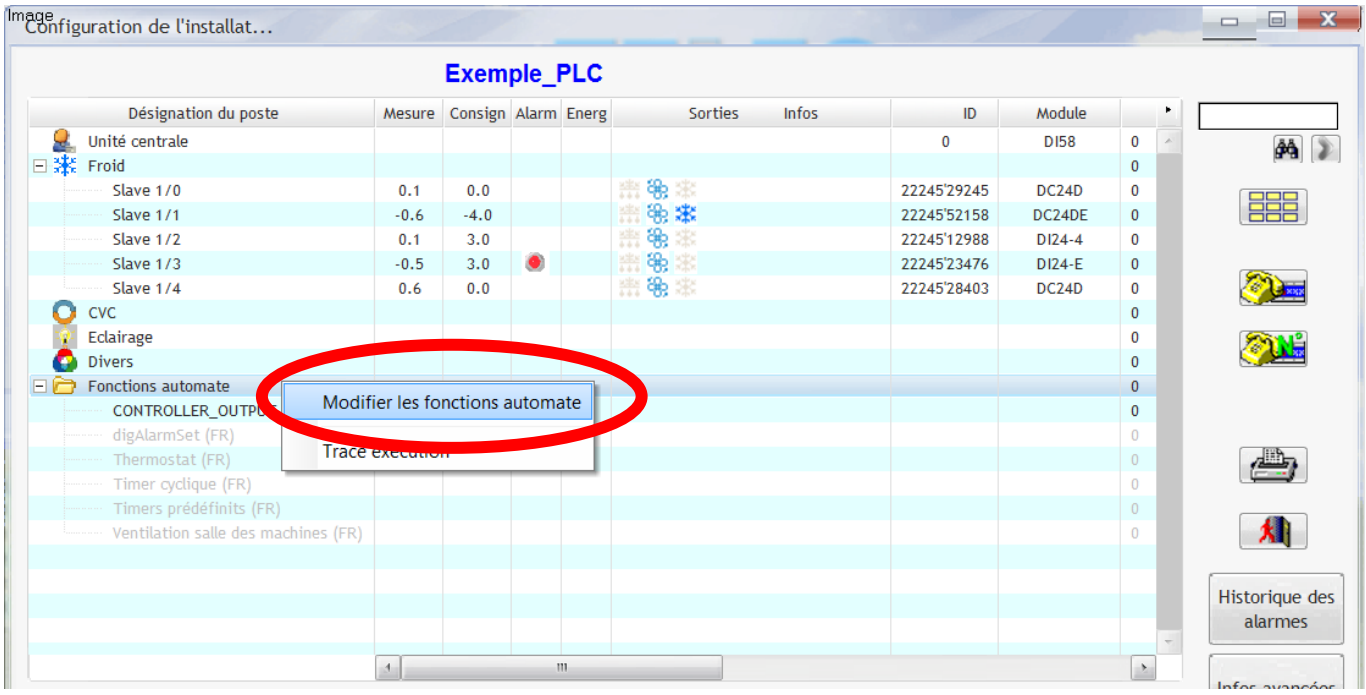

- Faites un double clic sur **`Unité centrale`**.



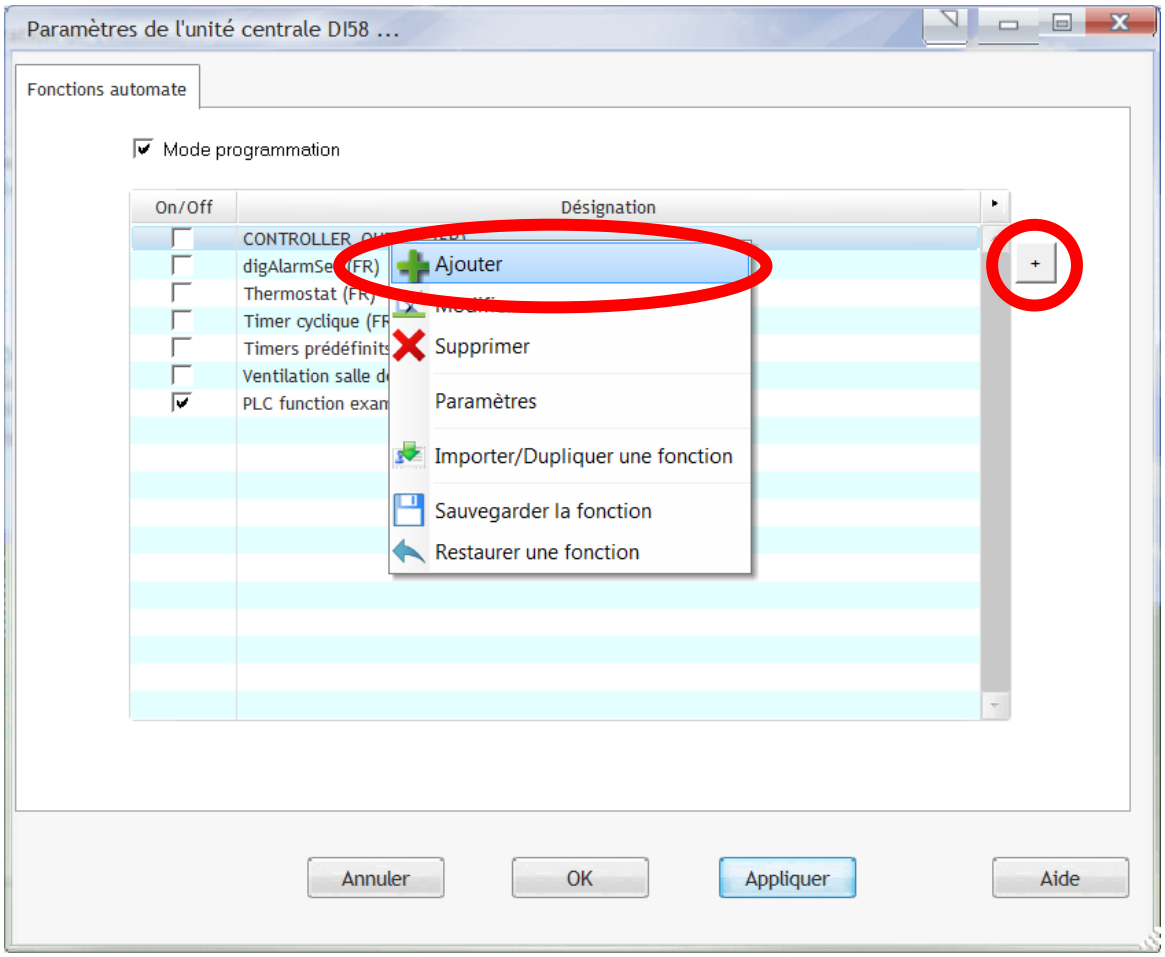
- La fenêtre **Paramètres de l'unité centrale** s'ouvre.
- Cliquez sur le bouton **Fonctions automate**.



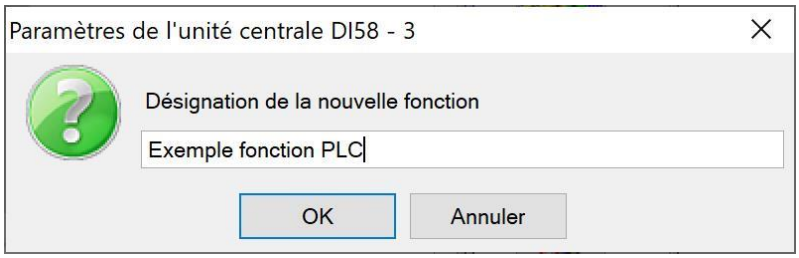

- Ou bien cliquez sur **Modifier les fonctions automate** du menu contextuel accessible en cliquant avec le bouton droit de la souris sur **Fonctions automate** ou sur une des fonctions automate présente.



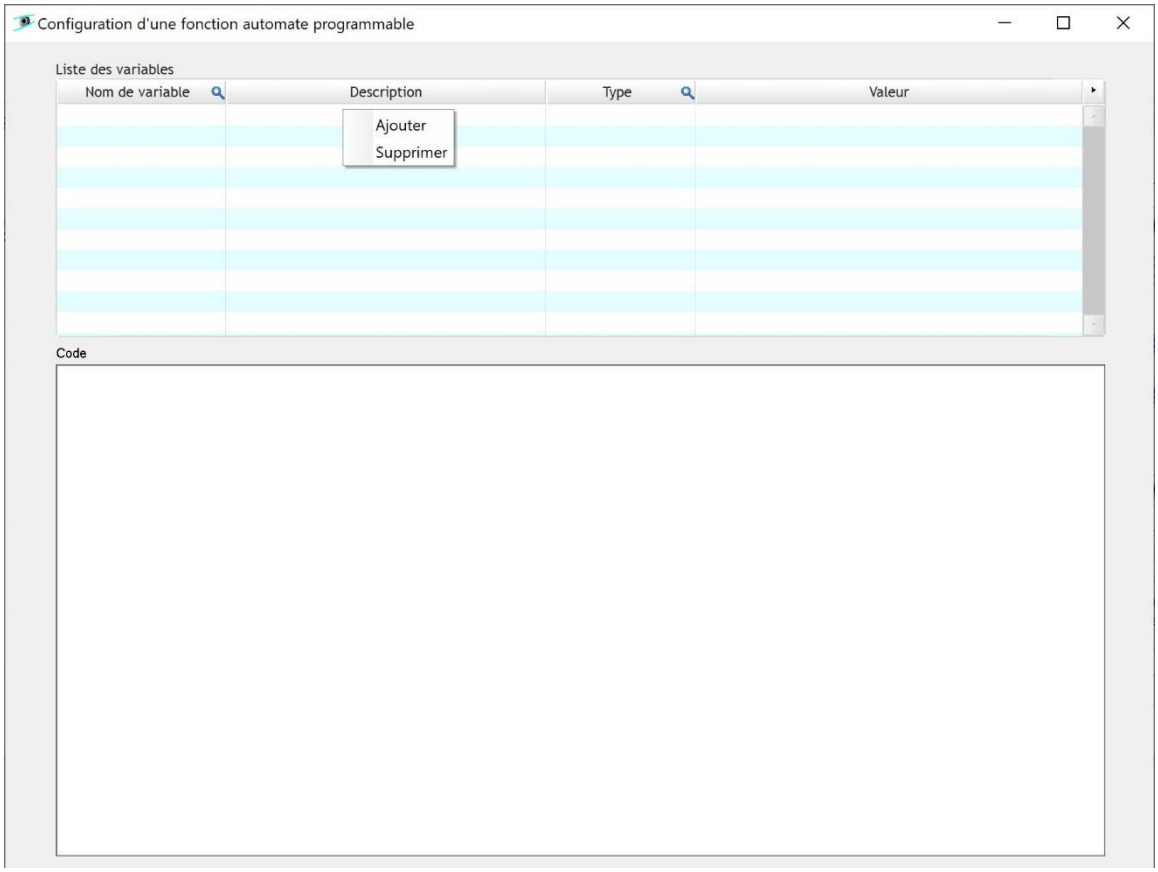
- La fenêtre **Paramètres de l'unité centrale -3** s'ouvre
- Cliquez sur le bouton **+** pour ajouter une fonction. Il est également possible d'utiliser la fonction **Ajouter** du menu contextuel accessible en cliquant dans le tableau avec le bouton droit de la souris.




- Une fenêtre surgissante s'ouvre.
- Entrez le nom de la nouvelle fonction PLC. Par exemple **Exemple fonction PLC**.
- Cliquez sur OK.

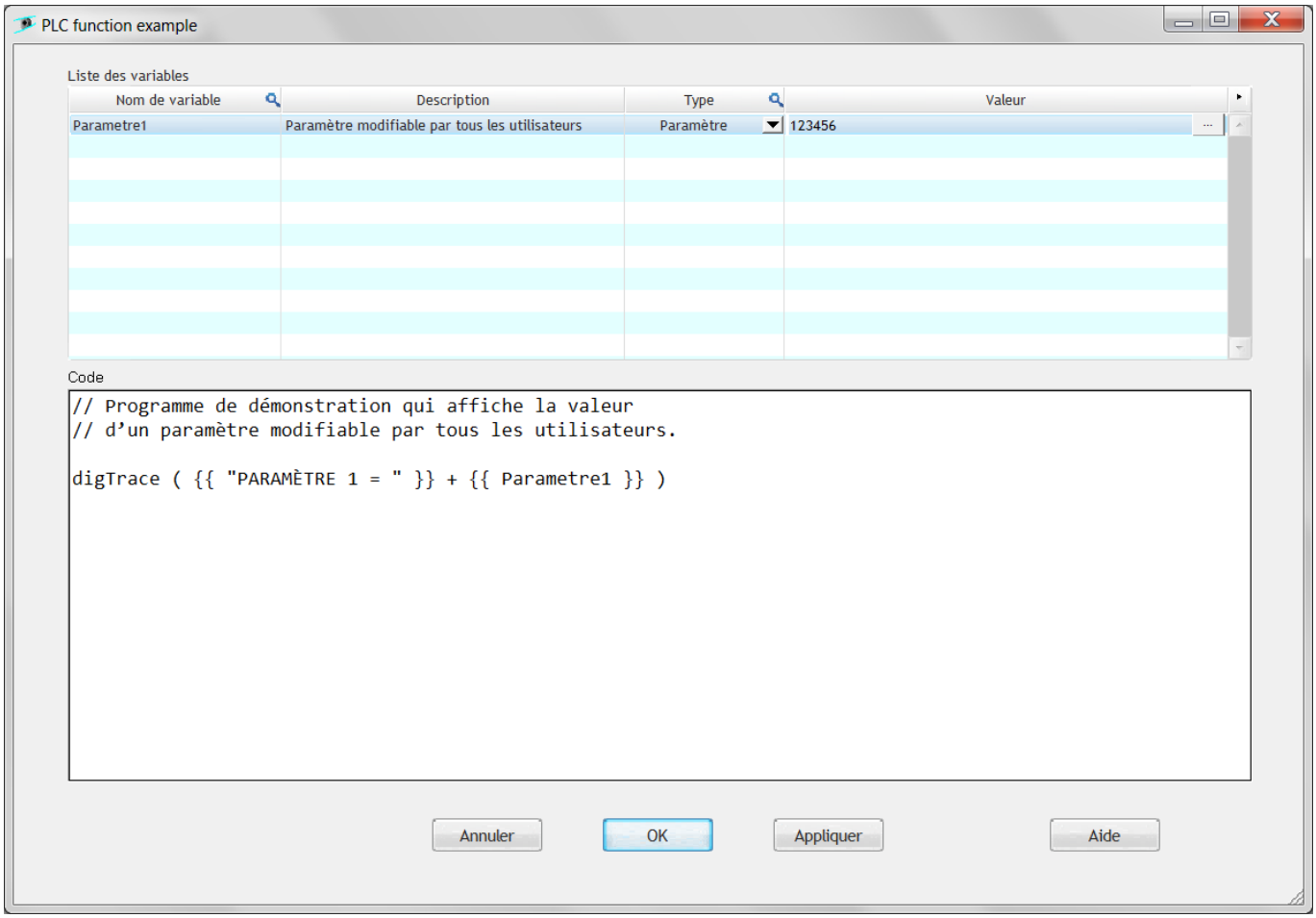


- La fenêtre **Configuration d'une fonction automate programmable** s'ouvre.
- Cette fenêtre permet de créer une fonction PLC.
- Le tableau **Liste des variables** permet de déclarer les variables qui seront utilisées par la fonction PLC.
- Le champ **Code** permet de taper le code.
- Faites un clic droit dans le tableau **Liste des variables** et cliquez sur **Ajouter**.

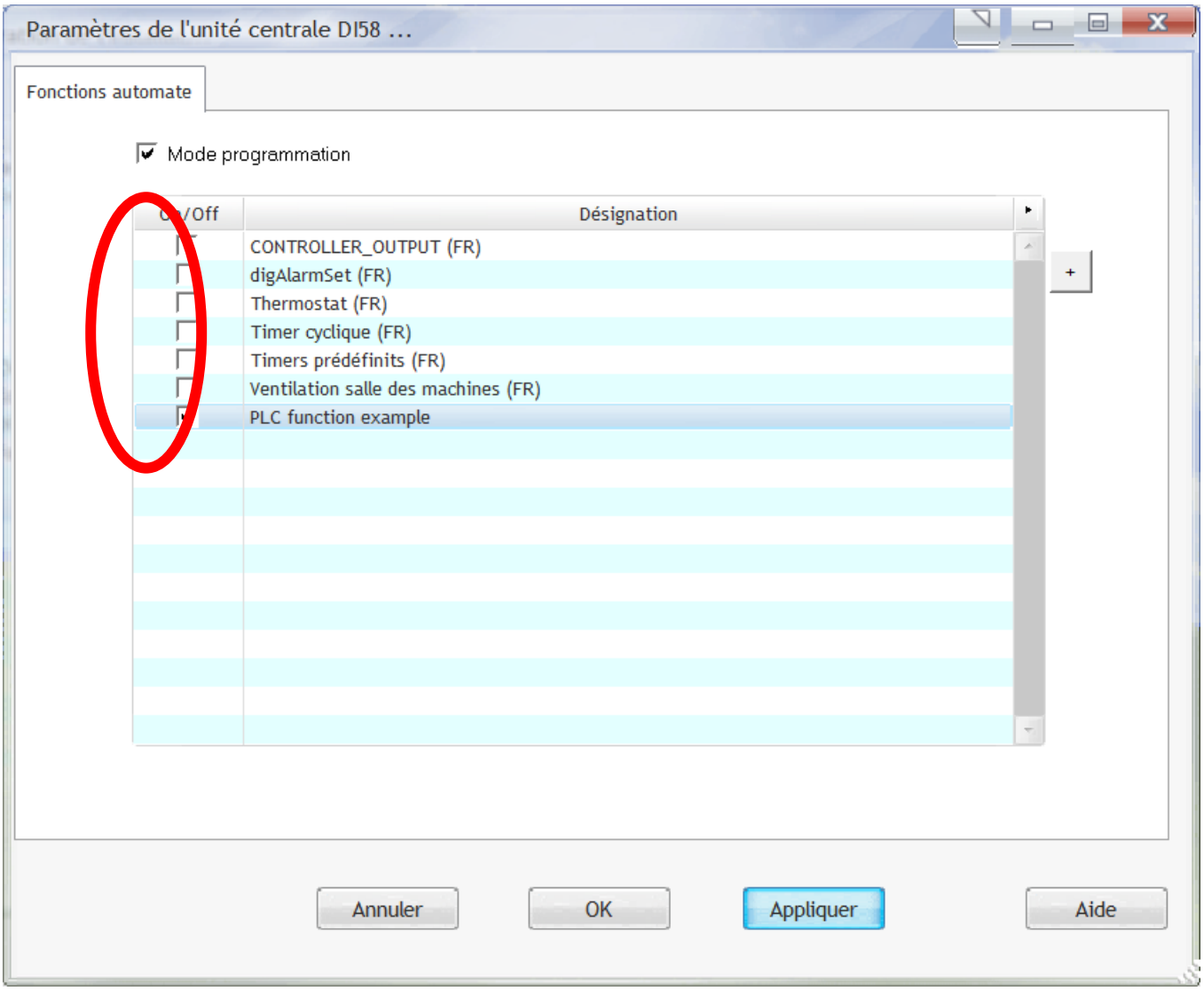


- Renseignez la table **`Liste des variables`** de façon à ce qu'elle corresponde à l'image ci-dessous.
- Pour renseigner le champ **`Valeur`**, il est nécessaire de cliquer sur le bouton « … » à droite du champ (voir image ci-dessous).
- Renseignez également le champ **`Code`**. Vous pouvez copier-coller l'exemple ci-dessous.

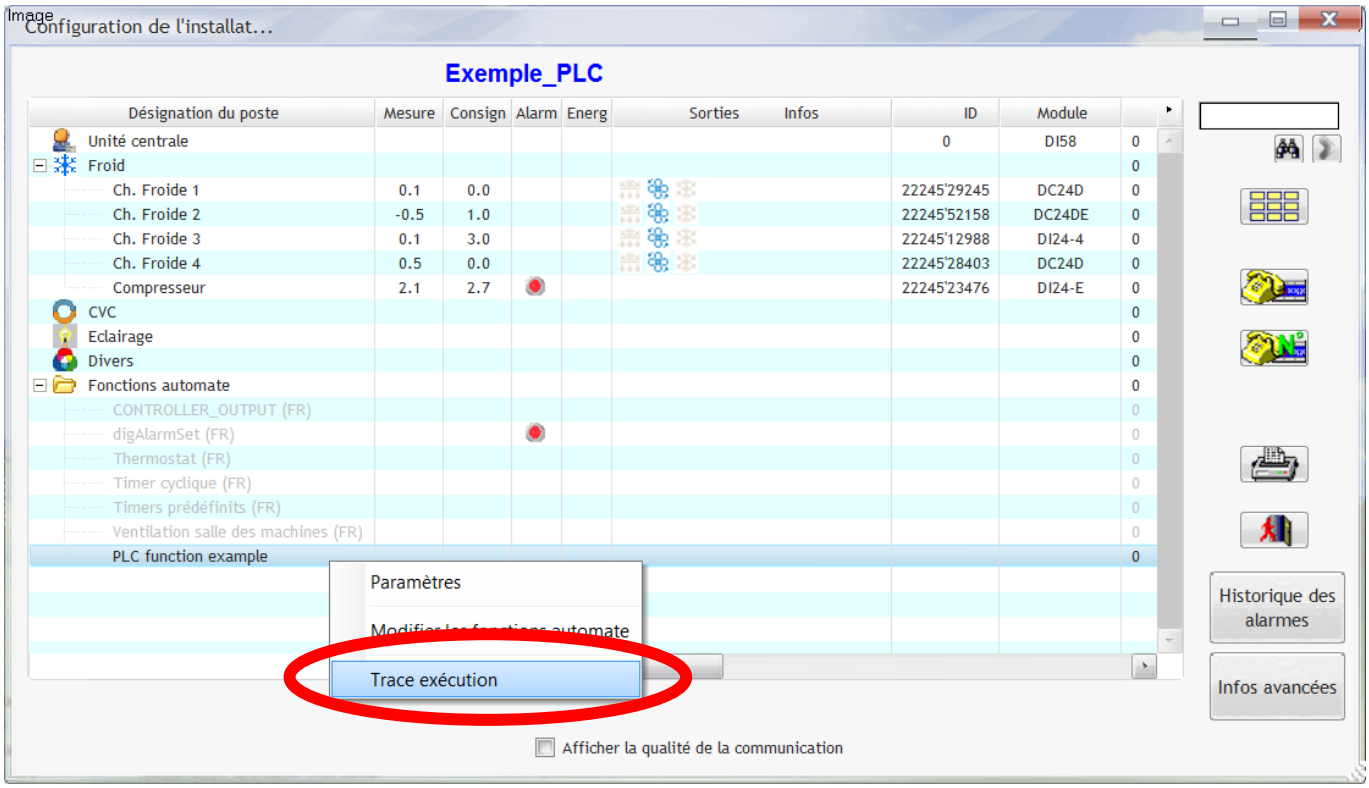
```
// Programme de démonstration qui affiche la valeur
// d'un paramètre modifiable par tous les utilisateurs.

digTrace ( {{ "PARAMÈTRE 1 = " }} + {{ Parametre1 }} )
```

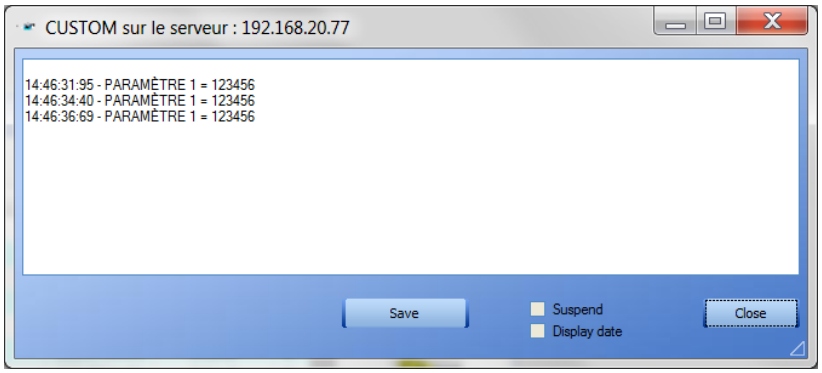


- Une fois les paramètres et le code renseignés, cliquez sur OK.
- Cliquez sur la case à cocher **ON/OFF** à gauche de la fonction que vous venez de créer pour qu'elle soit exécutée. La période d'exécution est de deux secondes environ.
-

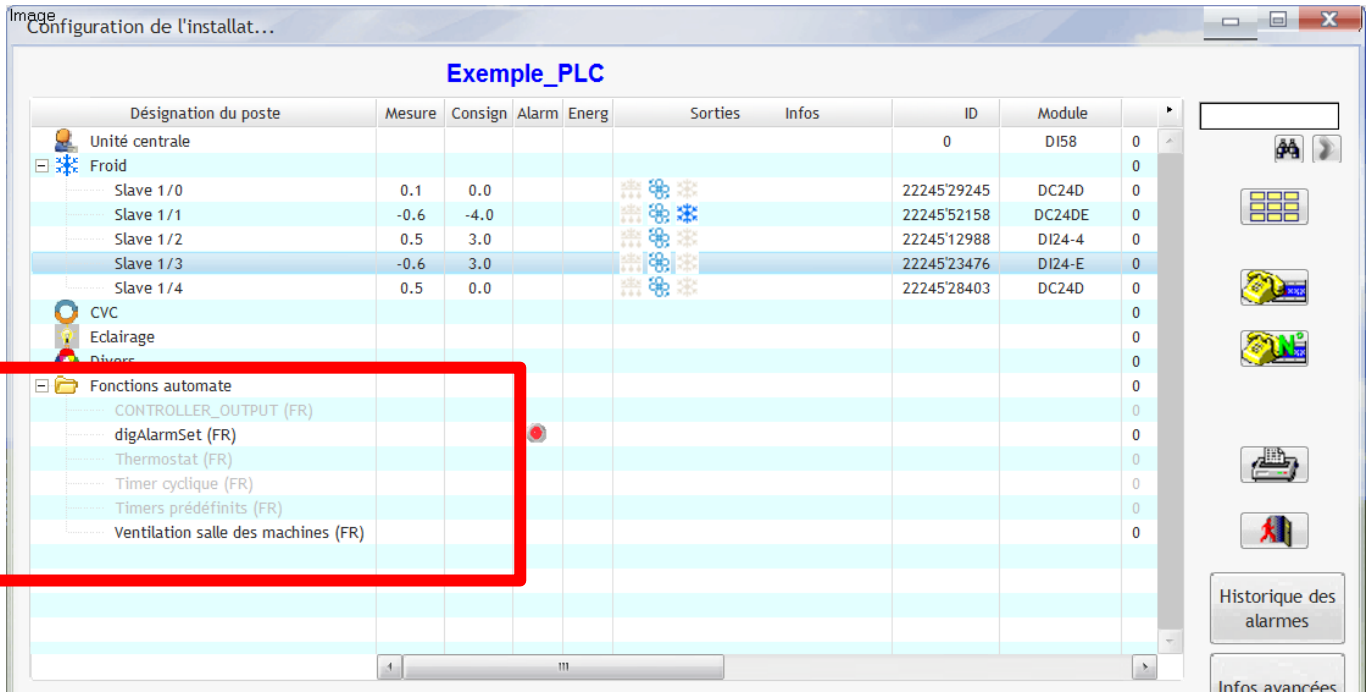


- Pour suivre le déroulement de l'exécution du programme, sélectionnez « Trace exécution » dans le menu contextuel des Fontions automate. Une fenêtre « Trace » s'ouvre et affiche le texte saisie dans la fonction digTrace.
- On peut voir que la fonction PLC affiche des textes qui pour l'instant ne changent pas au cours du temps.
- Nous verrons par la suite des exemples plus intéressants, avec des valeurs mesurées en entrée et des relais en sortie.



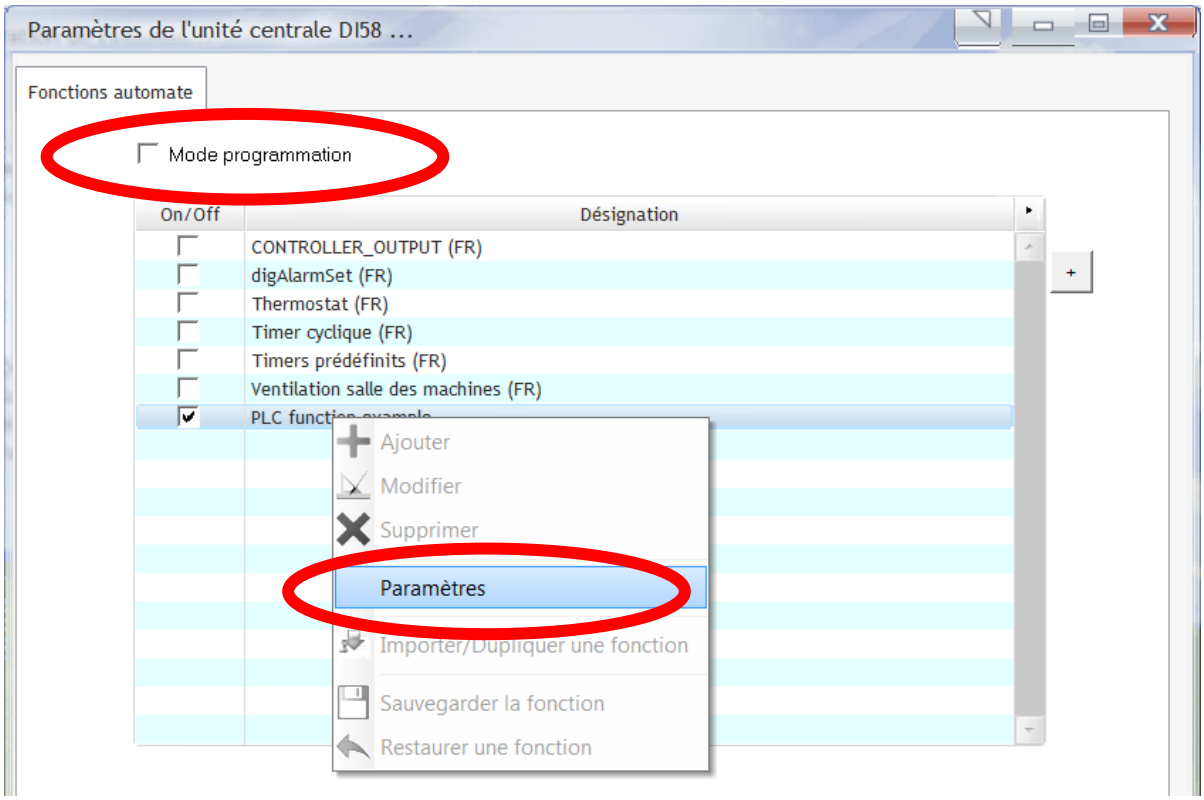
- Une fois la fonction créée et activée, il est possible de la voir dans la visualisation complète de l'installation. Les fonctions automates actives sont indiquées en noir et celles qui sont désactivées en gris. Le point rouge dans la colonne **Alarm** indique que cette fonction automate a généré une alarme.
-



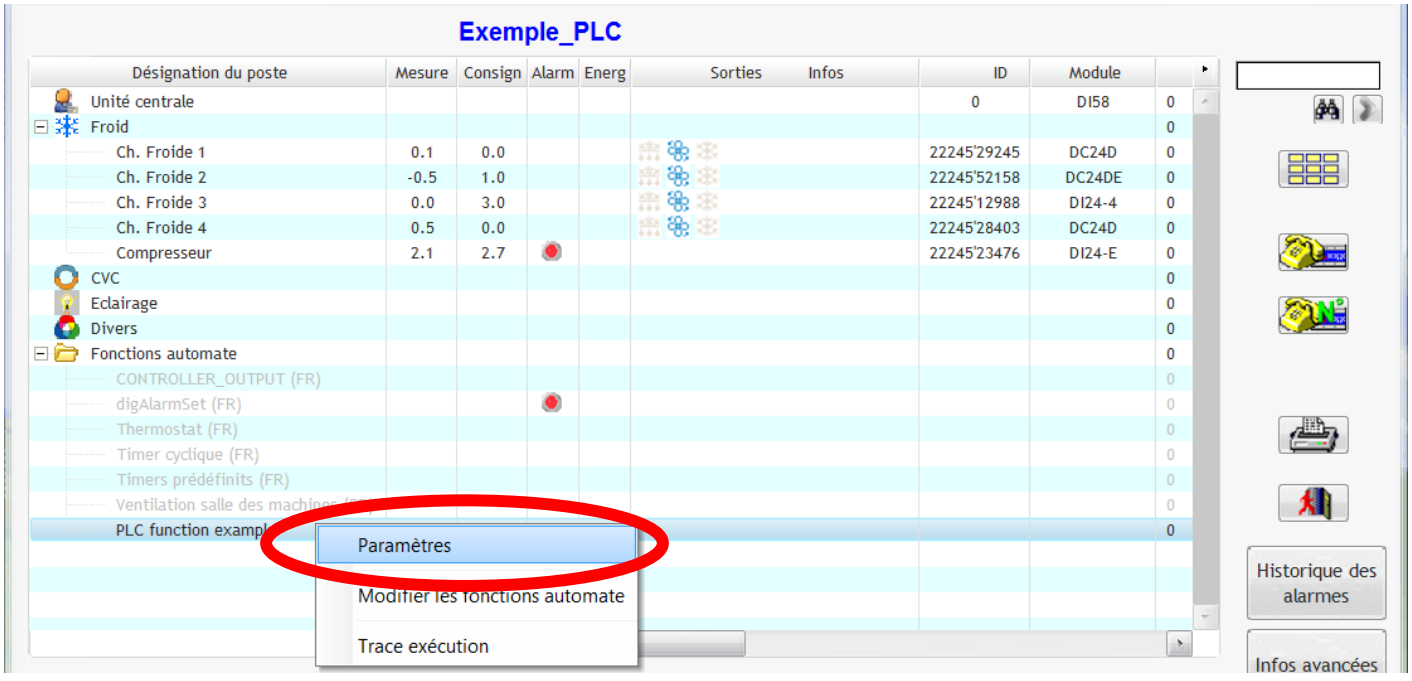

-

## 5.1. MODIFICATION DES PARAMÈTRES PAR UN UTILISATEUR

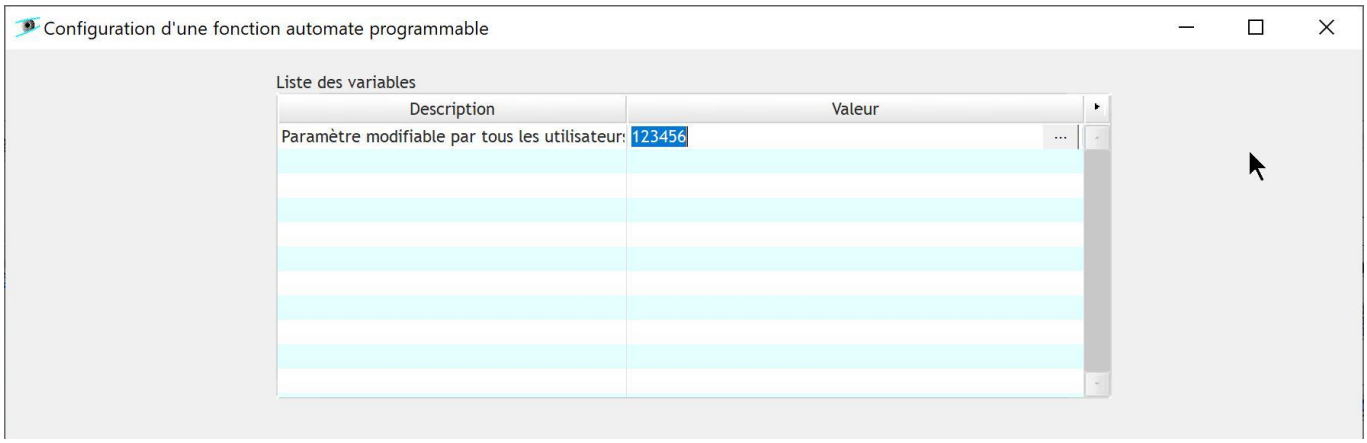
Les variables déclarées avec le type `paramètres` sont modifiables par un utilisateur qui n'a pas les droits de programmation.




Il est aussi possible de modifier les paramètres depuis la vue globale de l'installation. Il faut juste cliquer sur l'option **`Paramètres`** du menu contextuel, ou en faisant un double clic sur la fonction dont vous voulez modifier les paramètres.



Pour les utilisateurs sans droit de programmation, la case **Mode programmation** est toujours décochée. Lorsqu'ils cliquent avec le bouton droit de la souris pour afficher le menu contextuel, la seule option à laquelle ils ont accès est l'option **Paramètres** qui leur permet de modifier les paramètres.



# 6. RÉFÉRENCES DU LANGAGE DE PROGRAMMATION DES FONCTIONS PLC

Le langage de programmation que nous utilisons est basé sur le langage Windev Mobile développé par la société PC Soft. Le présent manuel décrit uniquement les opérateurs de base, les plus souvent utilisées. Ils sont suffisants pour créer des fonctions simples. Pour créer des fonctions avancées, nous vous conseillons d'étudier le manuel de Windev Mobile qui contient la description détaillée de tous les opérateurs et les fonctions de ce langage.

***Les noms des variables des fonctions PLC sont sensibles à la casse, c'est-à-dire qu'il faut respecter les majuscules et les minuscules quand on utilise une variable.***

***Les noms de variables ne doivent contenir que des lettres, des chiffres et le caractère souligné (_). Ils ne peuvent pas contenir d'espaces.***

## 6.1. STRUCTURES

### 6.1.1. Commentaires

Des commentaires peuvent être ajoutés au code. Ils commencent par une double barre oblique `//` et finissent à la fin de la ligne. Les commentaires sont utiles pour documenter le code et ne sont pas exécutés.

### 6.1.2. Assignation de variables

Les variables doivent être déclarées dans le tableau **`Liste de variables`**. Elles doivent être entourées de deux accolades `{{…}}` lors de leur utilisation dans le code, que ce soit en lecture ou en écriture. Les types de variables possibles sont :

`E/S d'un module` : Pour assigner la valeur d'une entrée ou d'une sortie (E/S) à une variable du code PLC.

`Variable interne` : Pour déclarer une nouvelle variable qui sera définie et accessible dans le code PLC.

`Paramètre` : Pour assigner une valeur de paramètre qui pourra être modifiée par un utilisateur qui n'a pas les droits de programmeur.

`Alarme` : Pour générer une alarme.

`Timer` : Pour utiliser la valeur de retour d'un timer déjà créé dans l'unité centrale (**`Paramètres de l'unité centrale / Paramètres avancés / Timers`**)

### 6.1.3. Les opérateurs

#### CONCATÉNATION DES CHAÎNES DE CARACTÈRES

Il est possible de concaténer des chaînes de caractères entre-elles et également de concaténer des chaînes et des variables. Le résultat peut ensuite être utilisé pour afficher des messages informatifs. L'opérateur de concaténation de chaînes est le signe `+`.

#### EXEMPLE

Le code ci-dessous affichera dans la fenêtre **`Trace`** la chaîne de caractères `Temp = xx°C`, où xx est la valeur de la température, par exemple `Temp = -18°C`. La sonde `Temp1` doit être définie dans la liste des variables.

Les chaînes de caractères constantes doivent être entourées des doubles accolades et des guillemets `{{"Temp = "}}`, alors que les variables n'ont besoin que des accolades, mais pas des guillemets `{{Temp1}}`. Les

accolades peuvent optionnellement être suivies et précédées d'un espace, mais elles ne peuvent pas être séparées par un espace.

Les paramètres des fonctions sont entourés par des parenthèses qui peuvent optionnellement être précédées et suivies d'un espace.

```
digTrace({{"Temp = "}} + {{Temp1}} + {{"°C"}})
```

## ADDITION

L'addition utilise aussi le signe +, comme la concaténation de chaînes. Si on veut utiliser l'addition et la concaténation dans la même fonction, il faut réaliser les opérations sur deux lignes distinctes.

```
{{Val}} = {{Temp1}} + 1.5                 // Le signe + réalise une addition
digTrace ( {{"Temp1 = "}} + {{ Temp1 }} ) // Le signe + réalise une concaténation
digTrace ( {{"Val = "}} + {{ Val }} )     // Le signe + réalise une concaténation
```

## SOUSTRACTION

```
{{Val}} = {{Temp1}} - 1.5
```

## MULTIPLICATION

```
{{Val}} = {{Temp1}} * 1.5
```

## DIVISION

```
{{Val}} = {{Temp1}} / 1.5
```

## VALEUR ABSOLUE

```
{{Val}} = Abs ( {{Temp1}} )
```

## LES OPÉRATEURS DE COMPARAISON

Les opérateurs de comparaison renvoient des valeurs booléennes, c'est-à-dire Vrai ou Faux.

```
{{Val}} = {{Temp1}} >  -20 // Plus grand
{{Val}} = {{Temp1}} >= -20 // Plus grand ou égal

{{Val}} = {{Temp1}} <  -20 // Plus petit
{{Val}} = {{Temp1}} <= -20 // Plus petit ou égal

{{Val}} = {{Temp1}} =  -20 // Exactement égal
{{Val}} = {{Temp1}} <> -20 // Différent
```

## LES OPÉRATEURS LOGIQUES

```
{{Val}} = ( {{Temp1}} > -20 ) AND ( {{Temp2}} > -20 ) // ET logique
{{Val}} = ( {{Temp1}} > -20 ) OR  ( {{Temp2}} > -20 ) // OU logique
{{Val}} = NOT ( {{Temp2}} > -20 )                     // NÉGATION logique
```

## LES VALEURS LOGIQUES

```
{{Val}} = True  // Toutes les valeurs numériques différentes de 0
                // sont aussi évaluées comme Vraies
{{Val}} = False // La valeur numérique 0 est aussi évaluée comme Fausse
```

### MODULO

Le modulo est le reste de la division entière. Par exemple, 7 modulo 3 = 1, car 7 / 3 = 2 reste 1, ou autrement dit 3 × 2 + 1 = 7. Le modulo peut être utilisé pour créer facilement des signaux cadencés. Par exemple, si on veut générer un signal ayant une période de 10 minutes et qui est à 1 pendant 2 minutes et donc à 0 pendant 8 minutes :

```
{{Val}} = modulo (minuteFrom2000, 10) < 2
```

#### 6.1.4. IF..THEN..ELSE..

L'instruction conditionnelle `IF` permet de choisir d'exécuter une action en fonction d'une condition.

### EXEMPLE

```
IF {{Temp1}} > 0 THEN
    // Action 1
ELSE IF {{Temp1}} > -20 THEN
    // Action 2
ELSE
    // Action 3
END
```

## 6.2. FONCTIONS PRÉDÉFINIES

### 6.2.1. digAlarmSet

Déclenche une alarme lorsqu'une condition est vraie pendant un certain temps. Le message de l'alarme doit être configuré dans le champ **`Description`** de la liste des variables.

#### SYNTAXE

```
digAlarmSet(nom, condition, retard)
```

`<nom>` : Chaîne de caractères

Référence de l'alarme (Nom de la variable dans le tableau **`Liste des variables`**).

`<condition>` : Booléen

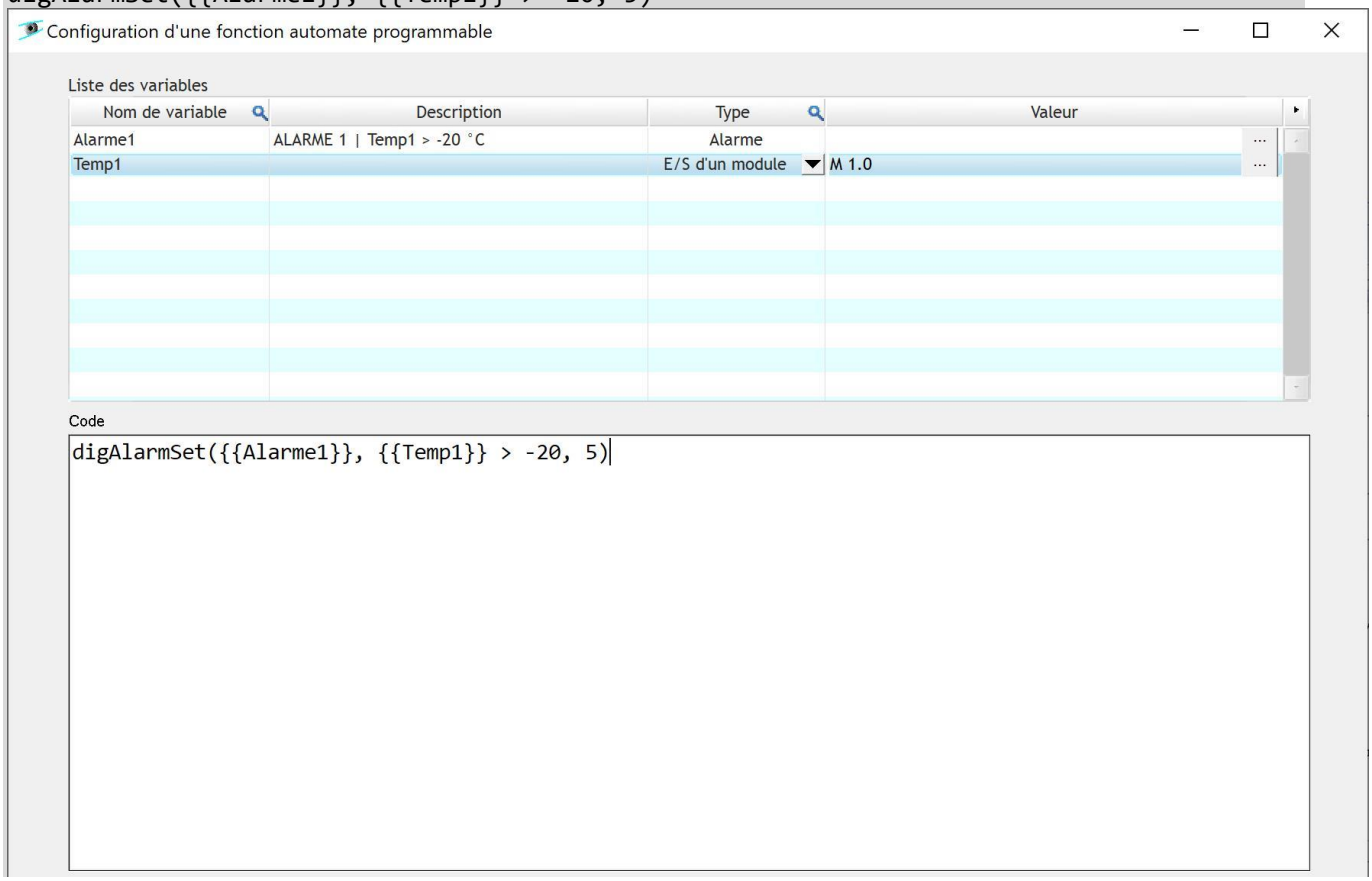
Condition qui peut prendre la valeur Faux (valeur égale à 0) ou Vrai (valeur différente de 0). Cette condition est typiquement écrite sous forme d'un test, par exemple : `{{Temp1}} > -20`.

`<retard>` : Entier ou réel

Délai en minutes avant le déclenchement de l'alarme.

#### EXEMPLE

```
digAlarmSet({{Alarme1}}, {{Temp1}} > -20, 5)
```



### 6.2.2. digAlarmGetState

Renvoie l'état d'alarme actuel d'une alarme donnée, 0 = pas d'alarme en cours, 1 = alarme en cours.

#### SYNTAXE

```
digAlarmGetState(nom)
```

`<nom>` : Chaîne de caractères

Référence de l'alarme (Nom de la variable dans le tableau **`Liste des variables`**).

#### EXEMPLE

```
IF digAlarmGetState({{Alarme1}}) THEN
    // Action à exécuter
END
```

### 6.2.3. digMessageSend

Envoie un message par mail ou par SMS.

Attention ! Les paramètres de la « Messagerie Email » dans l'unité centrale doivent être configurés correctement pour que l'envoi des e-mails fonctionne. De même, les paramètres de la « Messagerie SMS » doivent être configurés et le modem GSM doit être connecté pour que l'envoi des SMS fonctionne.

#### SYNTAXE

```
digMessageSend(destination, message, type)
```

`<destination>` : chaîne de caractères

adresse e-mail du destinataire lorsque `type = "EMail"` ou numéro de téléphone lorsque `type = "SMS"`.

`<message>` : chaîne de caractères

message à envoyer

```
<type>
```

Type du message. `type = "EMail"` ou `"SMS"`

#### EXEMPLE

```
{{Message1}} = {{ "Message à envoyer" }}
digMessageSend({{"test@example.com"}}, {{Message1}}, {{"EMail"}})
digMessageSend({{"+411234567"}}, {{Message1}}, {{"SMS"}})
```

### 6.2.4. digTrace

Envoie une chaîne de caractères dans la fenêtre **Trace**.

#### SYNTAXE

```
digTrace(message)
```

`< message >` : chaîne de caractères

Le message à afficher dans la fenêtre **Trace**.

#### EXEMPLE

```
digTrace({{"Température = "}} + {{TempLue}} + {{" °C. Chauffage = "}} + {{Chauffage}})
```

### 6.2.5. digSetpointShift

Utilisable avec les versions du firmware des régulateurs égale ou supérieure à 21011 et avec les régulateurs de type DC24D, DC24DE, DC24E, DC24EE fonctionnant en modes 0, 1 ou 2.

Décale la consigne par rapport à la consigne programmée dans les paramètres du régulateur. La consigne sera décalée pendant 10 minutes depuis la dernière commande `digSetpointShift()`. Après ce délai la consigne programmée dans les paramètres du régulateur sera automatiquement restaurée. La consigne d'origine peut être restaurée immédiatement avec l'envoie de la même commande avec la constante `CONTROLLER_SETPOINT` comme paramètre `rShift`.

#### SYNTAXE

```
digSetpointShift(unitID, rShift)
```

`<unitID>` : Entier ou réel

Identificateur du régulateur visible dans la colonne `unitID` du tableau « Configuration de l'installation »

`<rShift>` : Entier ou réel

Décalage (positif ou négatif) par rapport à la consigne programmée dans le régulateur.

#### EXEMPLE

```
digSetpointShift(7, 5)
```

Voir chapitre 6.4.2 pour un exemple supplémentaire.

### 6.2.6. digSetpointSetTR

Utilisable avec les versions du firmware des régulateurs égale ou supérieure à 21011 et avec les versions du software DC58 21011 ou plus récentes. Utilisable uniquement avec les régulateurs de type DC24TR fonctionnant en modes 0 ou 1.

Change la consigne du régulateur pour la valeur passée dans le paramètre `rNewSetpoint`. La consigne sera modifiée pendant 10 minutes depuis la dernière commande `digSetpointSetTR()`. Après ce délai la consigne programmée dans les paramètres du régulateur sera automatiquement restaurée. La consigne d'origine peut être restaurée immédiatement avec l'envoie de la même commande avec la constante `CONTROLLER_SETPOINT` comme paramètre `rNewSetpoint`.

#### SYNTAXE

```
digSetpointSetTR(unitID, rNewSetpoint)
```

`<unitID>` : Entier ou réel

Identificateur du régulateur visible dans la colonne `unitID` du tableau « Configuration de l'installation »

`<rNewSetpoint>` : Entier ou réel

Nouvelle consigne du régulateur (consigne du gascooler en mode 0 ou consigne HP en mode 1).

#### EXEMPLE

```
IF {{TempEauChaude}} > {{ValeurLimite}} THEN
    digSetpointSetTR(7, 100)
ELSE
    digSetpointSetTR(7, CONTROLLER_SETPOINT)
END
```

### 6.2.7. digSetpointSetTR_MP

Utilisable avec les versions du firmware des régulateurs égale ou supérieure à 21011 et avec les versions du software DC58 21011 ou plus récentes. Utilisable uniquement avec les régulateurs de type DC24TR fonctionnant en mode 1.

Change la consigne moyenne pression (MP) du régulateur pour la valeur passée dans le paramètre `rNewSetpoint`. La consigne sera modifiée pendant 10 minutes depuis la dernière commande `digSetpointSetTR_MP()`. Après ce délai la consigne programmée dans les paramètres du régulateur sera automatiquement restaurée. La consigne d'origine peut être restaurée immédiatement avec l'envoie de la même commande avec la constante `CONTROLLER_SETPOINT` comme paramètre `rNewSetpoint`.

#### SYNTAXE

```
digSetpointSetTR_MP(unitID, rNewSetpoint)
```

`<unitID>` : Entier ou réel

Identificateur du régulateur visible dans la colonne `unitID` du tableau « Configuration de l'installation »

`<rNewSetpoint>` : Entier ou réel

Nouvelle consigne du régulateur (consigne MP en mode 1).

#### EXEMPLE

```
digSetpointSetTR_MP(7, 37)
```

## 6.3. VARIABLES SYSTÈME UTILISABLES DANS LES FONCTIONS D'AUTOMATE

`minuteFrom2000`

minutes depuis 1.01.2000 00:00

`secondFrom2000`

secondes depuis 1.01.2000 00:00

`hourFrom2000`

heures depuis 1.01.2000 00:00

`minuteOfDay`

minutes depuis minuit (00:00:00)

`secondOfDay`

secondes depuis minuit (00:00:00)

`hourOfDay`

heures depuis minuit (00:00:00)

## 6.4. CONSTANTE SYSTÈME UTILISABLE DANS LES FONCTIONS D'AUTOMATE

### 6.4.1. CONTROLLER_OUTPUT

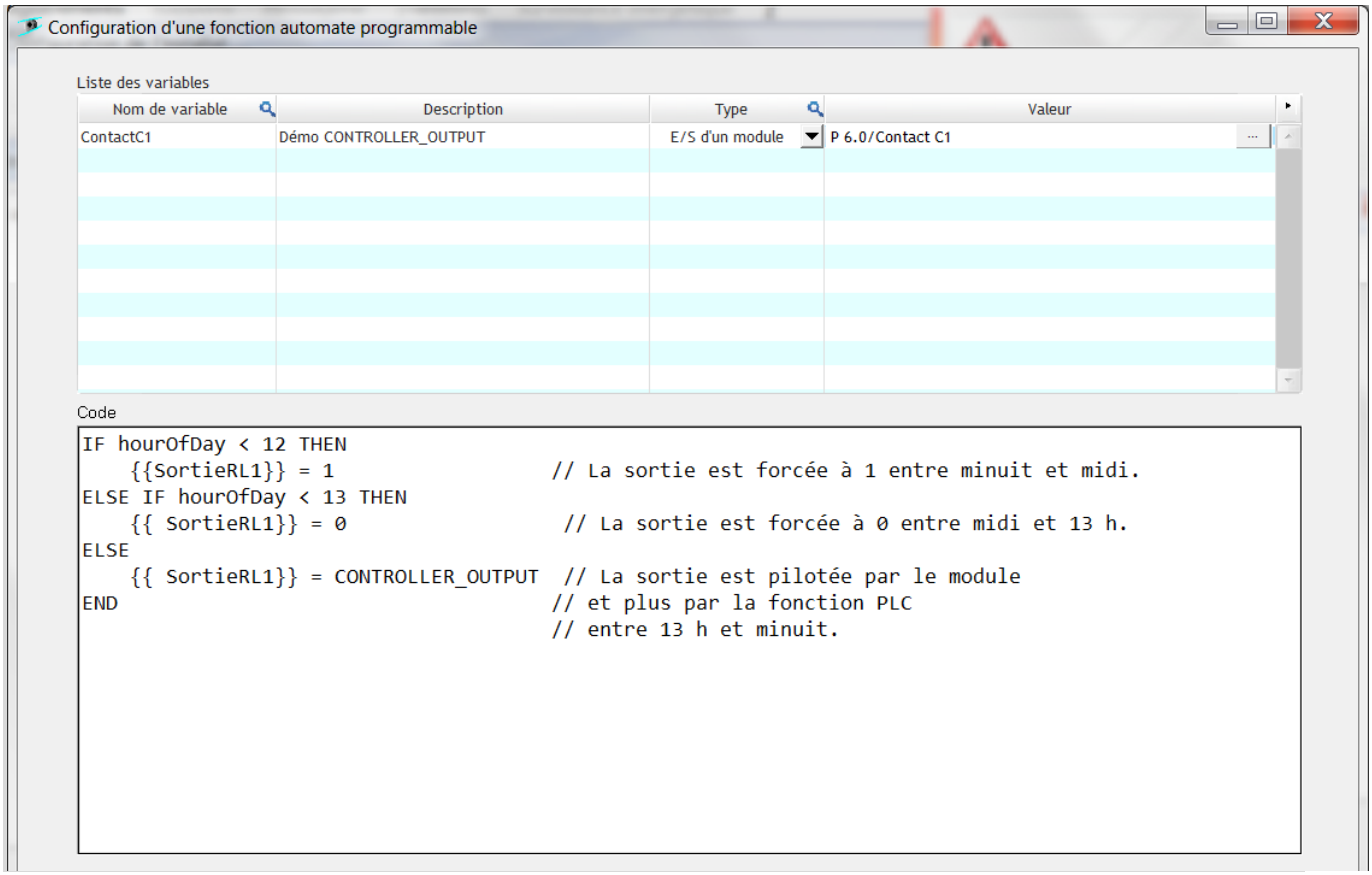
La constante système `CONTROLLER_OUTPUT` permet de rendre la main à un module après avoir forcé la sortie à 1 ou à 0 pour les sorties digitales ou entre 0 et 100% pour les sorties analogiques.

Expliqué d'une autre manière, on peut assigner 3 valeurs différentes à une sortie d'un module :

- `1` : La sortie est forcée à 1.
- `0` : La sortie est forcée à 0.
- `CONTROLLER_OUTPUT` : La sortie est pilotée par le module et plus par la fonction PLC.

#### EXEMPLE

```
IF hourOfDay < 12 THEN
    {{SortieRL1}} = 1                   // La sortie est forcée à 1 entre minuit et midi.
ELSE IF hourOfDay < 13 THEN
    {{ SortieRL1}} = 0                   // La sortie est forcée à 0 entre midi et 13 h.
ELSE
    {{ SortieRL1}} = CONTROLLER_OUTPUT  // La sortie est pilotée par le module
END                                     // et plus par la fonction PLC
                                        // entre 13 h et minuit.
```



### 6.4.2. CONTROLLER_SETPOINT

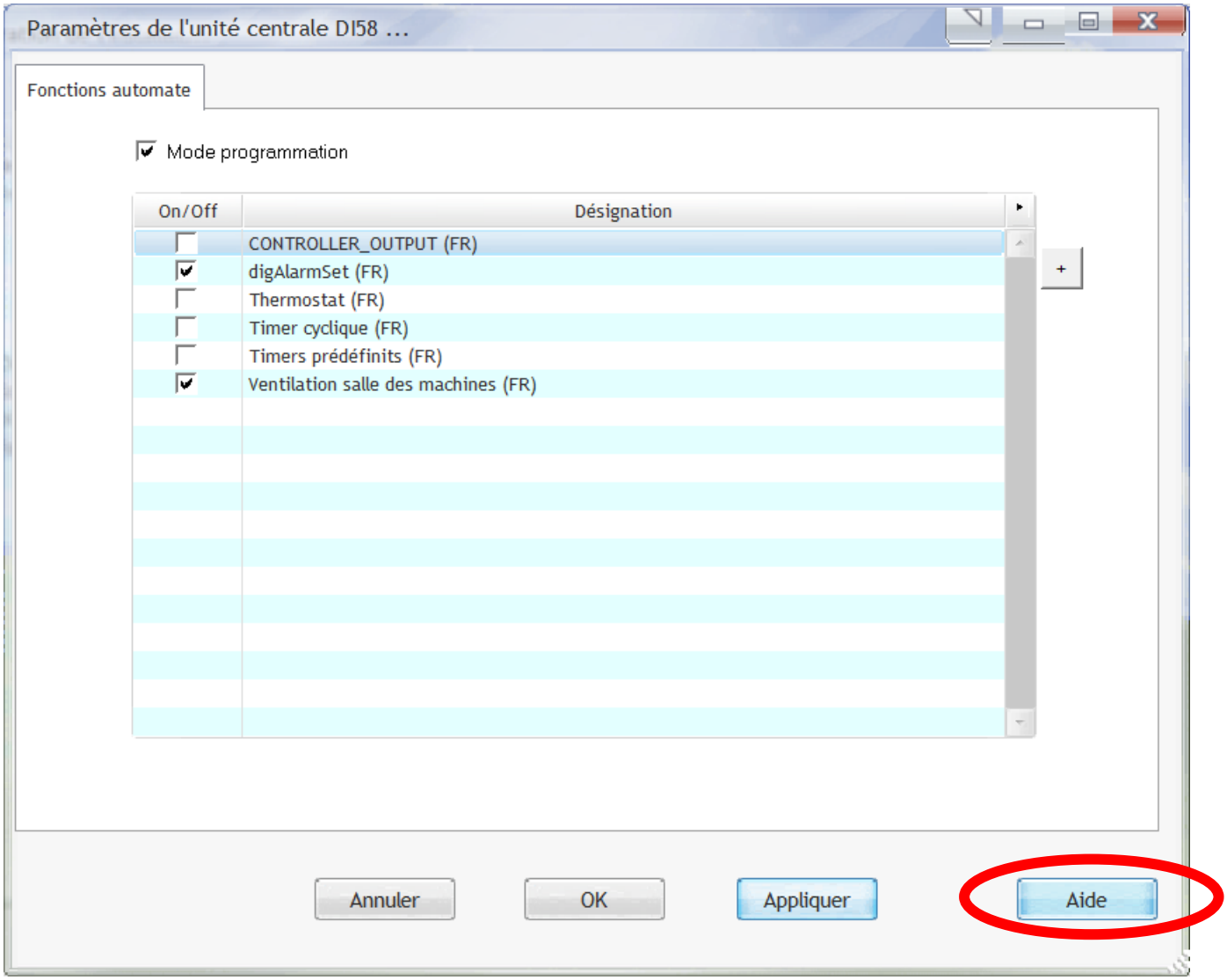
La constante système `CONTROLLER_SETPOINT` permet de rendre la main à un module après avoir forcé le décalage de la consigne (`digSetpointShift()`) ou une nouvelle consigne (`digSetpointSetTR()` ou `digSetpointSetTR_MP()`).

#### EXAMPLE

```
IF hourOfDay > 18 OR hourOfDay < 8 THEN
    digSetpointShift(10, 5)            // La consigne est décalé de +5° entre 18h et 8h.
ELSE
    digSetpointShift(10, CONTROLLER_SETPOINT)     // La sortie est pilotée par le module
END
```

## 7. BOUTON AIDE

Le bouton aide ouvre la documentation de la fonction automate (PLC).

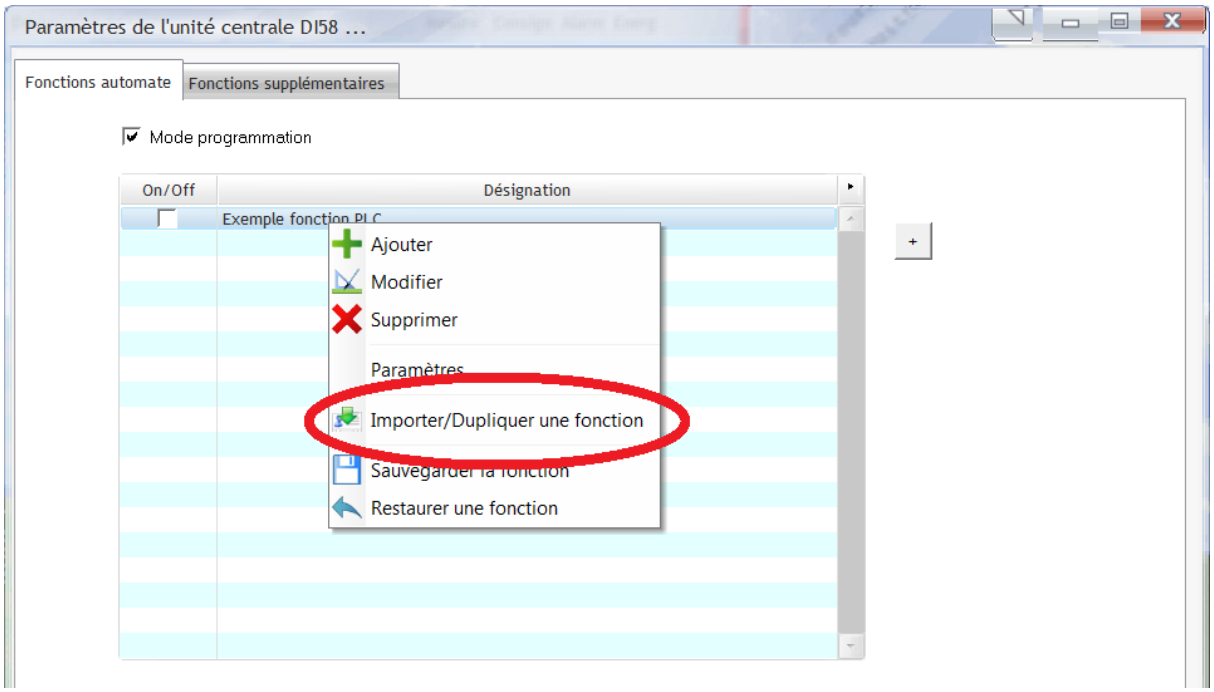


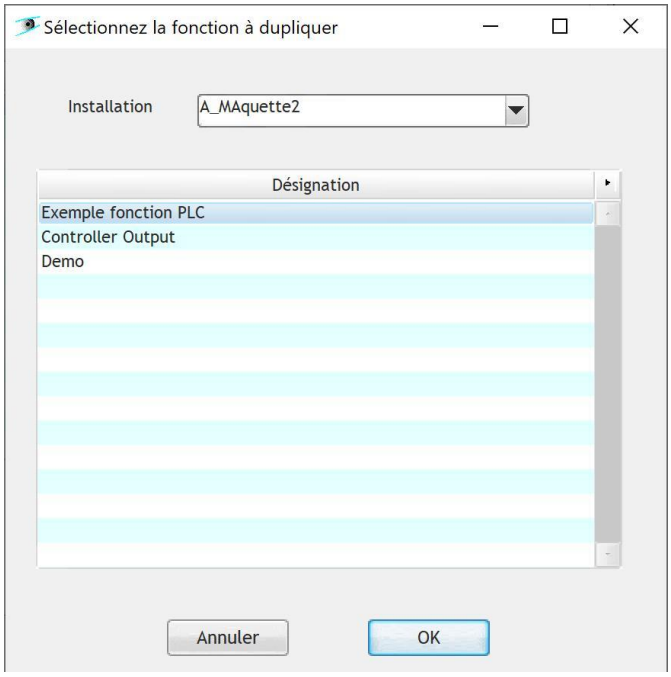
# 8. IMPORTATION ET DUPLICATION DE FONCTIONS

Il est possible d'importer et de dupliquer des fonctions depuis une autre installation. Pour faire cela, il faut ouvrir le menu contextuel en cliquant sur le bouton de droite de la souris et choisir la fonction **`Importer/Dupliquer une fonction existante`**. Dans la fenêtre qui apparaît, on peut sélectionner l'installation sur laquelle se trouve la fonction. Si cette installation est différente de celle sur laquelle on est connecté, la fonction sera importée. Si au contraire, c'est la même installation, alors la fonction sera dupliquée.

Les liens des variables du type « E/S d'un module » avec les entrées et les sorties des modules (colonne « Valeur ») ne seront pas copiés, car la nouvelle fonction utilisera généralement d'autres entrées et sorties. Ces liens doivent être sélectionnés manuellement en cliquant sur le bouton « … » de la colonne « Valeur ».

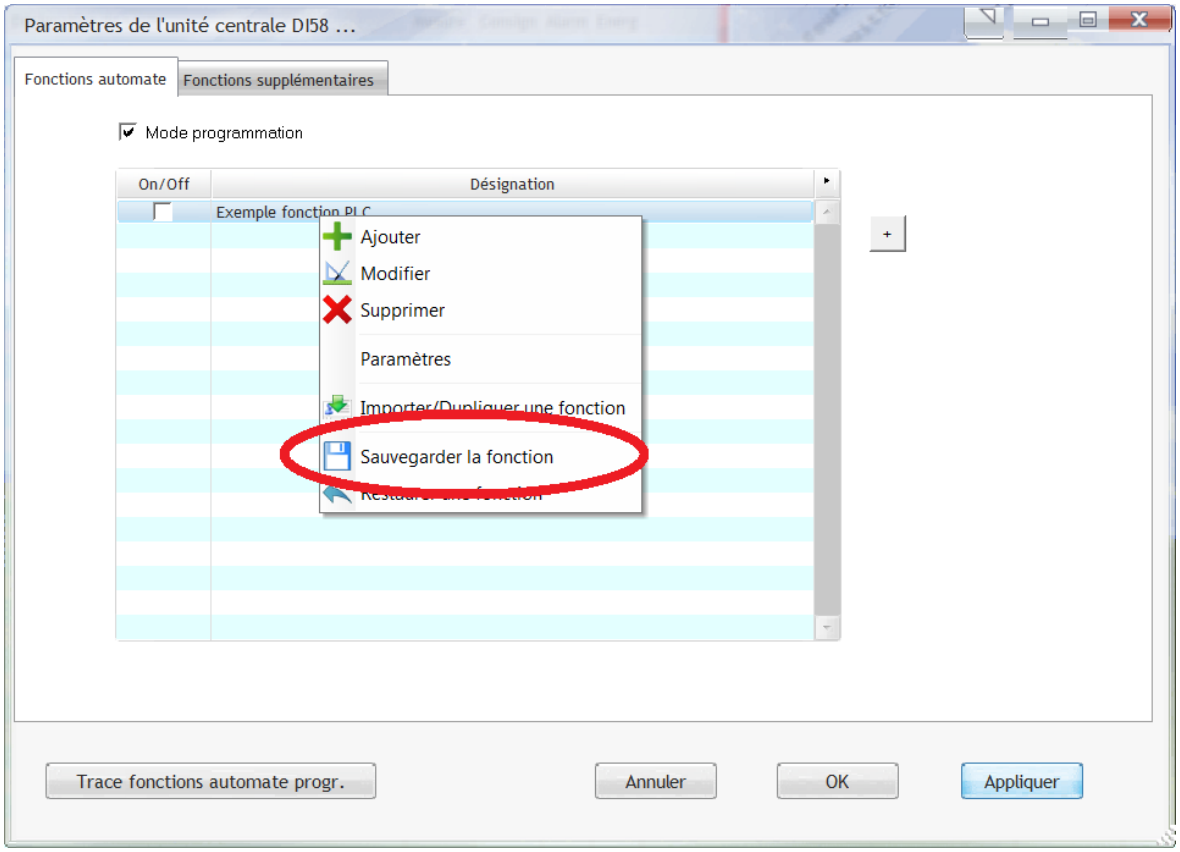
Il est possible de renommer les fonctions en cliquant deux fois sur leur nom.

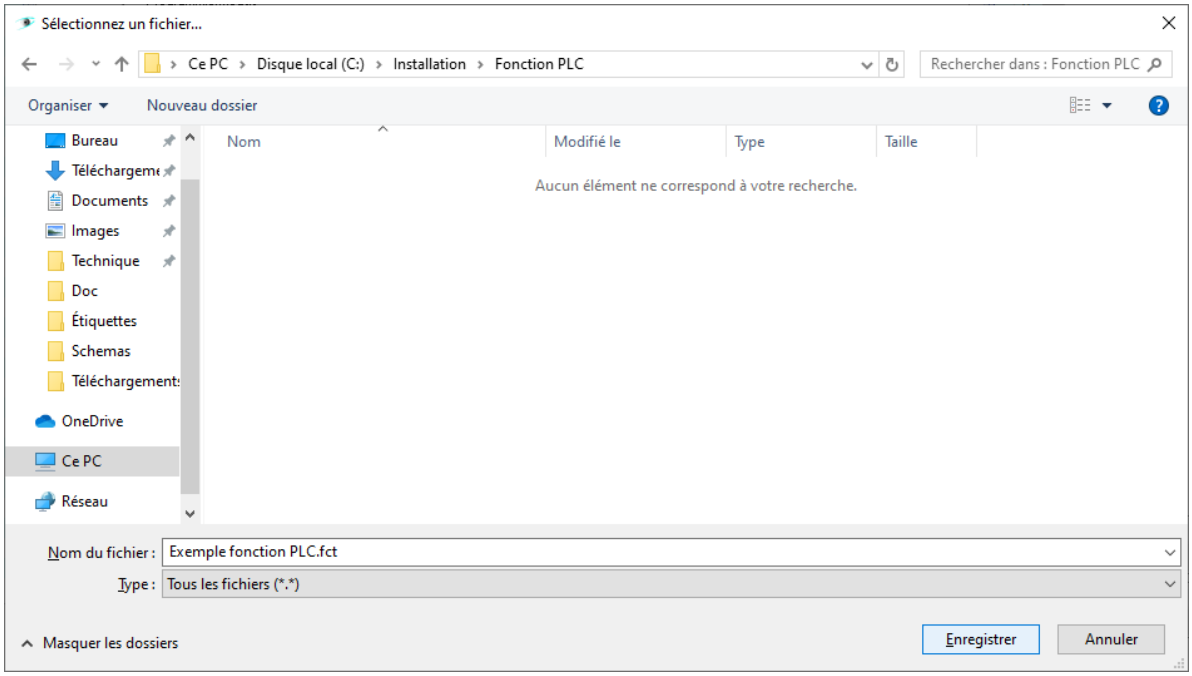


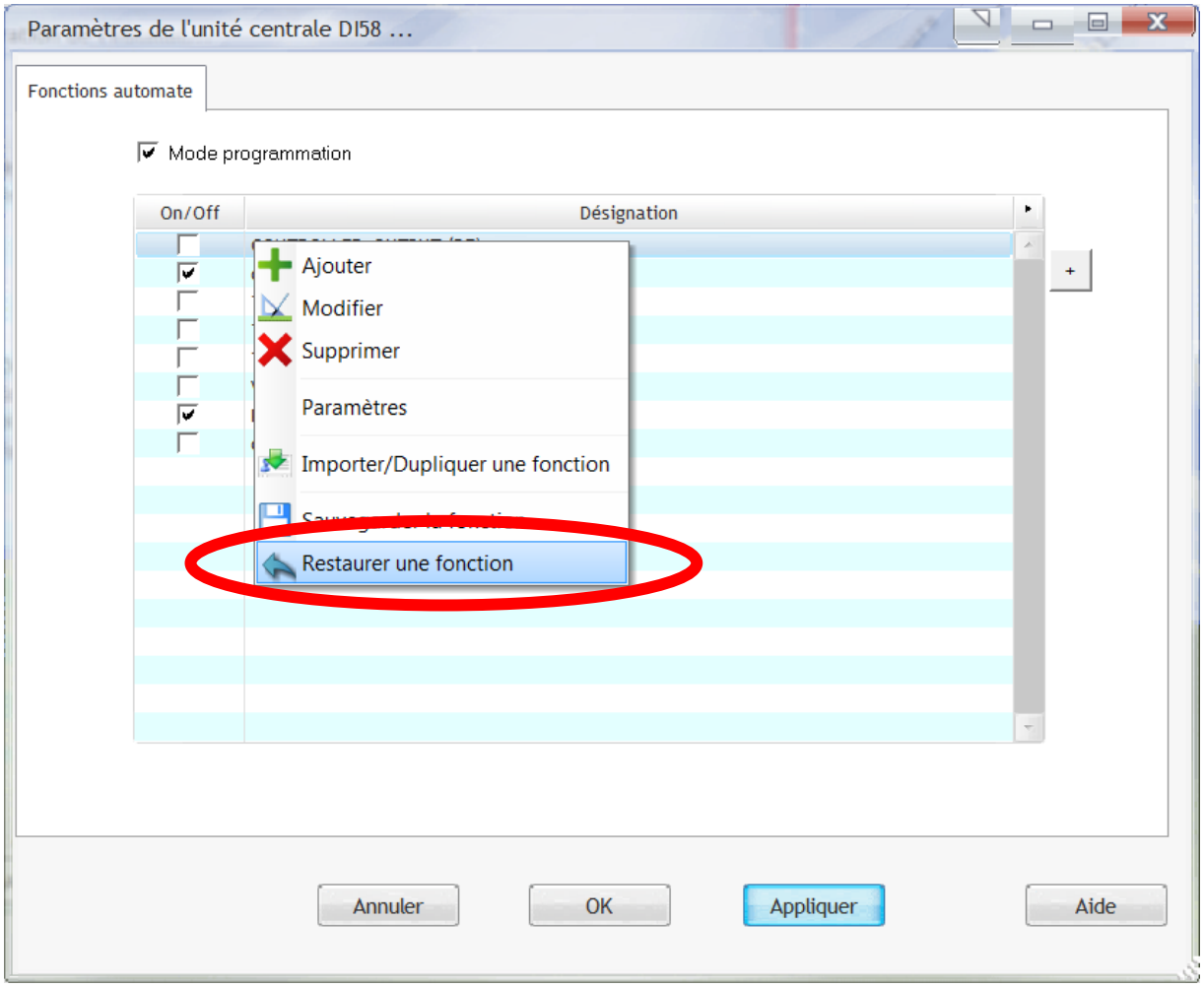
# 9. SAUVEGARDER ET RESTAURER UNE FONCTION

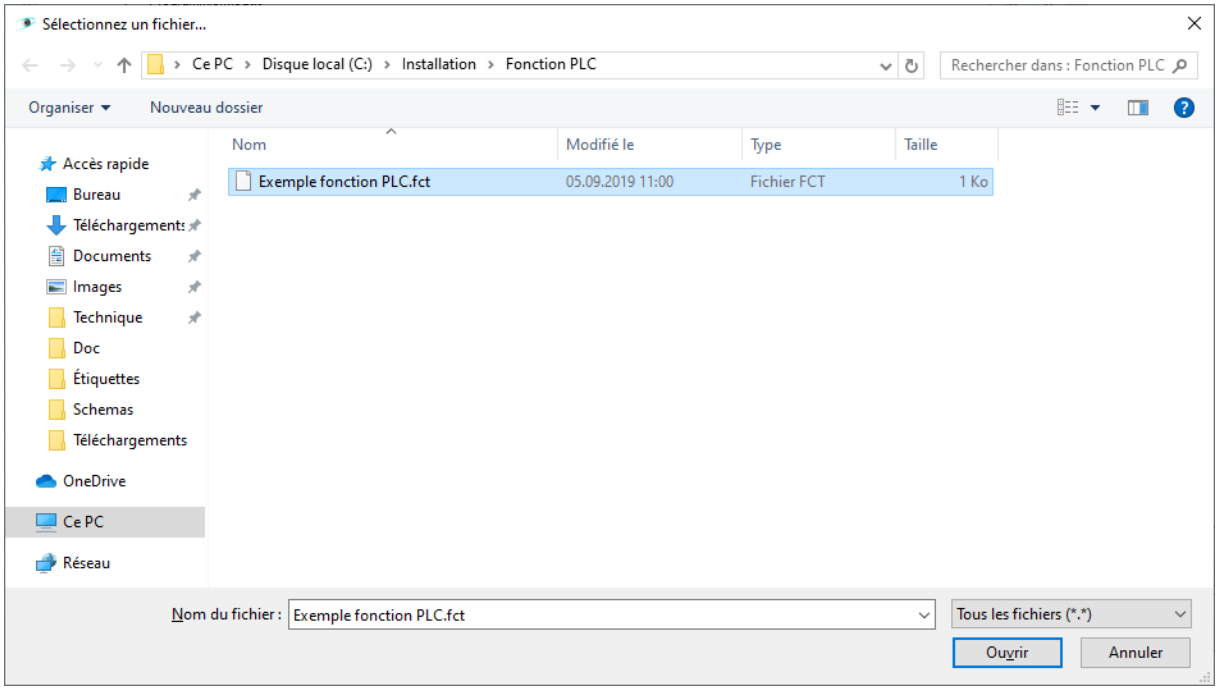
Il est possible de sauvegarder des fonctions sur votre ordinateur. Il faut ouvrir le menu contextuel en cliquant sur le bouton de droite de la souris sur la fonction que vous voulez sauvegarder et choisir la fonction **`Sauvegarder la fonction`**. Dans la fenêtre qui apparaît, on peut sélectionner le dossier et le nom sous lequel vous voulez sauvegarder la fonction.



Pour restaurer une fonction, il faut aussi ouvrir le menu contextuel en cliquant sur le bouton de droite de la souris sur la fonction et en choisissant la fonction **`Restaurer une fonction`**. Dans la fenêtre qui apparaît, on peut sélectionner une fonction qui est sauvegardée sur votre ordinateur que vous voulez restaurer.



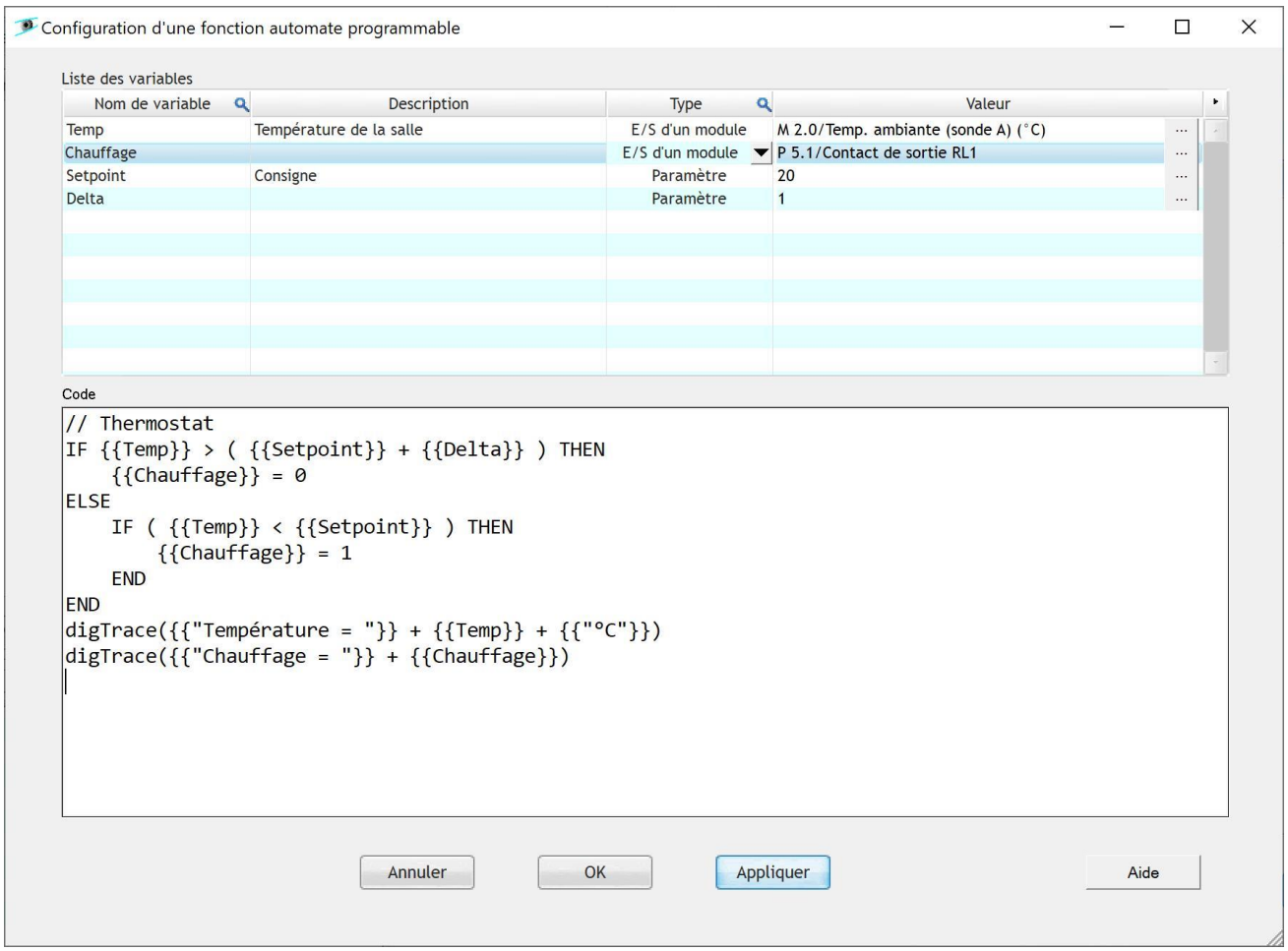
# 10. EXEMPLES

## 10.1. VENTILATION DE LA SALLE DES MACHINES

```
// Ventilation salle des machines
IF  {{TSalleMachines}}  >  (  {{Consigne}}  +  {{Delta}}  )  AND  {{TExterieure}}  <
{{TSalleMachines}} THEN
    {{Ventilateur}} = 1
ELSE
    IF {{TSalleMachines}} < {{Consigne}} OR {{TExterieure}} >= {{TSalleMachines}} THEN
        {{Ventilateur}} = 0
    END
END
digTrace({{"Ventilateur = "}} + {{Ventilateur}})
```

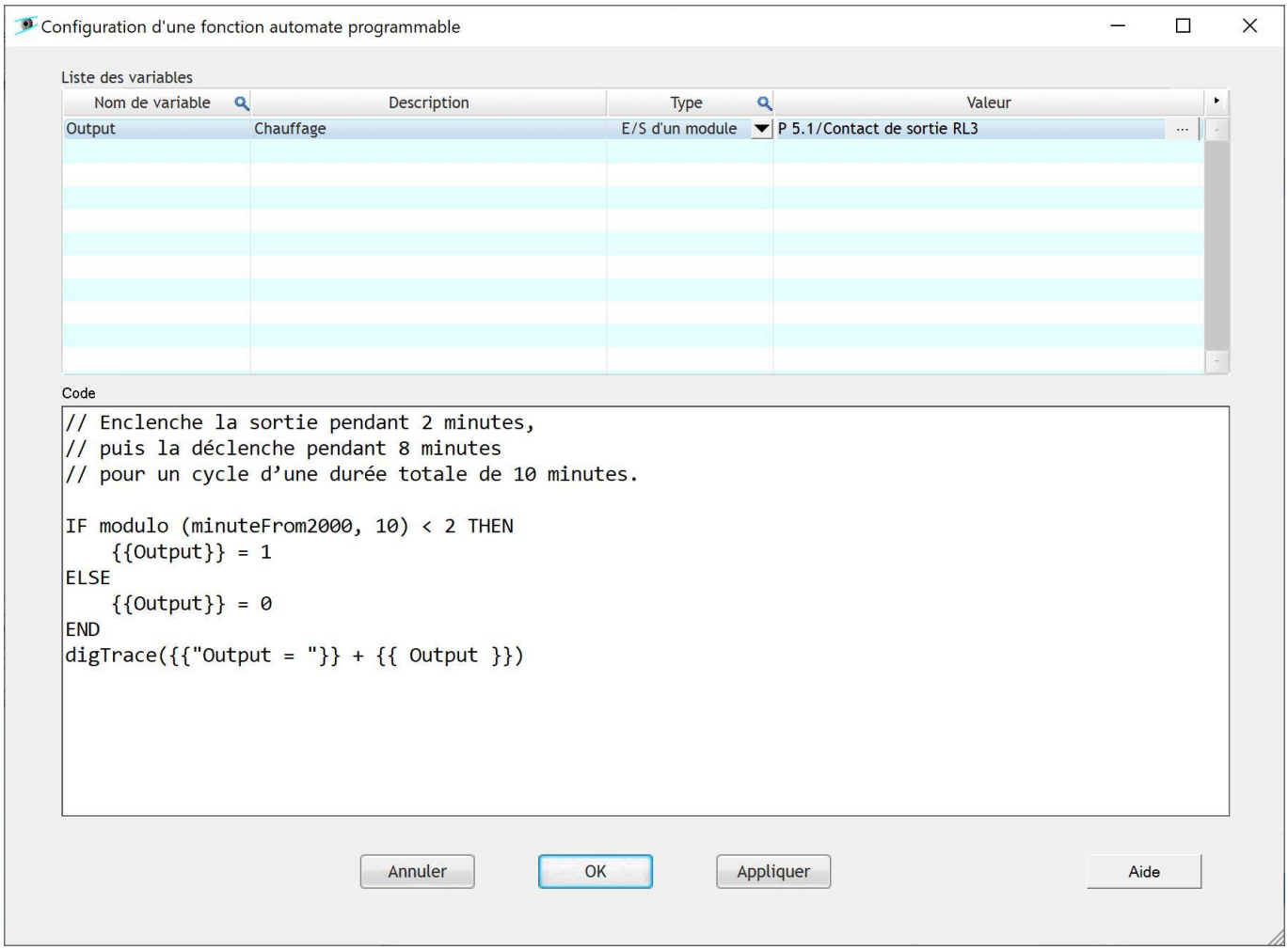
## 10.2. THERMOSTAT

```
// Thermostat
IF {{Temp}} > ( {{Setpoint}} + {{Delta}} ) THEN
    {{Chauffage}} = 0
ELSE
    IF ( {{Temp}} < {{Setpoint}} ) THEN
        {{Chauffage}} = 1
    END
END
digTrace({{"Température = "}} + {{Temp}} + {{"°C"}})
digTrace({{"Chauffage = "}} + {{Chauffage}})
```




## 10.3. TIMER CYCLIQUE

```
// Enclenche la sortie pendant 2 minutes,
// puis la déclenche pendant 8 minutes
// pour un cycle d'une durée totale de 10 minutes.

IF modulo (minuteFrom2000, 10) < 2 THEN
    {{Output}} = 1
ELSE
    {{Output}} = 0
END
digTrace({{"Output = "}} + {{ Output }})
```


Configuration d'une fonction automate programmable
Liste des variables
Nom de variable
Description
Type
Valeur
Output
Chauffage
E/S d'un module
P 5.1/Contact de sortie RL3
Code
// Enclenche la sortie pendant 2 minutes,
// puis la déclenche pendant 8 minutes
// pour un cycle d'une durée totale de 10 minutes.
IF modulo (minuteFrom2000, 10) < 2 THEN
{{Output}} = 1
ELSE
{{Output}} = 0
END
digTrace({{"Output = "}} + {{ Output }})
Annuler
OK
Appliquer
Aide

## 10.4. UTILISATION DES TIMERS PRÉDÉFINIS DANS L'UNITÉ CENTRALE

```
// Pour la configuration du timer, voir le manuel
// "Newel 3 - Complet - FR.pdf"
// au chapitre 10.12.13

IF {{Timer_1}} THEN
    digTrace({{"Timer = 1"}})
ELSE
    digTrace({{"Timer = 0"}})
END
```