

MicroLADDER

Description

1. Lecteurs ciblés

Cette documentation est à destination des utilisateurs du logiciel MicroLADDER, permettant de programmer tous les automates de la gamme SIREA. Les particularités de chaque automate sont décrites dans leur notice respective.

2. Version installée

Cette notice s'adresse à la version 18.3 de MicroLADDER.

Avertissement

Il est important de noter que cette notice est pour la dernière version de Mic

3. Installation

Pour installer le logiciel MicroLADDER, plusieurs manipulations vont devoir être réalisées pour qu'il fonctionne correctement. On va devoir installer plusieurs entités. Le tableau suivant décrit les entités ainsi que leur utilité.

Entité	Utilité
Compilateur	Ce logiciel a pour rôle de rechercher toutes les erreurs possibles dans un programme source, telles que des fautes d'orthographe, les variables, les types, etc.
MicroLADDER	Ce logiciel sert à réaliser l'application, la compiler, faire de la visualisation dynamique et du forçage de variables.
Code système	Le code système est une archive qui rassemble l'ensemble des macros et des fonctions (dans des fichiers .h et .c) qui font fonctionner les automates. C'est macros et fonctions peuvent être utilisés dans les programmes. Il sert aussi à gérer le côté hardware avec les différentes entrées/sorties des automates
MicroCONTROL	Ce logiciel sert à charger l'application, faire de la visualisation dynamique et du forçage.
MicroDRIVER	Ce logiciel est utilisé par MicroCONTROL et MicroLADDER pour la communication avec l'automate.

Tableau sur les entités et leurs utilités

3.1 Étape 1: Installation du Compilateur

3.1.1 Sous le système d'exploitation Windows

Pour Windows, exécuter le fichier d'installation « gcc-arm-none-eabi-4_9-2015q1-20150306-win32.exe » en double cliquant dessus.

Cocher « Add path to environment variable » à la fin de l'installation et fermer la fenêtre de commande qui reste ouverte.

Avertissement

Sous Windows 7, il faut installer le compilateur à partir du compte administrateur.

3.1.2 Sous le système d'exploitation Linux

Ouvrir tout d'abord un terminal.

Taper dans le terminal « sudo apt-get install gcc-arm-none-eabi ». Cela permet d'installer la version du compilateur GCC compatible avec la distribution Linux.

Pour connaître la version installée, taper dans un terminal « arm-none-eabi-gcc --version ».

3.2 Étape 2: Installation de MicroLADDER

3.2.1 Sous le système d'exploitation Windows

Pour l'installer, sous Windows, il faut exécuter le fichier « setup-mladder-x.x.exe » en double cliquant dessus.

3.2.2 Sous le système d'exploitation Linux

Ouvrir un terminal.

Taper puis exécuter la commande « sudo dpkg -i mladder-x.x.deb ». « x.x » correspond à la version choisie.

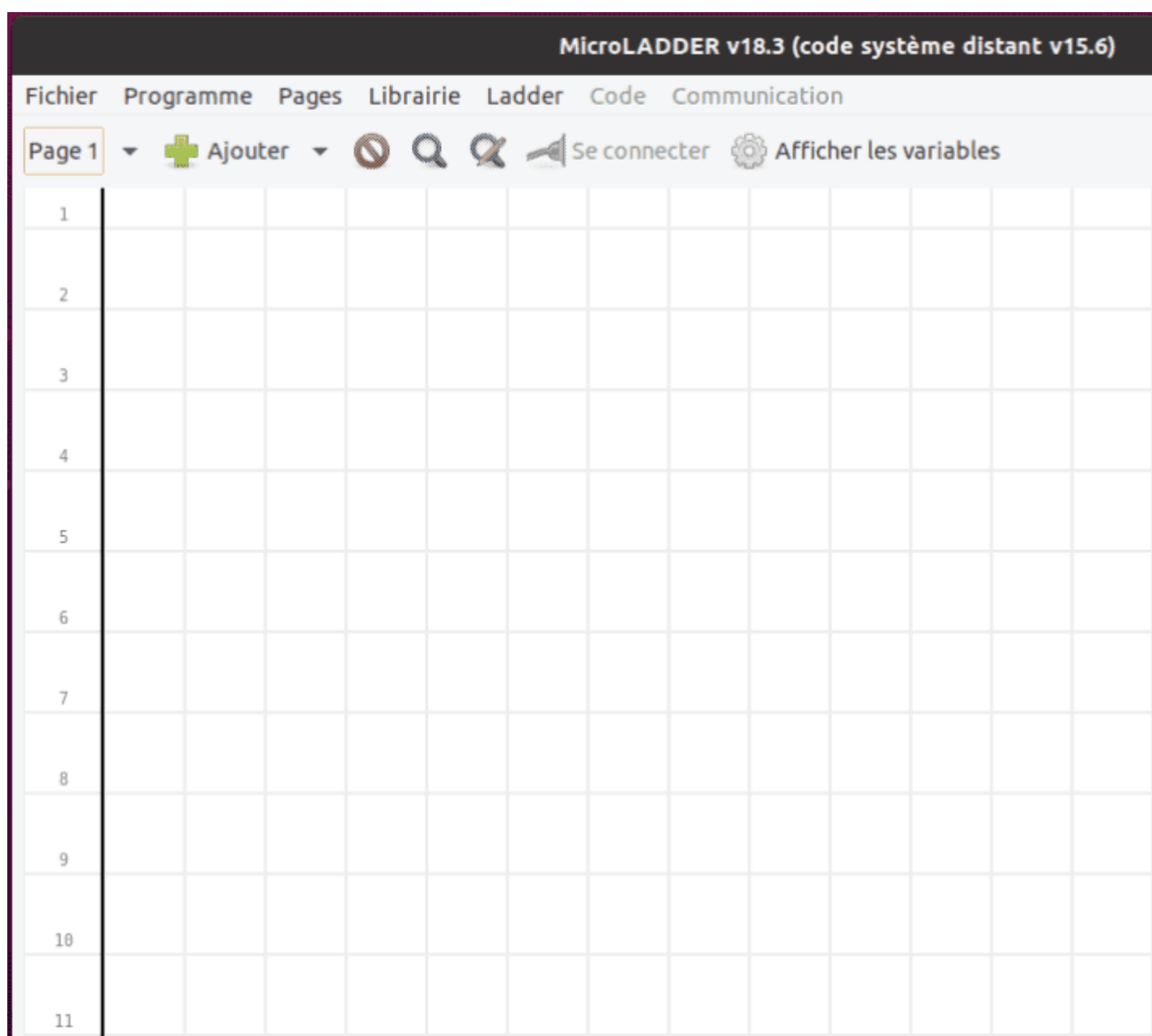
Si l'installation ne fonctionne pas par manque de dépendance, il est nécessaire de lancer la commande « sudo apt-get -f install ».

3.3 Étape 3: Installation du code système

Ouvrir le logiciel MicroLADDER dont l'icône est la suivante:



La fenêtre suivante s'ouvre:



Capture d'écran de la fenêtre principale du logiciel MicroLADDER

Dans l'onglet « Programme », cliquez sur « Importer le code système ». Une fenêtre *popup* s'ouvre. Rechercher puis sélectionner le fichier « mArm.sys ».

Avertissement

Le choix du code système doit se faire en fonction de la version du boot installé.

Information

Cette étape est optionnelle. En effet, lors de l'exécution d'un programme, un

3.4 Étape 4: Installation de MicroCONTROL

3.4.1 Sous le système d'exploitation Windows

Pour l'installer MicroCONTROL, il faut exécuter le fichier « setup-mcontrol.exe » en double cliquant dessus.

3.4.2 Sous le système d'exploitation Linux

Ouvrir un terminal.

Taper puis exécuter la commande « `sudo dpkg -i mcontrol.deb` ».

3.5 Étape 5: Installation de MicroDRIVER

3.5.1 Sous le système d'exploitation Windows

Pour l'installer, sous Windows, il faut exécuter le fichier « setup-mdriver.exe » en double cliquant dessus.

3.5.2 Sous le système d'exploitation Linux

Ouvrir un terminal.

Taper puis exécuter la commande « `sudo dpkg -i mdriver.deb` ».

3.6 Étape 6: Redémarrer votre ordinateur

La dernière étape, quoique la plus simple est de redémarrer votre ordinateur.

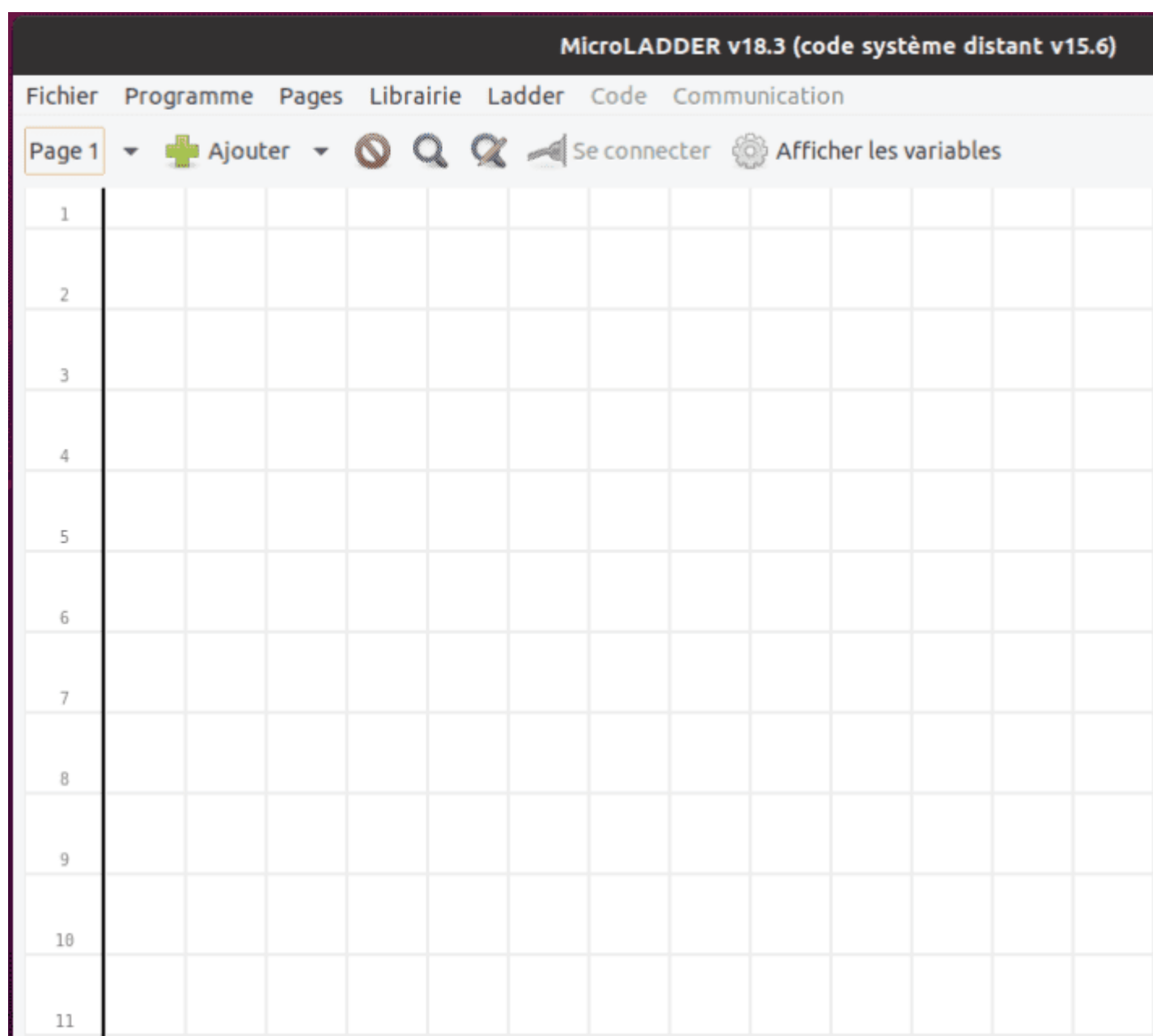
4. Prise en main de l'interface

4.1 Lancer et quitter MicroLADDER

Pour démarrer le logiciel, double-cliquer sur l'icône de MicroLADDER.



Le logiciel MicroLADDER s'ouvre alors sur la fenêtre principale suivante:

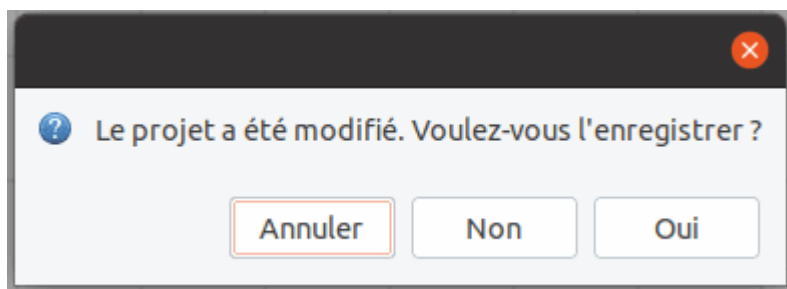


Capture d'écran de la fenêtre principale du logiciel MicroLADDER

Pour quitter le logiciel, cliquez soit sur l'icône « Fermer » (la croix rouge en haut à droite) soit aller dans la barre du menu puis Fichier>Quitter.

Une fenêtre *popup* s'ouvre alors nous demandant si on souhaite sauvegarder les modifications qui ont été faites à notre programme. Si vous cliquez sur « Non » alors vous perdrez toute dernières modifications non enregistrées.

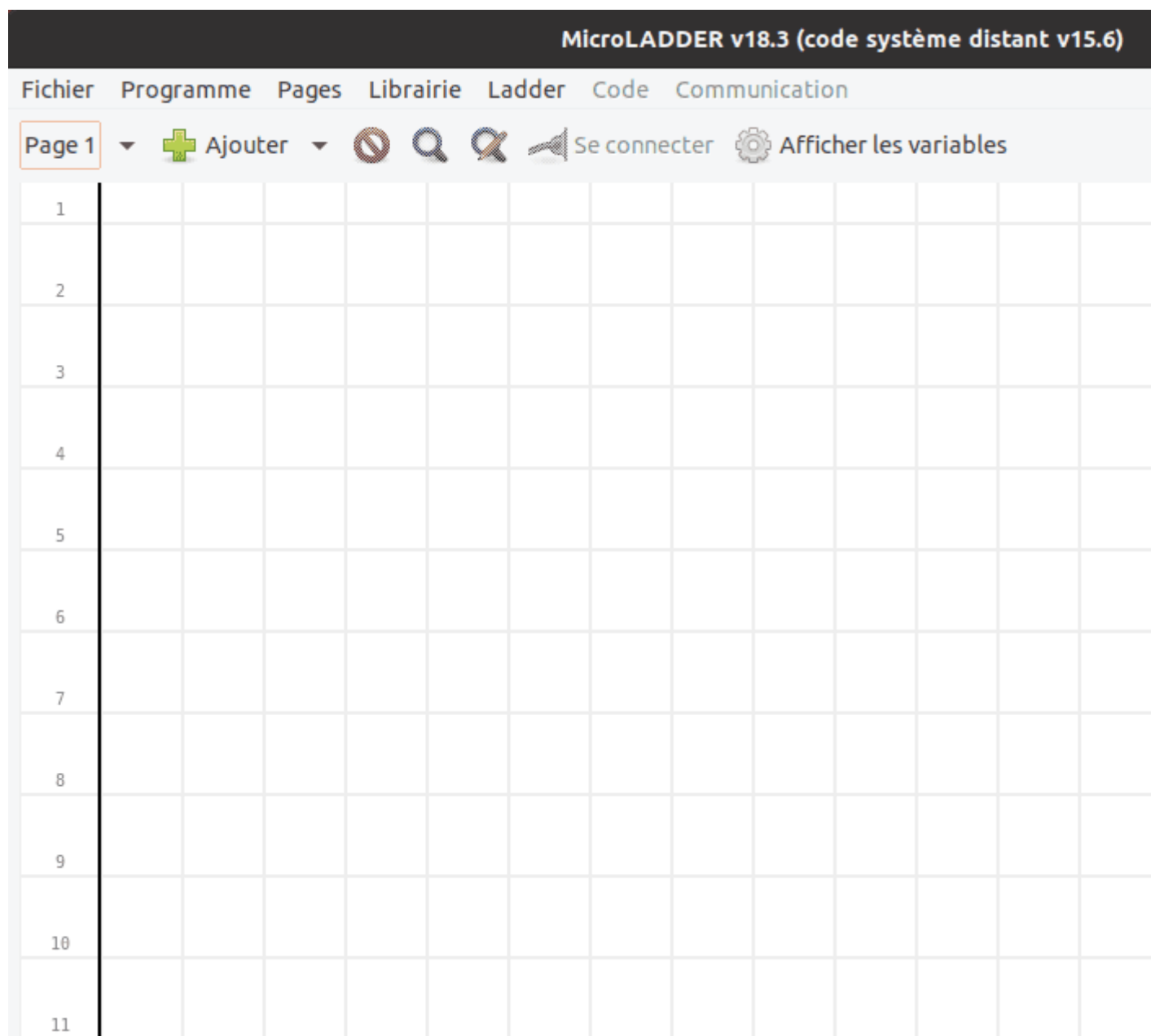
Si on clique sur « Annuler », on ne ferme plus le logiciel.



4.2 Caractéristiques de la fenêtre principale

Les éléments suivants peuvent être trouvés sur la fenêtre principale :

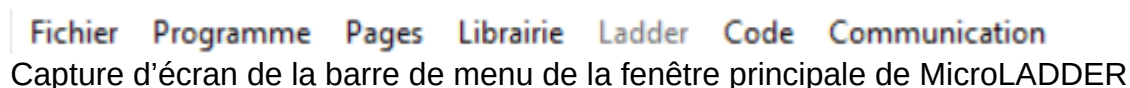
- **Barre de titre** : vous pouvez y trouver la version du MicroLADDER installé. Dans l'exemple de la Capture d'écran ci-dessous, vous pouvez voir « MicroLADDER v18.3 ». Elle indique également si vous utilisez le code système à distance ou embarqué. Ici, on utilise le code système distant v15.6.
- **Barre du menu**: vous pouvez y trouver les onglets:
 - Fichier
 - Programme
 - Pages
 - Librairie
 - Ladder
 - Code
 - Communication
- **Barre d'outils** : la barre d'outils comprend les éléments suivants :
 - Sélectionneur de page
 - Bouton « Ajouter »
 - Bouton de suppression
 - Bouton de recherche
 - Bouton de recherche et de remplacement
 - Bouton « Se connecter »
 - Bouton « Afficher les variables »
- **Fenêtre de programmation** : cette fenêtre délimite l'espace pour programmer. On peut se déplacer dans cette fenêtre en utilisant la barre de défilement située sur le côté droit.



Capture d'écran de la fenêtre principale de MicroLADDER

4.3 Barre de menu

La barre de menu MicroLADDER contient une série de menu déroulants qui peuvent être utilisés pour accéder aux différents outils et utilitaires de configuration du logiciel.



Capture d'écran de la barre de menu de la fenêtre principale de MicroLADDER

Les menu déroulants suivants par ordre de la gauche vers la droite sont inclus dans la barre de menu :

4.3.1 Fichier (Alt+F)

Les fonctionnalités suivantes ainsi que leur raccourci (explicité entre parenthèses) sont disponibles:

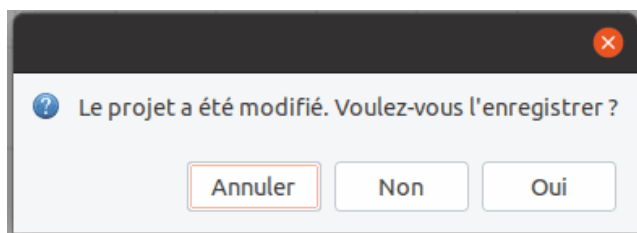
- **Nouveau** (Ctrl+N): crée un nouveau projet. Si vous avez déjà un projet ouvert, il vous sera demandé d'enregistrer ou non les modifications.
- **Ouvrir** (Ctrl+O): ouvre un projet déjà existant. Une fenêtre *popup* pour sélectionner le fichier dans vos dossiers apparaîtra.
- **Enregistrer** (Ctrl+S): enregistre les modifications du projet ouvert avec le nom et l'emplacement prédéfinis. Si le projet sur lequel on travaille n'a pas encore été ouvert alors une fenêtre *popup* s'ouvre pour sélectionner l'endroit dans lequel on souhaite sauvegarder le fichier. Un fichier .lad sera alors enregistré à l'emplacement spécifié. Un astérisque situé dans la barre de titre à côté de la version du code système indique si les dernières modifications effectuées sur le projet en cours ont déjà été enregistrées (sans astérisque) ou non (avec astérisque).

MicroLADDER v18.3 (code système distant v15.6) *

MicroLADDER v18.3 (code système distant v15.6)

Capture d'écran de la barre pour un projet non sauvegardé (à gauche) et sauvegardé (à droite)

- **Enregistrer sous** : enregistre les modifications du projet en choisissant le nom et l'emplacement dans une une fenêtre *popup*. Les cinq projets les plus récents seront également affichés dans ce menu, pour un accès facile.
- **Quitter** (Ctrl+Q): quitte MicroLADDER. Si on n'a pas enregistré les modifications manuellement, alors une fenêtre *popup* nous demandant si on souhaite sauvegarder apparaîtra. Si vous cliquez sur « Non » alors on perdra toutes dernières modifications non enregistrées. Si on clique sur « Annuler », on ne ferme plus le logiciel.



Capture d'écran de la fenêtre *popup* de fermeture de MicroLADDER

4.3.2 Programme (Alt + P)

- **Compiler**: compile un programme. Après la compilation, une nouvelle fenêtre *popup* s'ouvrira nous permettant de choisir un emplacement où l'enregistrer.

Avertissement

Pour compiler, le code système doit d'abord être importé dans le projet ou être

- **Définir le type de programme:** sélectionne le type de programme à développer, soit un bloc fonction, soit un programme spécifique automate parmi:
 - µArm A1
 - µArm A2
 - µArm A2-C
 - µArm A3
 - µArm A3-B
 - µArm A4
 - µArm A4+
 - µArm A5
 - µArm A6
 - µArm A7
 - µArm A8
 - µArm A9 A
 - µArm A9 B
 - µArm A10
 - µArm A11
 - µArm A12
 - µArm A13
 - µArm A14
 - µArm A15
 - µArm A16
 - µArm G1
 - µArm H1

Avertissement

Il est à noter que le code système doit être importé pour voir toutes les options.

- **Définir une icône:** associe une icône au projet. Cette icône apparaîtra dans l'icône de la fenêtre à la place de l'icône de MicroLADDER. C'est une fonctionnalité disponible à la fois pour les programmes automates et les blocs fonctions. En revanche, il est plus utile de définir une icône pour un bloc fonction car une image sur le bloc sera affichée lorsqu'il est importé dans un projet et appelé sur une page Ladder.
- **Équipements:** donne la liste des appareils, divisée en deux types :
 - Équipement de base : cette option crée une liste de paramètres qui permettront de configurer la remontée des données sur le serveur.
 - Ajouter un équipement distant : cette option permet de définir le ou les esclaves utilisé(s) pour créer un réseau Modbus (voir la section [7. IO Bus](#)).
- **Afficher les variables:** cette option permet d'ouvrir la fenêtre de l'éditeur de variables. Cette fenêtre contient toutes les informations concernant les variables du programme, tant les variables système que les variables utilisateur. Cette fenêtre sert également à vérifier la valeur en temps réel des variables lors de la communication avec l'API. On peut également accéder à cette option en utilisant le bouton « Afficher les variables » situé sur la barre d'outils.
- **Configurer le code système :** configure les options pour la compilation. Cliquez sur cette option pour modifier les paramètres du code système en sélectionnant ceux qui nous intéressent parmi:
 - AUTO-STOP (par défaut)
 - DHCP (par défaut)

- DNS (par défaut)
- GFX
- HTTP
- LCD (par défaut)
- RF

L'explication pour chaque paramètres est dans la section [6.4 Configuration du code système](#).

- **Importer le code système:** avant de créer un nouveau programme, le code système doit être importé. Cliquez sur cette option, une fenêtre *popup* s'ouvre alors, nous permettant de sélectionner le fichier mArm.sys. Une fois le code système correctement importé, les différentes options pour le type de programme seront disponibles.
Si vous disposez d'une connexion Internet, vous n'avez pas besoin d'importer le code système car MicroLADDER utilisera le code système en ligne mais ceci est déconseillé.
- **Exporter le code système:** cette option permet d'exporter le code système actuellement importé dans le projet, afin de l'utiliser dans d'autres projets. On peut utiliser cette option pour développer un nouveau programme en utilisant le même code système, ou simplement pour connaître la version du code système d'un automate.
- **Supprimez le code système:** cette option permet de supprimer le code système précédemment importé.
- **Configurer l'IHM:** configure les options pour l'IHM.
- **Importer l'IHM:** cette option permet, si on a créé une IHM avec MicroHMI de l'inclure dans le projet actuel.
- **Exporter l'IHM:** cette option permet d'exporter une IHM précédemment importée.
- **Supprimer l'IHM:** cette option permet de supprimer une IHM précédemment importée.

4.3.3 Pages (Alt+G)

- **Ajouter une page:** ajoute une nouvelle page à votre projet actuel. Une fenêtre *popup* s'ouvre. Les champs suivants y sont disponibles:
 - « Libellé »: chaîne de caractères du nom que va prendre la page si on ne remplit pas ce champ un libellé sera créé automatiquement en attribuant un numéro à la page
 - « Langage »: menu déroulant permettant de choisir Ladder ou C. Par défaut le langage est le Ladder.
 - « Appeler sur interruption »: champs à cocher si on souhaite que la page soit appelée par une interruption
 - « Timer (en ms) »: entier indiquant le temps (en millisecondes) au bout duquel la page sera appelée par interruption

Capture d'écran de la fenêtre *popup* « Ajouter une page »

- **Modifier cette page:** cette option permet de modifier les champs explicités juste au dessus d'une page spécifique déjà créée.
- **Dupliquer cette page:** duplique la page sur laquelle on se trouve. Les propriétés et le code seront conservés dans cette nouvelle page.

Information

Il est à noter que la nouvelle page ajoutée aura une étiquette automatiquement

- **Déplacer cette page:** déplace la page sur laquelle on se trouve actuellement. En cliquant sur cette option, une fenêtre *popup* s'ouvre nous demandant de sélectionner la nouvelle position de la page grâce à un menu déroulant.

Capture d'écran d'une fenêtre *popup* « Déplacer cette page »

- **Supprimer cette page:** supprime la page sur laquelle on se trouve. Une nouvelle fenêtre de confirmation de suppression de la page s'ouvre. Si on souhaite bien supprimer la page, on clique sur « Oui » sinon fermer la fenêtre ou cliquez sur « Non ».

Capture d'écran de la fenêtre *popup* de confirmation de suppression de la page actuelle

- **Page 1 (et suivante):** liste des différentes pages disponibles dans le projet directement accessibles depuis ce menu.

4.3.4 Librairie (Alt+L)

- **Importer une fonction:** importe une fonction dans le projet en cours en la sélectionnant dans la fenêtre *popup* qui apparaît.
- **Exporter une fonction:** exporte une fonction vers le projet en cours en la sélectionnant dans la nouvelle fenêtre *popup* qui apparaît.
- **Supprimer une fonction:** supprime une fonction du projet en cours en la sélectionnant dans la nouvelle fenêtre *popup* qui apparaît.

4.3.5 Ladder (Alt+D)

Avertissement

Ce menu déroulant ne sera pas accessible depuis une page C.

- **Tout sélectionner** (Ctrl+A): cette option permet de sélectionner tous les éléments inclus dans une page active.
- **Inverser la sélection** (Ctrl+Maj+I): cette option permet de sélectionner les éléments que l'on ne sélectionne pas actuellement (inverse la sélection). Si aucun élément n'est sélectionné, tous les éléments seront alors sélectionnés en cliquant sur cette option.
- **Couper** (Ctrl+X): cette option permet de couper un/des élément(s). Une fois sélectionné le ou les éléments que vous souhaitez couper, cliquez sur cette option. Le ou les éléments sélectionnés ne disparaîtront pas et ne changeront pas d'apparence pour indiquer qu'ils sont en train d'être coupés. Il (Ils) disparaîtront simplement de sa (leur) position précédente lorsque vous le (les) collerez. On peut également couper le ou les éléments en sélectionnant cette option dans le menu contextuel.
- **Copier** (Ctrl+C): cette option sert à copier un/des éléments. Une fois sélectionné le ou les éléments que vous souhaitez copier, cliquez sur cette option. Vous pouvez également copier le ou les éléments en sélectionnant cette option dans le menu contextuel.
- **Coller** (Ctrl+V): cette option permet de supprimer un/des éléments de la page. Positionnez le curseur de la souris à l'endroit où vous souhaitez coller le ou les éléments précédemment coupé(s) ou copié(s). On peut également coller le ou les éléments en sélectionnant cette option dans le menu contextuel.
- **Supprimer** (suppr): cette option sert à supprimer un/des éléments de la page. Une fois sélectionné(s) le(s) élément(s) que l'on souhaite supprimer, cliquez sur cette option. On peut supprimer également le ou les éléments en sélectionnant cette option dans le menu contextuel ou en utilisant le bouton « Supprimer » placé dans la barre d'outils. Ce bouton s'active lorsqu'il y a une sélection.
- **Propriétés:** cette option permet d'affecter la variable ou le code associé à l'élément sélectionné (un seul élément à chaque fois). La variable peut être affectée à la fois avec l'adresse et le mnémonique. On peut également accéder aux propriétés de l'élément en sélectionnant cette option dans la menu contextuel ou en double-cliquant sur l'élément. Notez que le double-clic ne fonctionne que si MicroLADDER n'est pas connecté à l'automate. Si il y a une connexion alors le double-clic permet de forcer la valeur d'une variable.
- **Ajouter:** cette option sert à inclure un nouvel élément à la page actuelle Ladder. Il suffit de sélectionner l'élément désiré dans le menu déroulant et de le placer avec le curseur de la souris à l'endroit voulu par un simple clic gauche. On peut également ajouter des éléments à partir du

menu contextuel ou en utilisant le bouton « Ajouter » placé dans la barre d'outils. Lors de l'accès via le menu contextuel, assurez vous de placer d'abord le curseur de la souris à l'emplacement souhaité avant de cliquer avec le bouton droit, car l'élément sera placé directement sans avoir besoin d'un clic gauche supplémentaire.

4.3.6 Code (Alt+O)

Avertissement

Ce menu déroulant ne sera pas accessible à partir d'une page Ladder.

- **Rechercher** (Ctrl+F): cette option à partir d'une page de code C active la barre de recherche pour la recherche de code. Cette barre apparaîtra en bas de la fenêtre. Écrivez le texte à rechercher dans la barre de recherche et activez « Respecter la casse » pour une recherche sensible à la casse et « Retourner circulaire » si on souhaite que toutes les correspondances de la page soient marquées.



Capture d'écran de la fenêtre *popup* « Rechercher »

- **Rechercher et remplacer** (Ctrl+R): cette option à partir d'une page de code C ouvre une fenêtre *popup*. Précisez ensuite les critères de recherche et le texte de remplacement. On peut soit remplacer manuellement une seule partie d'un texte soit remplacer tout. Dans le cas d'un remplacement manuel, vous pouvez également le faire en avant ou en arrière, en activant ou en désactivant l'option Recherche en arrière.



Capture d'écran de la fenêtre *popup* « Rechercher et remplacer »

4.3.7 Communication (Alt+C)

Information

Ce menu déroulant ne sera pas accessible à partir d'un bloc fonction mais seul

- **Se connecter**: cette option est disponible lorsqu'aucune connexion n'a encore été établie. Une fenêtre *popup* « Connexion à MicroDRIVER » apparaîtra lorsque l'on clique sur cette option. Plusieurs champs (méthode de communication, le numéro d'esclave et l'adresse du MicroDRIVER) sont à renseigner afin de se connecter à l'automate. On peut également accéder à

la fenêtre « Connexion à MicroDRIVER en utilisant le bouton « Se connecter » placé dans le menu Outils. Pour plus d'informations sur comment établir une connexion, se référer à la section 5.3.2 Établissement de la connexion.

Capture d'écran de la fenêtre *popup* « Connexion à MicroDRIVER »

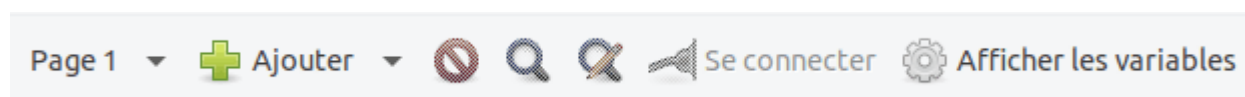
- **Déconnecter**: cette option est disponible lorsqu'une connexion a déjà été établie. En cliquant sur cette fonction la communication avec l'automate va être arrêtée.
- **Lecture du numéro d'esclave** : cette option est uniquement disponible lorsqu'une connexion avec un automate a été établie. Elle est utilisée pour obtenir le numéro d'esclave d'un esclave spécifique, lorsqu'il est prévu d'en connecter plusieurs en série, sur la même ligne.
- **Écriture du numéro d'esclave** : cette option est uniquement disponible lorsqu'une (et une seule) connexion avec un automate a été établie. Elle permet de modifier son numéro d'esclave.
- **Démarrage du programme** : cette option démarre un programme lorsque l'automate est à l'état STOP. Une fenêtre *popup* nous indiquant que le programme a bien démarré apparaît.
- **Arrêt du programme** : cette fonction arrête un programme lorsque l'automate est à l'état RUN.
- **Réinitialisation de l'équipement** : cette option réinitialise l'automate à son état BOOT. Elle est utile pour tester ses réactions lorsqu'il est allumé et éteint par exemple.
- **Réinitialisation des variables** : cette fonction remet toutes les variables à leur valeur initiale et n'arrête pas le programme. L'automate est toujours dans l'état RUN. Elle est utile pour simuler un comportement.
- **Chargement d'un programme** : cette option ouvre une fenêtre *popup* « Chargement d'un programme » nous permettant de choisir le fichier hexadécimal (.hex) que l'on souhaite charger sur l'automate. Cette action prend plusieurs dizaines de secondes, il est donc plus rapide de charger à l'aide d'une carte SD si l'automate dispose d'un lecteur de carte SD.

Information :

Il est possible de connaître la touche correspondante associée aux raccourcis

4.4 Barre d'outils

La barre d'outils se situe en haut de la fenêtre principale.



Capture d'écran de la barre d'outils de MicroLADDER

Les éléments suivants sont inclus dans la barre d'outils :

- **Sélectionneur de page**: cette option ouvre un menu déroulant permettant de sélectionner la page avec laquelle on souhaite travailler.
- **Ajouter**: cette option sert à inclure un nouvel élément Ladder sur la page actuelle. Il suffit de sélectionner l'élément souhaité dans le menu déroulant et de le placer avec le curseur de la souris à l'endroit souhaité par un simple clic gauche. On peut également ajouter des éléments à partir du menu par un clic droit. Lorsqu'on accède au menu par le clic droit, il faut s'assurer de placer d'abord le curseur de la souris à l'endroit désiré avant de cliquer avec le bouton droit de la souris. L'élément sera placé directement sans qu'il soit nécessaire de faire un clic gauche supplémentaire.
- **Icône « Supprimer »**: ce bouton est activé uniquement lorsqu'un élément est sélectionné sur la page. Une fois le ou les éléments que vous souhaitez supprimer sont sélectionnés, cliquez sur cette icône. Vous pouvez également supprimer le(s) élément(s) en sélectionnant cette option dans le menu du clic droit.
- **Icône « Rechercher »**: cette option à partir d'une page de code C active la barre de recherche pour la recherche de code. Cette barre apparaîtra en bas de la fenêtre. Écrivez le texte à rechercher dans la barre de recherche et activez « Respecter la casse » pour une recherche sensible à la casse et « Retourner circulaire » si on souhaite que toutes les correspondances de la page soient marquées.



Capture d'écran de la fenêtre *popup* « Rechercher »

- **Icône « Rechercher et remplacer »**: cette option à partir d'une page de code C ouvre une fenêtre *popup*. Précisez ensuite les critères de recherche et le texte de remplacement. Vous pouvez soit remplacer manuellement une partie d'un texte à un seul endroit soit tout remplacer. Dans le cas d'un remplacement manuel, vous pouvez également le faire en avant ou en arrière, en activant ou en désactivant l'option Recherche en arrière.



Capture d'écran de la fenêtre *popup* « Rechercher et remplacer »

- **Se connecter**: cette option est disponible lorsqu'aucune connexion n'a encore été établie. Une fenêtre *popup* « Connexion à MicroDRIVER » apparaîtra lorsque l'on clique sur cette option. Plusieurs champs (méthode de communication, le numéro d'esclave et l'adresse du MicroDRIVER) sont à renseigner afin de se connecter à l'automate. On peut également accéder à la fenêtre « Connexion à MicroDRIVER en utilisant le bouton « Se connecter » placé dans le

menu Outils. Pour plus d'informations sur comment établir une connexion, se référer à la section [5.3.2 Établissement de la connexion](#).

Avant cette étape, on doit avoir sélectionné le type d'automate puis s'être connecté à l'automate.

Capture d'écran de la fenêtre *popup* « Connexion à MicroDRIVER »

- **Afficher les variables:** cette option ouvre la fenêtre de l'éditeur de variables. Cette fenêtre contient toutes les informations concernant les variables du programme, tant les variables système que les variables utilisateur. On peut également l'utiliser pour vérifier la valeur en temps réel des variables lors de la communication avec l'API. On peut accéder à cette option à partir du menu « Programme » situé dans la barre de menu.

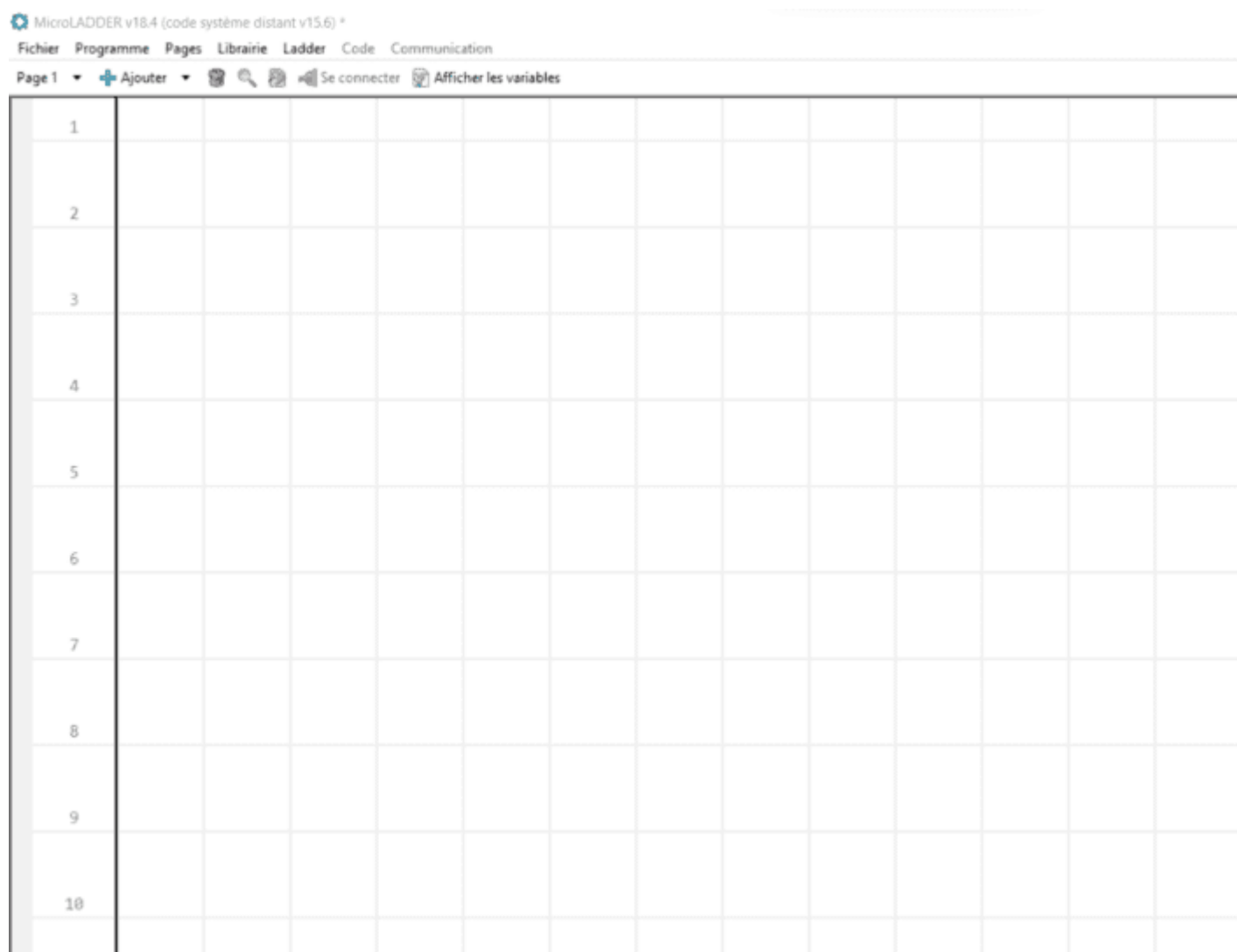
Capture d'écran de la fenêtre *popup* « Éditeur de variables »

4.5 Fenêtre de programmation

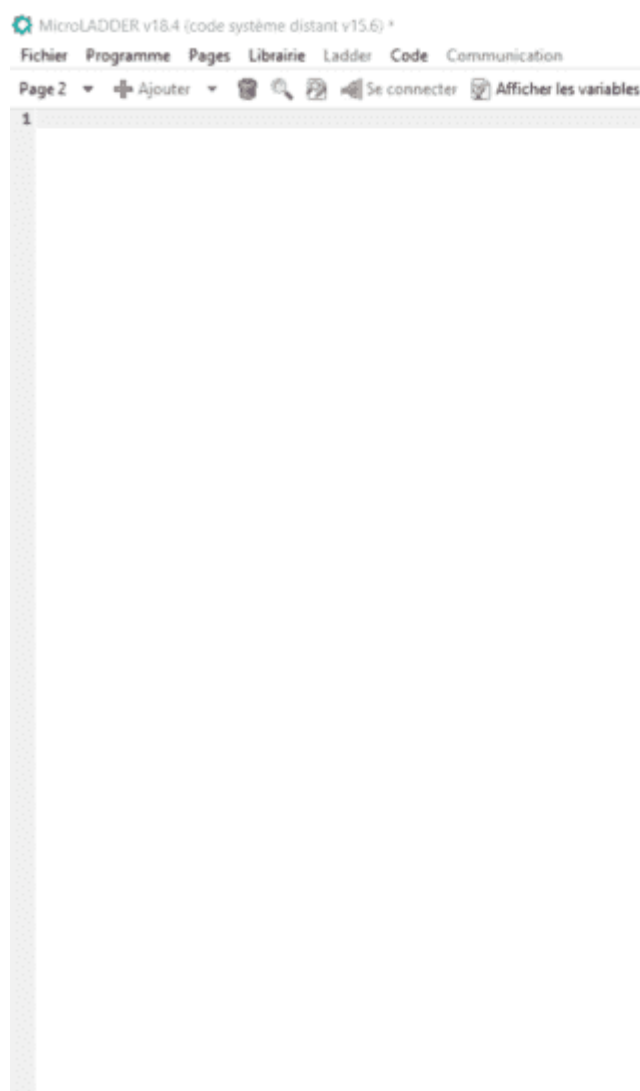
La fenêtre de programmation est le grand espace vide situé sous la barre d'outils utilisée pour le

développement de programmes. Deux types de fenêtres de programmation peuvent être affichées :

- la fenêtre Ladder
- la fenêtre C



Capture d'écran de la fenêtre principale en Ladder de MicroLADDER



Capture d'écran de la fenêtre principale en C de MicroLADDER

5. Programmer sur MicroLADDER

5.1 Les langages de programmation

5.1.1 Langage Ladder

L'idée initiale du Ladder est la représentation de fonction logique sous la forme de schémas électriques. Cette représentation est originellement matérielle : quand l'automate programmable industriel n'existait pas, les fonctions étaient réalisées par des câblages. Le Ladder a été créé et normalisé dans la norme CEI 61131-3. Il est encore aujourd'hui souvent utilisé dans la programmation des automates programmables industriels.

Un programme Ladder se lit de haut en bas et l'évaluation des valeurs se fait de gauche à droite. Les valeurs correspondent, si on les compare à un schéma électrique, à la présence ou non d'un potentiel électrique à chaque nœud de connexion. En effet, le Ladder est basé sur le principe d'une alimentation en tension représentée par deux traits verticaux reliés horizontalement par des bobines, des contacts et

des blocs fonctionnels, d'où le nom « Ladder » (échelle). C'est un langage volontairement simple et graphique pour être compréhensible. Cela a permis, dans les années 1990, son utilisation sans formation poussée par les électriciens.

5.1.2 Langage C

Le langage de programmation C a été initialement développé par Dennis Ritchie entre 1969 et 1973 chez Bell Laboratories. Il est approprié pour écrire des programmes au niveau du système en raison de la simplicité d'expression, de la compacité du code et de la large gamme d'applicabilité. Il permet au programmeur un large éventail d'opérations allant du haut niveau jusqu'au bas niveau approchant le niveau du langage assembleur. La flexibilité disponible est large ce qui en fait un langage de programmation parfait pour le développement de programmes automate de haut degré de complexité.

5.2 Importation du micrologiciel

Avec cette version de MicroLADDER, le firmware est en ligne et fonctionne automatiquement avec votre système. Aucune importation n'est nécessaire, mais une connexion Internet l'est.

5.3 Connexion avec l'automate

L'établissement d'une connexion avec l'automate est nécessaire lors du chargement d'un programme via un port et lors de la surveillance de variables lorsque l'automate est en RUN.

5.3.1 MicroDRIVER

MicroDRIVER est le logiciel chargé d'établir la connexion entre l'automate et les différentes applications du système. Il peut être considéré comme une « boîte noire » des communications. Il est indépendant du protocole de communication du système.

L'utilisateur ne sera pas au courant de son fonctionnement car il s'exécute en arrière-plan. Cependant, si MicroDRIVER n'est pas installé, il sera impossible d'établir la connexion avec l'automate.

5.3.2 Établissement de la connexion

Pour établir la connexion entre MicroLADDER et l'automate, il faut d'abord cliquer sur l'option « Se connecter » dans la barre outils. Une fois cette option sélectionnée, une fenêtre *popup* de connexion apparaîtra.

Capture d'écran de la fenêtre *popup* « Connexion à MicroDRIVER »

Pour la connexion avec MicroDRIVER, plusieurs champs sont à remplir parmi:

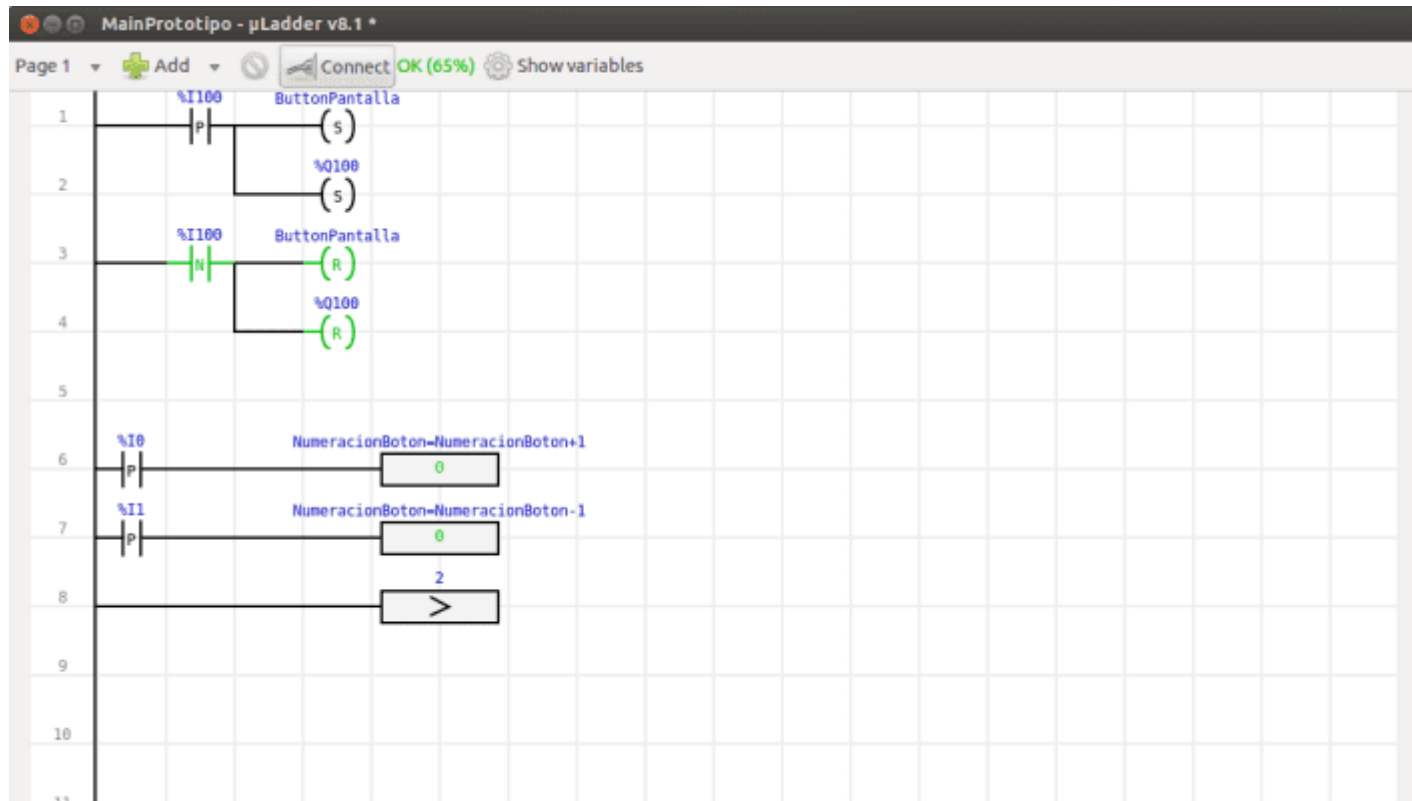
- **Méthode de communication** : plusieurs méthodes de communication sont disponibles:
 - Hôte réseau : la connexion est établie par le réseau. L'adresse IP ou le nom de domaine de l'hôte est à renseigner dans le champs à droite du menu déroulant. L'hôte réseau peut par exemple être notre ordinateur.
 - Port série : choisissez un port série dans la liste ainsi que sa configuration
 - Canal : fonctionnalité obsolète
- **Numéro d'esclave** : ce numéro identifie l'automate au sein d'un réseau d'automates. Vous n'avez pas besoin de définir le numéro d'esclave si vous programmez un seul automate. Il sera déterminé après la première connexion.
- **Adresse de MicroDRIVER** : MicroDRIVER est l'application serveur. Il peut s'exécuter localement ou sur un ordinateur distant. S'il s'exécute sur un ordinateur distant, il est nécessaire de spécifier l'adresse IP de cet ordinateur. La configuration de la connexion au MicroDRIVER peut différer selon la façon dont on se connecte à l'automate depuis votre ordinateur. Les paramètres de configuration les plus courants peuvent être vu dans le tableau ci dessous:

Type de connexion	Paramétrage	
PC -> Ethernet -> PLC	Méthode de communication	Hôte du réseau Adresse IP de l'automate spécifiée dans le fichier main.cf
	Numéro d'esclave	Généralement 1. Vous n'avez pas besoin de changer le numéro d'esclave
	Adresse MicroDriver	Rien
PC -> Serie -> PLC	Méthode de communication	Port série Sélectionnez votre numéro de port La configuration typique est "3"
	Numéro d'esclave	Numéro d'esclave du port PLC spécifié dans le fichier Lorsque plusieurs API sont connectés en série au même Port
	Adresse MicroDriver	Rien
PC -> Ethernet -> μFox -> Serial -> PLC	Méthode de communication	Canal Sélectionnez le numéro de canal pour μDriver (0 à 3) La configuration typique est "3"
	Numéro d'esclave	Numéro d'esclave du port PLC spécifié dans le fichier Lorsque plusieurs API sont connectés en série au même Port
	Adresse MicroDriver	Adresse du système communiquant directement avec l'automate L'adresse du MicroDRIVER est dans ce cas l'adresse du MicroDRIVER

Tableau sur les paramètres de configuration en fonction du type de connexion

Une fois les paramètres de connexion correctement renseignés, cliquez sur « Valider » et la connexion sera établie. Vous pouvez vous assurer que la connexion a été établie en vérifiant ce qui suit :

- Le bouton « Se connecter » de la barre d'outils est enfoncé (si on clique dessus, on se déconnecte directement à nouveau).
- L'option « Se connecter » du menu déroulant « Communication » de la barre de menu est désactivée et l'option « Déconnecter » est disponible.
- Un message vert « OK » est apparu sur la barre d'outils.
- Les conditions actives du programme apparaissent également en vert sur les pages Ladder



Capture d'écran de la fenêtre principale de MicroLADDER avec l'automate connecté et en ligne

Information

Un pourcentage est affiché en vert près de l'état de la connexion. Il s'agit s

5.4 Utilisation des pages

Le nombre de pages pouvant être utilisées dans n'importe quel programme MicroLADDER est illimité. Chaque page Ladder est limitée à 100 lignes alors que les pages C ne sont pas soumises à cette limitation. La limitation du nombre de lignes à l'intérieur d'une page en Ladder oblige l'utilisateur à structurer le programme en différentes pages et/ou à utiliser des fonctions, ce qui facilite l'interprétation du code.

La page 1 est appelée à chaque cycle automate mais on peut appeler d'autres pages facilement. On peut décider si une page spécifique doit être programmée soit en Ladder soit en C. Des pages de différents langages de programmation peuvent être appelées et combinées facilement. Cela offre à l'utilisateur une grande flexibilité et une interprétation facile de la structure lors de la création d'applications.

Assurez vous qu'il n'y a pas de code déjà généré dans une page lors du passage de C à Ladder ou vice-versa, car il sera perdu.

5.4.1 Créer une page

Un nouveau projet MicroLADDER inclut par défaut une seule page (Page 1) qui est configurée par défaut en Ladder. Il est possible de la modifier en allant dans le menu déroulant « Page » de la barre de menu puis en cliquant sur « Modifier la page ».

On peut facilement créer une page supplémentaire en ajoutant une nouvelle page ou en copiant une page existante (voir section [4.3.3 Pages \(Alt+G\)](#) pour plus de renseignements).

5.4.2 Appel d'une page

La page 1 est appelée automatiquement alors que les autres pages doivent être appelées sur interruption, timer, ou avec des ordres d'appel en C ou en Ladder.

5.4.2.1 Par interruption

Il est possible d'appeler les pages sur interruption, la page sera donc appelée à une certaine fréquence. Cette fréquence est paramétrée dans la variable système %SW25 pour une durée précise et rapide.

Information

Il est conseillé d'appeler des pages par interruption lorsqu'elles exécutent d

5.4.2.2 Par timer

Il est possible également d'appeler une page via un timer. On va définir ce timer dans les réglages de la propriété de la page. A chaque cycle automate, l'automate va aller vérifier si son temps exécution est supérieur au timer défini sur la page. Si il est supérieur alors la page va être exécutée.

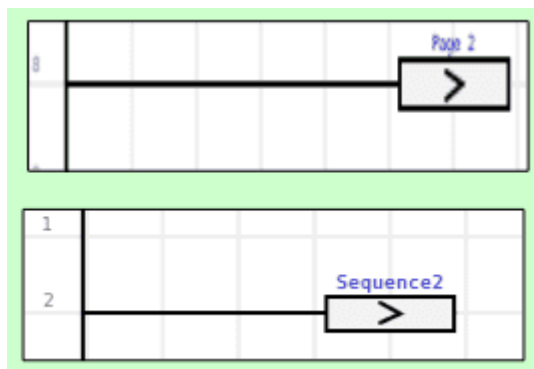
Information

Il est conseillé d'appeler des pages par timer lorsqu'elles s'exécutent lentem

5.4.2.3 Par ordre d'appel C ou Ladder

Une page peut être appelée à partir d'une autre page à l'intérieur d'un programme, quel que soit le type de code utilisé dans chaque page. Cela signifie qu'une page en Ladder peut être appelée à partir d'une page C et vice-versa.

Pour appeler une page en Ladder, il suffit d'ajouter une commande d'appel soit depuis le menu déroulant « Ladder » situé sur la barre de menu, soit par le menu contextuel.



Si la page appelée n'a pas d'étiquette, elle sera appelée par son numéro de page. Dans cet exemple, Page 2.

Si la page appelée a un libellé, elle peut être appelée soit par son numéro de page, soit par son libellé. Cependant en Ladder c'est le libellé qui s'affichera au lieu du numéro de la page. On observe cela avec l'appel de la page dont le libellé est « Sequence2 ».

Dans une page écrite en langage C, une autre page peut également être appelée soit par son numéro, soit par un libellé.

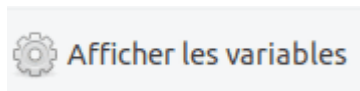
```
page_2(); //Pour appeler la page 2
Sequence2() ; //Pour appeler la page nommée "Sequence2".
```

5.5 Gérer les variables

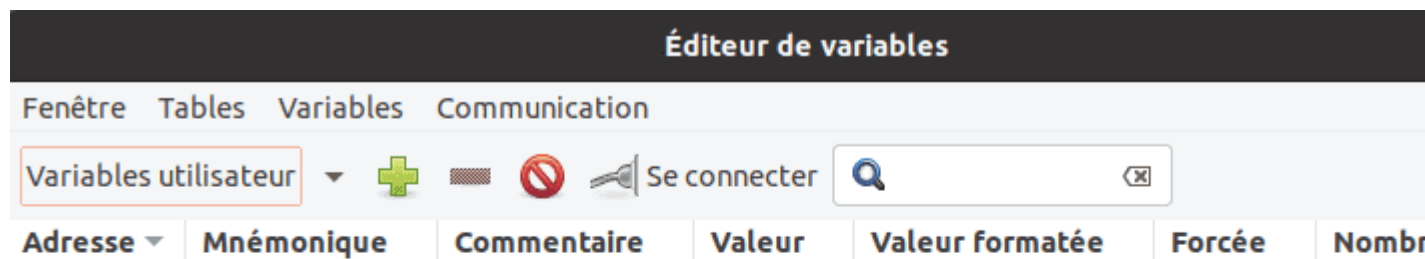
5.5.1 Éditeur de variables

Les variables dans MicroLADDER sont des emplacements de stockage avec un mnémonique associé qui contient une quantité connue ou inconnue d'informations.

Les variables disponibles dans un programme MicroLADDER peuvent être affichées et modifiées à partir de l'éditeur de variables.



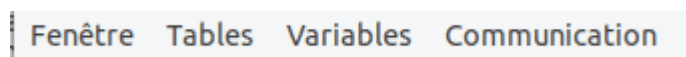
Vous pouvez accéder à l'éditeur de variables en cliquant sur le champ « Afficher les variables » en haut à droite dans la barre outils.



Capture d'écran de la fenêtre *popup* « Éditeur de variables »

5.5.1.1 Barre menu de l'éditeur de variables

L'éditeur de variables a sa propre barre de menu ayant chacun un menu déroulant.



Capture d'écran de la barre d'outils de l'éditeur de variables

Ces menu déroulants contiennent les options suivantes :

5.5.1.1.1 Fenêtre

- **Fermer:** on peut fermer l'éditeur de variables en cliquant sur cette option ou simplement en cliquant sur le bouton de fermeture en haut à droite

5.5.1.1.2 Tables

- **Ajouter une table:** dans MicroLADDER, une catégorie ou un groupe de variables est appelé table. Par défaut, MicroLADDER considère deux tables : les variables utilisateur et les variables système. De nouvelles tables peuvent être définies en cliquant sur cette option et en donnant un nom.
- **Renommer cette table:** cliquez sur cette option pour renommer un tableau. Les tables par défaut ne peuvent pas être renommées.
- **Supprimer cette table:** cliquez sur cette option pour supprimer une table. Les tables par défaut ne peuvent pas être supprimées.
- **Importer des variables dans cette table:** cliquez sur cette option pour importer un fichier de variables CSV dans le projet MicroLADDER en cours.
- **Exporter les variables de cette table:** cliquez sur cette option pour exporter un tableau de variables dans un fichier CSV à partir du projet MicroLADDER en cours.
- **Importer des valeurs dans cette table:** cliquez sur cette option pour importer des valeurs dans une table à partir du projet MicroLADDER en cours.
- **Exporter les valeurs de cette table:** cliquez sur cette option pour exporter les valeurs dans une table à partir du projet MicroLADDER en cours. Il est uniquement possible d'exporter dans un fichier CSV.
- **Variables système:** en cliquant sur cette option, un menu déroulant avec les différentes tables variables système apparaît. Les variables incluses dans chaque catégorie sélectionnée seront affichées dans la fenêtre principale de l'éditeur de variables.
- **Variables utilisateur:** en cliquant sur cette fonction, les variables créées par l'utilisateur seront affichées dans la fenêtre principale de l'éditeur de variables.

5.5.1.1.3 Variables

- **Ajouter:** cliquez sur cette option pour inclure une nouvelle variable au projet MicroLADDER en cours. Vous pouvez également ajouter une variable via la barre d'outils.
- **Sélectionner les variables inutilisées:** cliquez sur cette option pour sélectionner automatiquement toutes les variables qui ne sont pas utilisées dans le programme. Cette fonction offre une meilleure visibilité pour supprimer des variables.
- **Propriétés :** cliquez sur cette option pour configurer les variables.

Information

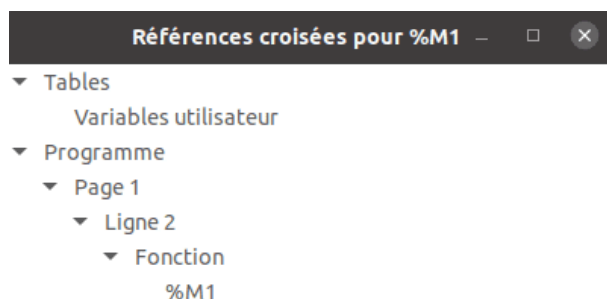
Il est à noter que le raccourci pour cette fonction est le double-clic sur une

- **Modifier la valeur:** certaines valeurs peuvent être forcées manuellement lors de la connexion avec l'automate. Cette option permet de définir une valeur spécifique en temps réel pour une variable.

Information

Il est à noter que le raccourci pour cette fonction est le double-clic sur une

- **Ajouter à la table:** sélectionnez une ou plusieurs variables (avec Ctrl ou Maj) et ajoutez là dans l'une des tables créées.
- **Enlever de la table:** cliquez sur cette option pour supprimer une ou plusieurs variables (avec Ctrl ou Maj) d'une table, mais pas du projet.
- **Supprimer:** sélectionnez une ou plusieurs variables (avec Ctrl ou Maj) et cliquez sur cette option pour les supprimer. Les variables ne peuvent pas être supprimées lorsqu'elles sont utilisées dans le programme. On peut également les supprimer à partir de la barre d'outils.
- **Références croisées :** cliquez sur cette option pour afficher une fenêtre *popup* indiquant tous les endroits où une variable spécifique est utilisée. Par exemple, la capture d'écran ci-dessous indique les références croisées pour la variable %M1 (elle n'est utilisée qu'une seule fois dans le programme: dans la page 1 dans la fonction à la ligne 2).



Capture d'écran de la fenêtre *popup* « Référence croisées pour %M1 »

5.5.1.1.4 Communication (Alt+C)

Ce menu déroulant est similaire au menu déroulant « Communication » de la barre de menu principal de MicroLADDER.

Information

Cet onglet est identique à celui que l'on trouve sur la fenêtre principale dans MicroLADDER. Ce menu déroulant ne sera pas accessible à partir d'un bloc fonction mais seulement

- **Se connecter:** cette option est disponible lorsqu'aucune connexion n'a encore été établie. Une fenêtre *popup* « Connexion à MicroDRIVER » apparaîtra lorsque l'on clique sur cette option. Plusieurs champs (méthode de communication, le numéro d'esclave et l'adresse du MicroDRIVER) sont à renseigner afin de se connecter à l'automate. On peut également accéder à la fenêtre « Connexion à MicroDRIVER en utilisant le bouton « Se connecter » placé dans le menu Outils. Pour plus d'informations sur comment établir une connexion, se référer à la section [5.3.2 Établissement de la connexion](#).

Capture d'écran de la fenêtre *popup* « Connexion à MicroDRIVER »

- **Déconnecter**: cette option est disponible lorsqu'une connexion a déjà été établie. En cliquant sur cette fonction la communication avec l'automate va être arrêtée.
- **Lecture du numéro d'esclave** : cette option est uniquement disponible lorsqu'une connexion avec un automate a été établie. Elle est utilisée pour obtenir le numéro d'esclave d'un esclave spécifique, lorsqu'il est prévu d'en connecter plusieurs en série, sur la même ligne.
- **Écriture du numéro d'esclave** : cette option est uniquement disponible lorsqu'une (et une seule) connexion avec un automate a été établie. Elle permet de modifier son numéro d'esclave.
- **Démarrage du programme** : cette option démarre un programme lorsque l'automate est à l'état STOP. Une fenêtre *popup* nous indiquant que le programme a bien démarré apparaît.
- **Arrêt du programme** : cette fonction arrête un programme lorsque l'automate est à l'état RUN.
- **Réinitialisation de l'équipement** : cette option réinitialise l'automate à son état BOOT. Elle est utile pour tester ses réactions lorsqu'il est allumé et éteint par exemple.
- **Réinitialisation des variables** : cette fonction remet toutes les variables à leur valeur initiale et n'arrête pas le programme. L'automate est toujours dans l'état RUN. Elle est utile pour simuler un comportement.
- **Chargement d'un programme** : cette option ouvre une fenêtre *popup* « Chargement d'un programme » nous permettant de choisir le fichier hexadécimal (.hex) que l'on souhaite charger sur l'automate. Cette action prend plusieurs dizaines de secondes, il est donc plus rapide de charger à l'aide d'une carte SD si l'automate dispose d'un lecteur de carte SD.

Information :

Il est possible de connaître la touche correspondante associée aux raccourcis

5.5.1.2 Barre d'outils de l'éditeur de variables

La barre d'outils de l'éditeur de variables apparaît en haut de la fenêtre de l'éditeur de variables. Il comprend les éléments suivants:



Capture d'écran de la barre d'outils de l'éditeur de variables

Les éléments suivants sont inclus dans la barre d'outils de l'éditeur de variables :

Variables utilisateur ▼

Variables utilisateur/Variables système/Tables utilisateur créées: sélectionnez le type de variables que vous souhaitez afficher dans la liste déroulante : variables système, variables utilisateur ou l'une des tables que vous avez éventuellement créées.



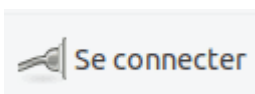
Ajouter: cliquez sur cette option pour inclure une nouvelle variable au projet MicroLADDER en cours. Donnez un nom à cette nouvelle variable (soit une adresse ou une mnémonique) et définissez ses propriétés dans la nouvelle fenêtre *popup* de propriétés qui apparaît (voir section [5.5.1.3 Propriétés variables](#))



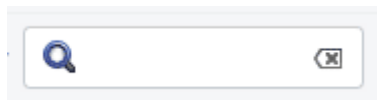
Retirer: cette option supprime une variable d'une table, mais pas du projet.



Effacer: sélectionnez une variable (ou des variables en les sélectionnant avec Ctrl ou Maj) puis cliquez dessus afin de la/les supprimer. Les variables ne peuvent pas être supprimées lorsqu'elles sont utilisées dans le programme.



Se connecter: cliquez sur cette option pour se connecter à un automate lorsqu'il est déconnecté ou pour le déconnecter lorsqu'il est connecté. Lors de la connexion, une nouvelle fenêtre apparaîtra en cliquant sur cette option. Il est alors demandé de définir les paramètres appropriés pour la méthode de communication, le numéro d'esclave et l'adresse du MicroDRIVER afin de se connecter à l'automate. Pour se déconnecter, cliquez simplement sur cette option tant que la connexion avec l'automate est active. On peut également accéder à la fenêtre de connexion à MicroDRIVER depuis l'onglet « Communication » situé sur la barre de menu de l'éditeur de variables.



Rechercher: utilisez cette barre de recherche pour rechercher une variable spécifique soit par son adresse (comme par exemple « %S1 ») soit par son mnémonique. La recherche n'est pas sensible à la casse. Il faut s'assurer que l'on se trouve dans le bon type de variable (si on recherche une variable de contrôle de cycle et que l'on a sélectionné les variables utilisateur, la recherche ne renverra aucun résultat)

5.5.1.3 Propriétés variables

Les informations suivantes sont affichées lorsque l'on crée une variable ou si on l'a modifie. Pour y accéder double-cliquez sur la variable ou bien clic-droit puis sélectionner « Propriétés ».

5.5.1.3.1 Onglet Général

- **Type** : type de variable. Pour mieux comprendre les préfixes utilisés pour l'adresse des variables, voir section [6.2 Types de données](#).
- **Adresse** : adresse mémoire de la variable dans l'automate. Pour une meilleure compréhension des préfixes utilisés pour l'adresse des variables, voir section [6.2 Types de données](#).
- **Mnémonique** : nom donné à la variable.
- **Commentaire** : brève explication de la variable.
- **Nombre d'éléments** : dans le cas d'un tableau, nombre d'éléments alloués dans le tableau. Si la variable n'est pas un tableau, cette case sera grisée.
- **Nombre de caractères** : dans le cas d'une chaîne, nombre maximum de caractères de la chaîne de caractères. Si la variable n'est pas une chaîne de caractères, cette case sera grisée.

Capture d'écran de l'onglet « Général » de la fenêtre *popup* « Propriétés de la variable »

5.5.1.3.2 Onglet Programmation

- **Valeur initiale** : valeur chargée sur la variable au démarrage de l'application ou lors d'une requête d'initialisation.
- **Sauvegardée** : si l'automate dispose d'une mémoire sauvegardée, vous pouvez sélectionner cette option et la valeur sera conservée pendant l'interruption de l'alimentation. L'utilisateur peut déterminer l'instant de la sauvegarde sur EEPROM.
- **Timer (en ms)** : période de temps au bout de laquelle la variable est décrétementée de 1.

En plus de ces fonctionnalités, si le programme est un bloc fonction alors on peut remplir ces champs:

- **Globale** : cette propriété peut être utilisée avec des fonctions. Elle permet à la variable de conserver sa valeur entre deux appels.
- **Paramètre** : utilisé pour la déclaration des variables d'un bloc fonction. Voir la section [5.8 Créer un bloc fonction](#) pour plus de renseignements.

- **Position** : permet de définir l'ordre des entrées/sorties dans une fonction.
- **Libellé** : permet de donner un nom à une variable lorsqu'elle apparaît dans une fonction.

Capture d'écran de l'onglet « Programmation » de la fenêtre Propriétés de la variable

5.5.1.3.3 Onglet Entrée/Sortie

Cet onglet n'est pas disponible pour des programmes bloc fonction.

- **Configuration** : paramétrage des entrées analogiques
- **Variable associée** : uniquement pour les variables d'entrée/sortie qui correspondent à des éléments connectés. Cette fonctionnalité associe une variable utilisateur à une variable matérielle.
- **Inverser l'état** : disponible uniquement lors de l'utilisation d'une variable liée booléenne. Cette fonctionnalité permet d'inverser l'état entre la variable locale et la variable distante.

La mise à l'échelle est uniquement disponible lors de l'utilisation d'une variable liée non booléenne. Il est possible de mettre à l'échelle des valeurs, et pour cela de spécifier une échelle de valeur brute et

de lui associer une plage de valeur mise à l'échelle.

- **Valeur brute min.** : valeur brute minimale pour l'entrée analogique
- **Valeur brute max.** : valeur brute maximale pour l'entrée analogique
- **Valeur mise à l'échelle min.** : valeur mise à l'échelle minimale pour la variable liée
- **Valeur mise à l'échelle max.** : valeur mise à l'échelle maximale pour la variable liée

Un exemple est que si on a capteur renvoyant des données comprises entre 0 et 20000 (pour 0-20mA) alors ces valeurs se reportent à des températures comprises entre -20°C et +80°C (0 représentant -20°C et 20000 représentant 80°C). Dans le champs « Variable associée » associez une variable liée de type %MF.

Pour mettre en place la mise à l'échelle, remplissez le champs « Valeur brute min » par 0, le champs « Valeur brute max » par 20000, le champs « Valeur mise à l'échelle min. » par -20 et le champs « Valeur mise à l'échelle max. » par 80.

Propriétés de la variable

Général Programmation **Entrée / Sortie** Communication Historisation Serveur distant Affichage

Configuration : Aucune

Variable associée :

☐ Inverser l'état

Mise à l'échelle

Valeur brute min. :

Valeur brute max. :

Valeur mise à l'échelle min. :

Valeur mise à l'échelle max. :

Annuler Valider

Capture d'écran de l'onglet Entrée/Sortie » de la fenêtre *popup* « Propriétés de la variable »

5.5.1.3.4 Onglet Communication

- **Accès à distance:**
 - « Aucun » : pas de communication pour cette variable
 - « Lecture seule » : le cycle de communication est une trame de lecture périodique. La variable dans l'automate est une copie de la variable dans l'équipement distant.
 - « Écriture seule » : le cycle de communication est une trame de lecture périodique. La variable dans l'équipement distant est une copie de la variable dans l'automate.
 - « Lecture / écriture » : le cycle de communication est une trame de lecture périodique et une trame d'écriture lorsque nécessaire. Chaque fois que la variable change dans l'automate, la valeur est écrite dans l'appareil distant. Chaque fois que la variable change dans l'équipement distant, la valeur est mise à jour dans l'automate.
- **Adresse distante** : adresse de la variable dans l'équipement distant. L'adresse doit spécifier le nom de la variable suivi d'un point et du numéro de l'équipement. Par exemple « %MW2.3 » est une variable %MW à l'adresse 2 de l'équipement distant numéro 3.
- **Inverser l'état** : disponible uniquement lors de l'utilisation d'une variable liée booléenne. Cette fonctionnalité permet d'inverser l'état entre la variable locale et la variable distante.

Il est possible de faire une mise à l'échelle des valeurs, et pour cela, spécifier une échelle de valeur brute et lui associer une plage de valeur mise à l'échelle.

- **Valeur brute min.** : valeur brute minimale dans l'appareil distant
- **Valeur brute max.** : valeur brute maximale dans l'appareil distant
- **Valeur mise à l'échelle min.** : valeur mise à l'échelle minimale dans l'automate
- **Valeur mise à l'échelle max.** : valeur mise à l'échelle maximale dans l'automate

Un exemple est il y a des données comprises entre 0 et 100 dans l'équipement distant et que l'on veut que ces valeurs soient comprises entre 0 et 10 dans l'automate alors il faut remplir le champs « Valeur brute min » par 0, le champs « Valeur brute max » par 100, le champs « Valeur mise à l'échelle min. » par 0 et le champs « Valeur mise à l'échelle max. » par 10.

Ainsi, par exemple lorsque la valeur est 233 dans l'équipement distant, la valeur est 23,3 dans l'automate et vice versa.

Capture d'écran de l'onglet « Communication » de la fenêtre *popup* « Propriétés de la variable »

5.5.1.3.5 Onglet Historisation

- **Condition de l'alarme** : menu déroulant permettant de choisir l'opérateur logique de la condition de l'alarme (<, <=, >, >=, != et ==). Si on choisit « Aucune » alors il n'y aura pas d'alarmes pour cette variable.
- **Seuil de l'alarme** : valeur de la condition avec laquelle l'alarme est activée. Une alarme est un état, et un événement se produit et est créé chaque fois qu'un seuil est dépassé.
- **Libellé d'apparition de l'alarme** : texte qui apparaîtra lorsque la condition d'alarme est présente.
- **Libellé de disparition de l'alarme** : texte qui apparaîtra lorsque la condition d'alarme disparaîtra.
- **Condition de l'événement** : menu déroulant permettant de choisir l'opérateur logique de la condition de l'événement (<, <=, >, >=, != et ==). Si on choisit « Aucune » alors il n'y aura pas d'événements pour cette variable.
- **Seuil de l'événement** : valeur de la condition avec laquelle un événement sera activé.
- **Libellé de l'événement** : texte qui apparaîtra lorsque l'événement est présent.

- **Type d'historisation** : menu déroulant permettant de choisir comment on souhaite archiver les courbes de valeur dans le temps. Il y a trois types d'historisations :
 - « Aucune » : aucune historisation effectuée.
 - « Standard » : prend une valeur avec un intervalle régulier
 - « Moyennée » : prend une valeur moyenne sur une période de temps donnée
- **Délai d'historisation (en secondes)** : délai d'archivage. Il dépend du type de journalisation parmi deux possibles:
 - « Standard » : temps spécifié (1 minute par exemple)
 - « Moyennée » : période de temps spécifiée
- **Seuil d'historisation** : seuil au-delà duquel la valeur sera enregistrée.

Propriétés de la variable

Général

Programmation

Entrée/Sortie

Communication

Historisation

Serveur distant

Affichage

Alarme

Condition de l'alarme :

Aucune

Seuil de l'alarme :

0

Libellé d'apparition de l'alarme :

Libellé de disparition de l'alarme :

Événement

Condition de l'événement :

Aucune

Seuil de l'événement :

0

Libellé de l'événement :

Courbe

Type d'historisation :

Aucune

Délai d'historisation (en secondes) :

0

Seuil d'historisation :

0

Annuler

Valider

Capture d'écran de l'onglet « Historisation » de la fenêtre *popup* « Propriétés de la variable »

5.5.1.3.6 Onglet Serveur distant

Information

Cet onglet est utilisé pour communiquer avec un serveur MicroSERVER distant

- **Accès** : menu déroulant permettant d'indiquer si la variable est accessible en lecture et en écriture par MicroSERVER.
 - « Privé » : la variable n'est pas accessible
 - « Lecture seule » : la valeur de la variable va pouvoir être lue par MicroSERVER
 - « Écriture seule » : la valeur de la variable va pouvoir être écrite par MicroSERVER
 - « Lecture / écriture » : la valeur de la variable va pouvoir être lue et écrite par MicroSERVER

Les trois champs suivants ne seront pas disponibles si on a sélectionné « Privé ».

- **Libellé** : texte qui sera affiché dans le champs « Label » sur MicroSERVER pour cette variable
- **Valeur de forçage min.** : valeur minimale que peut avoir la variable lorsque que l'on veut la forcer
- **Valeur de forçage max.** : valeur maximale que peut avoir la variable lorsque que l'on veut la forcer

The screenshot shows a software window titled "Propriétés de la variable" with a close button (X) in the top right corner. It features several tabs: "Général", "Programmation", "Entrée / Sortie", "Communication", "Historisation", "Serveur distant" (which is the active tab), and "Affichage". The "Serveur distant" tab contains four labeled input fields: "Accès :", "Libellé :", "Valeur de forçage min. :", and "Valeur de forçage max. :". The "Accès :" field is a dropdown menu currently set to "Privée". The other three fields are empty text boxes. At the bottom right of the window, there are two buttons: "Annuler" and "Valider".

Capture d'écran de l'onglet « Serveur distant » de la fenêtre *popup* « Propriétés de la variable »

5.5.1.3.7 Onglet Affichage

Ces paramètres sont utilisés dans l'interface graphique (MicroHMI), dans le serveur (MicroSERVER) et dans l'affichage de la valeur formatée dans MicroLADDER. Les différents champs sont disponibles ci-dessous :

- **Représentation** : menu déroulant permettant de sélectionner comment la valeur de la variable sera représentée parmi:
 - décimal
 - binaire
 - hexadécimal
- **Format** : format de la valeur affichée. La valeur de la variable est représentée par « %v ». Ainsi si par exemple on écrit « %v°C » alors la valeur de la variable sera affichée suivie de l'unité « °C ».
- **Précision** : chiffre indiquant le nombre de chiffres après la virgule d'un nombre décimal. -1 signifie aucune limite c'est à dire tous les chiffres après la virgule.
- **Valeur d'affichage min.** : valeur minimale à afficher d'une variable pour les synoptiques, jauges, etc. dans MicroHMI.
- **Valeur d'affichage max.** : valeur maximale à afficher d'une variable pour les synoptiques, jauges, etc. dans MicroHMI.

Propriétés de la variable

Général Programmation Entrée/Sortie Communication Historisation Serveur distant **Affichage**

Représentation : Décimale

Format : |

Précision : -1

Valeur d'affichage min. : 0

Valeur d'affichage max. : 0

Annuler Valider

Capture d'écran de l'onglet « Affichage » de la fenêtre *popup* « Propriétés de la variable »

5.5.2 Type de variable

Les variables considérées dans un programme MicroLADDER peuvent être divisées en deux catégories:

- variables système
- variables utilisateurs

5.5.2.1 Variables système

Les variables système sont inhérentes à l'automate. Leurs propriétés ne peuvent pas être modifiées ainsi l'éditeur de variables n'affichera que leur valeur en temps réel.

Ces variables peuvent être classées en douze tables :

- Informations système : %SW13, %SW14, %SW23, %SW24
- Contrôle du cycle : %S0, %S1, %S16
- chien de garde : %S15, %S20 à %S22, %SW0 à %SW2, %SW4

- Horodateur : %S5 à %S8, %S24, %SW5 à %SW12
- Gestion des variables : %S2, %S18, %SW15 à %SW22
- Interruption : %SW25
- Ports séries : %SW34 à %SW81
- Ethernet : %S25, %SW96 à %SW137, %SW225 à %SW262
- WiFi : %S26, %SW91 à %SW95, %SW138 à %SW175, %SW189
- Radio : %SW190 à %SW191, %SW200 à %SW211
- Bluetooth : %SW214 à %SW229
- Gestion des entrées / sorties : %S11, %S12, %SW26

5.5.2.2 Variables utilisateur

L'utilisateur peut avoir besoin de variables supplémentaires dans un programme. Si aucune variable n'a encore été créée par l'utilisateur alors aucune variable ne sera affichée dans les variables utilisateur dans l'éditeur de variables. Les variables déjà créées dans le programme (en Ladder ou en C) sont créés par le logiciel MicroLADDER et seront donc affichées dans l'éditeur de variables. Cependant, une saisie manuelle est nécessaire pour définir les propriétés de la variable.

5.5.2.3 Création de variables

5.5.2.3.1 Création de variables automatiquement

Il est possible de créer des variables automatiquement dans un programme en Ladder ou en C. Pour cela il suffit d'écrire le type de la variable suivi de l'adresse. Par exemple si on veut créer une variable de type entier non signé sur 16 bits à l'adresse 1, alors on va écrire %MW1. Ceci génère alors automatiquement la variable dans la table des variables.

Avertissement

Cette manière de créer des variables est rapide mais on peut vite s'y perdre

5.5.2.3.2 Création de variables manuellement

Pour créer des variables ayant des caractéristiques plus précises que leur *type+adresse* alors on va créer des variables manuellement depuis l'éditeur de variables.



Pour créer des variables manuellement, cliquez sur l'icône « Ajouter », choisissez le type puis donnez un mnémonique à la variable et déterminez ses propriétés (voir la section [5.5.1.3 Propriétés variables](#)) dans la nouvelle fenêtre *popup* qui apparaît.

Avertissement

Quand on écrit un mnémonique, il n'est pas indispensable de définir une adresse

Capture d'écran de la fenêtre *popup* « Propriétés de la variable »

5.5.2.4 Identification des variables

La variable est identifiée par la combinaison de *type*+*adresse*.

Par exemple, si le type est booléen et l'adresse est 10. La variable est identifiée comme %M10. Plusieurs types de variables communes aux blocs fonction et aux programmes automates sont disponibles parmi :

- booléen
- entier non signé 16 bits
- entier signé 32 bits
- flottant signé 32 bits
- chaîne de caractères

Si le programme n'est pas un bloc fonction, ces variables sont disponibles additionally:

- entrées TOR
- entrées analogiques
- sorties TOR
- sorties analogiques

L'adresse est le décalage de la variable dans la mémoire de l'automate. L'adresse doit être unique pour chaque variable du même type. Par exemple, il n'est pas permis de définir deux variables %M0 mais une variable %M0 et une autre %MW0 sont autorisées. La définition d'une adresse n'est pas obligatoire. Dans ce cas, une compilation localisera la variable dans un espace libre de la zone de variables du même type.

Il est nécessaire si une variable doit remonter vers un serveur ou si l'automate doit être paramétré en esclave Modbus, de faire venir un autre équipement lire/écrire cette variable.

Les variables peuvent être utilisées dans un programme MicroLADDER en les appelant par leur adresse ou leur mnémonique.

Propriétés de la variable

Général

Programmation

Entrée / Sortie

Communication

Historisation

Serveur distant

Affichage

Type :

Booléen (%M)

Adresse :

1

Mnémonique :

Init

Commentaire :

Nombre d'éléments :

Nombre de caractères :

Annuler

Valider

Par exemple, dans notre programme, si on souhaite utiliser cette variable, on peut écrire soit %M1 ou soit Init. Les deux écritures fourniront le même résultat. Par exemple, ces deux lignes mettent la variable à 0:

```
%M1 = 0;  
Init = 0;
```

5.5.3 Exporter/Importer des variables

5.5.3.1 Exporter des variables

La liste variables peut être exportée vers un fichier CSV et importées à partir d'un fichier CSV pour être utilisée dans d'autres programmes.

Pour exporter des variables, allez dans l'onglet « Tables » puis « Exporter les variables de cette table ». Une fenêtre *popup* s'ouvre nous demandant de préciser la structure du fichier CSV. Il est préférable de ne pas modifier la structure par défaut pour éviter d'éventuels problèmes d'import. Si on la change lors de l'exportation, il faut s'assurer de respecter le même ordre lors de l'importation des variables. Cliquez ensuite sur « Valider ».

×

Exporter un fichier CSV

Format CSV

☒ Ajouter une ligne d'en-têtes

Séparateur :

;

Position des champs

Adresse :

1

Mnémonique :

2

Commentaire :

3

Valeur :

4

Valeur formatée :

5

Forcée :

6

Nombre d'éléments :

7

Nombre de caractères :

8

Valeur initiale :

9

Sauvegardée :

10

Timer (en ms) :

11

Globale :

12

Annuler

Valider

Capture d'écran de la fenêtre *popup* « Exporter un fichier CSV »

5.5.3.2 Modifier la tables des variables dans un fichier CSV

Une fois la liste des variables exportée sous format CSV, il est tout à fait possible de modifier cette table en ouvrant le fichier (avec le logiciel Excel ou LibreOffice Calc). Cette démarche est utile notamment lorsque que l'on veut créer un grand tableau. On crée notre tableau puis la variable correspondant à la première case du tableau. Après l'export des variables, on ouvre le fichier CSV puis on a juste a faire un copier-coller de la variable de la première case du tableau et changer à chaque fois l'adresse pour créer les variables des autres cases du tableau. On garde ainsi les mêmes attributs pour toutes les cases (pour plus d'informations sur les tableaux se référer à la section [6.2.11 Tableaux](#))

5.5.3.3 Importer des variables

Pour importer des variables dans un programme, allez dans l'onglet « Tables » puis sur « Importer des variables dans cette table ». Une fenêtre *popup* permettant de sélectionner le fichier CSV s'ouvre. Une fois le fichier sélectionné, cliquez sur « Valider » une nouvelle fenêtre nommée « Importer un fichier CSV » s'ouvre. Il n'est pas utile de modifier quelque chose sur cette fenêtre. Enfin, cliquez à nouveau sur « Valider ».

L'import de variables est effectué en écrasant les variables actuelles, assurez vous donc que vous n'avez pas dans votre projet actuel une variable avec la même adresse ou mnémonique, car elle sera directement remplacée.

Importer un fichier CSV

Format CSV

☒ Ignorer la première ligne

Séparateur : ;

Position des champs

Adresse : 1

Mnémonique : 2

Commentaire : 3

Valeur : 4

Valeur formatée : 5

Forcée : 6

Nombre d'éléments : 7

Nombre de caractères : 8

Valeur initiale : 9

Sauvegardée : 10

Timer (en ms) : 11

Globale : 12

Annuler Valider

Capture d'écran de la fenêtre *popup* « Importer un fichier CSV »

5.5.4 Multi-édition variable

Il est possible de modifier les propriétés de plusieurs variables ensemble au lieu de modifier chacune des variables séparément.

Pour cela il faut suivre la démarche suivante:

- Sélectionnez les variables avec Ctrl ou Maj.
- Cliquez ensuite sur l'onglet « Variables » puis sur « Propriétés » ou bien faites un clic-droit depuis la fenêtre de l'éditeur de variables puis cliquez sur « Propriétés »
- Une fenêtre *popup* « Propriétés des variables » s'ouvre. Les propriétés qui ont la même valeur sur toutes les variables apparaissent sélectionnées et encadrées en bleu tandis que les propriétés qui ont une valeur différente apparaissent non sélectionnées et grises.
- Sélectionnez les champs que vous souhaitez modifier et définissez la nouvelle valeur. Toutes les propriétés sélectionnées seront appliquées à toutes les variables. Les propriétés non sélectionnées resteront inchangées.
- Validez votre modification en cliquant sur « OK ». Sinon cliquez sur « Annuler » ou bien fermer la fenêtre.

Avertissement

Il est important de ne pas modifier les propriétés des variables qui doivent être

Propriétés des variables

Général Programmation Entrée / Sortie Communication Historisation Serveur distant Affichage

Type : Entier non signé 16 bits (%MW) ☒

Adresse :

Mnémonique :

Commentaire :

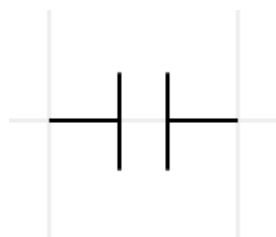
Nombre d'éléments : 0 ☒

Nombre de caractères :

Annuler Valider

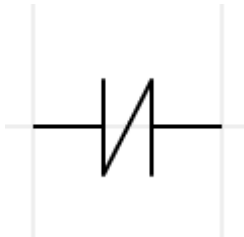
Capture d'écran de la fenêtre *popup* « Propriétés des variables »

5.6 Objets disponibles en langage Ladder



5.6.1 Contact ouvert

Il est activé lorsque la valeur de la variable représentée (entrée, variable interne ou bit système) est égale à 1. Il est désactivé lorsque sa valeur est égale à 0. Il peut être utilisé soit avec une variable binaire, soit comme comparaison du type $\%MW0 > 0$ avec des variables entières, longues ou flottantes. Cela signifie que chaque valeur de variable différente de 0 activera le contact.



5.6.2 Contact fermé

Il est activé lorsque la valeur de la variable représentée (entrée, variable interne ou bit de système) est 0. Il est désactivé lorsque sa valeur est 1. Les mêmes remarques que celles déjà faites pour le contact ouvert peuvent être faites.



5.6.3 Front montant

Il est activé lorsqu'il y a un changement de la valeur de la variable représentée (entrée, variable interne ou bit système) de 0 à 1.

Chaque front est géré par une variable interne (de taille 1 octet) indépendante de la variable interne utilisée. Le front est valide exactement pendant un cycle complet. Si la variable change après le front alors lors du cycle suivant, le front le détectera facilement.

Il y a potentiellement un bug avec les bits indexés, si l'indice a varié d'un cycle à l'autre.

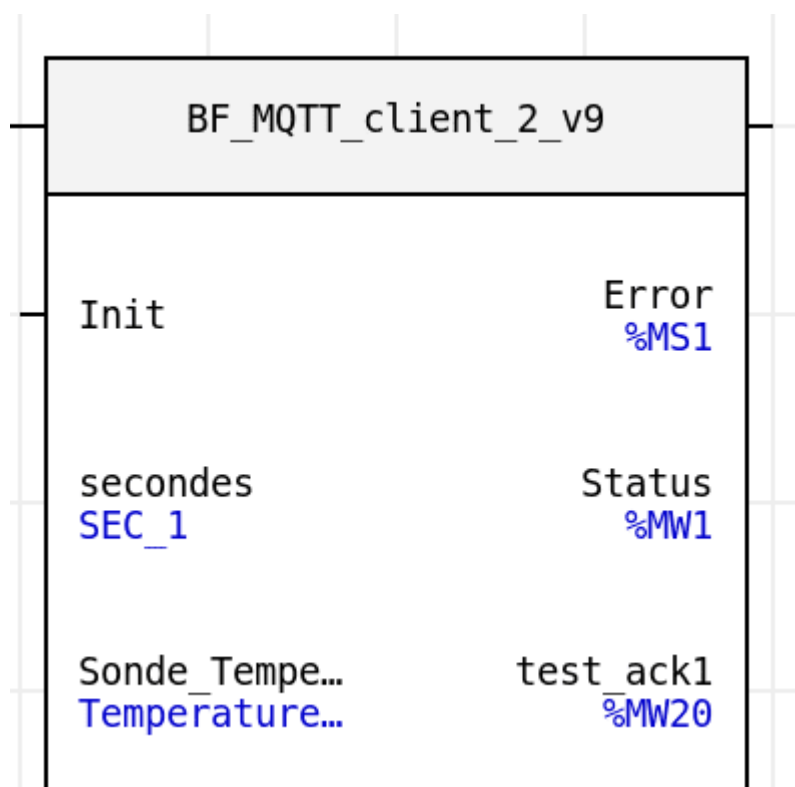
Les expressions complexes et les bits mots peuvent être gérés.



5.6.4 Front descendant

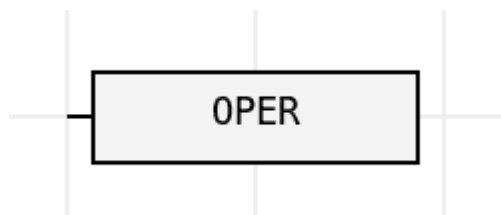
Il est activé lorsqu'il y a un changement de la valeur de la variable représentée (entrée, variable interne ou bit système) de 1 à 0.

Les mêmes remarques que celles déjà faites pour le front montant peuvent être faites.



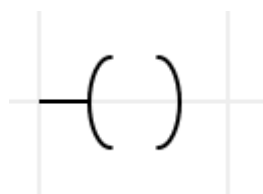
5.6.5 Fonction

Lorsqu'une fonction est importée dans le programme en cours, elle peut être insérée comme une instruction Ladder. Il suffit alors de définir les variables d'entrée et de connecter les sorties correctement.



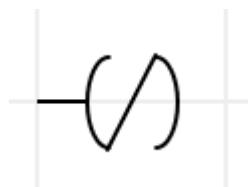
5.6.6 Opération

Ce bloc va permettre de définir des opérations entre variables.



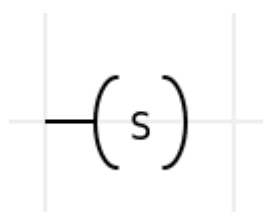
5.6.7 On

Il est activé lorsque la combinaison de gauche donne un résultat de 1.
Son activation signifie qu'il a va renvoyer une valeur logique de 1.



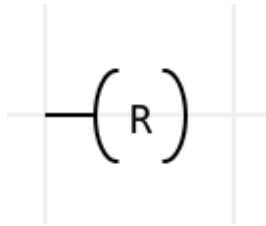
5.6.8 Off

Il est activé lorsque la combinaison de gauche donne un résultat de 0.
Son activation signifie qu'il a va renvoyer une valeur logique de 0.



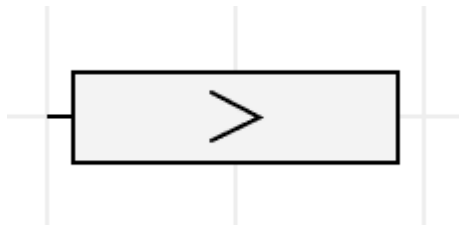
5.6.9 Set

Cet élément va forcer la variable associée à 1. La variable ne peut être mise à 0 qu'avec un « Reset ».
Il est généralement utilisé pour le stockage des bits.



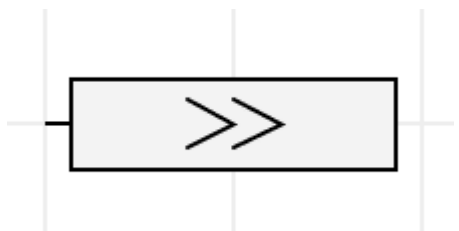
5.6.10 Reset

Cet élément va mettre à 0 la variable associée.
Il désactive une variable précédemment activée avec un « Set ».



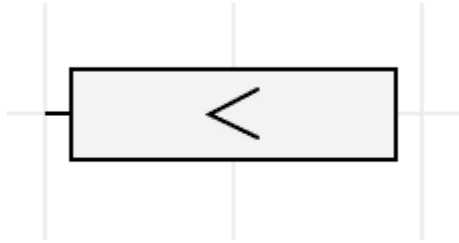
5.6.11 Appel

Ce bloc fait appel à une page du programme et exécute son code. Une fois le code exécuté, le programme principal reprend à la ligne suivante ou l'appel a été fait.



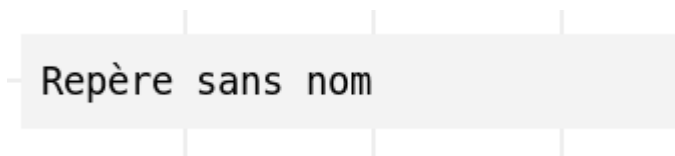
5.6.12 Saut

Ce bloc permet d'avancer en sautant un certains nombres d'instructions.



5.6.13 Retour

Ce bloc permet de retourner en arrière en sautant un certains nombres d'instructions.



5.6.14 Repère

Ce bloc est utilisé par les deux dernières instructions pour matérialiser l'endroit où s'arrête le Saut ou le Retour dans le programme.



5.6.15 Commentaire

Ce bloc permet d'ajouter un bloc de commentaire dans le programme.

5.6.16 Insérer une ligne vide

Cette commande permet d'ajouter des lignes vides pour faire de l'espace dans le programme.

5.6.17 Supprimer les lignes vides

Cette commande va supprimer les lignes vides

5.6.18 Coller

Cette commande permet de coller un élément que l'on a préalablement copié.

5.6.19 Connecter des éléments

Pour connecter des éléments entre eux, il suffit de double-cliquer sur une ligne du quadrillage de fond gris d'un programme où une connexion doit être établie ou bien cliquez et faites glisser d'un élément à un autre (ou à une ligne).

5.6.20 Déconnecter des éléments

Pour déconnecter des éléments entre eux, cliquez avec le bouton droit de la souris sur le connecteur et sélectionnez « Déconnecter » ou double-cliquez sur la connexion que vous souhaitez supprimer.

5.7 Créer un programme

Pour mieux comprendre comment fonctionne MicroLADDER, quelques exemples de programme sont présentés ci-dessous:

5.7.1 Créer un premier programme simple en LADDER

Afin d'avoir une première approche, on va programmer un automate pour exécuter une tâche simple. Notre système va se composer de:

- un automate MicroArm-A1
- l'écran de l'automate MicroArm-A1 dont le rétro-éclairage LED est une entrée à l'adresse %Q0
- un bouton poussoir: entrée numérique à l'adresse %I100
- une LED: sortie numérique à l'adresse %Q100

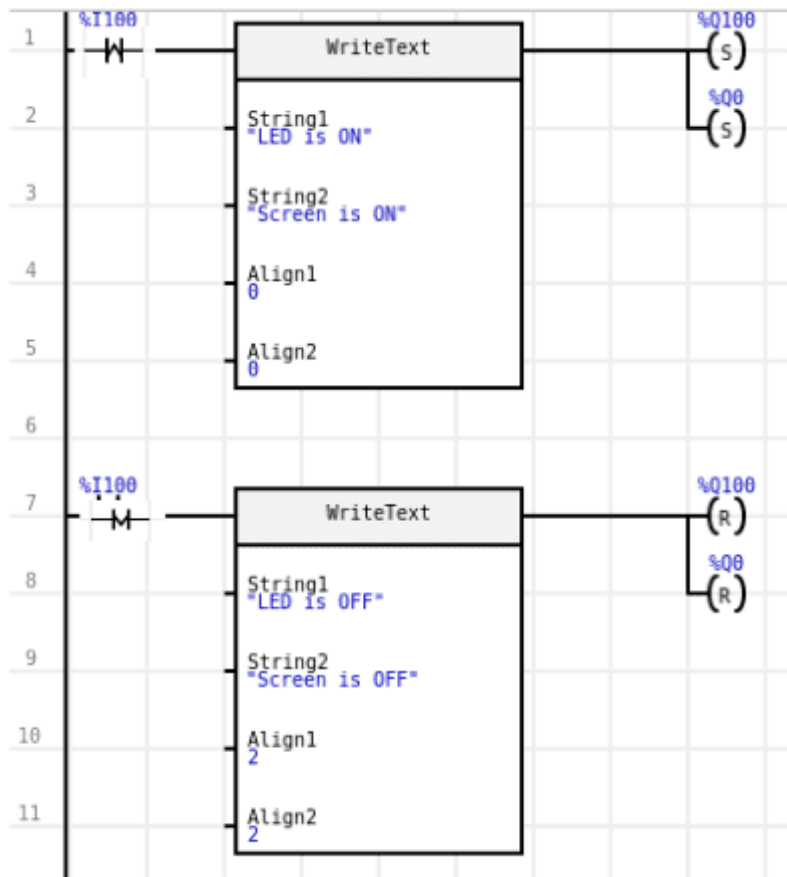
Information

Les valeurs de ces adresses ne sont pas données au hasard mais sont indiquées

Notre programme va devoir respecter le cahier des charges:

- La LED ne sera allumée que lorsqu'on appuie sur le bouton poussoir. Sinon, elle restera éteinte
- Quand la LED est allumée (ce qui signifie que le bouton poussoir est poussé), l'écran du MicroARM-A1 a aussi le rétro-éclairage allumé et le texte affiché est le suivant :
 - Première ligne : « LED is on » positionné sur le côté gauche de l'écran
 - Deuxième ligne : « Screen is on » positionné sur le côté gauche de l'écran
- Lorsque la LED est éteinte (ce qui signifie que le bouton poussoir est relâché), l'écran du MicroARM-A1 a le rétro-éclairage éteint et le texte affiché est le suivant :
 - Première ligne : « LED is OFF » positionné sur le côté droit de l'écran
 - Deuxième ligne : « L'écran est éteint » positionné sur le côté droit de l'écran

Pour écrire ce programme, la fonction *WriteText* est utilisée. La capture d'écran suivante montre ce programme déjà implémenté dans le MicroLADDER.



Capture d'écran de la fenêtre principale de MicroLADDER avec la fonction *WriteText*

Interprétation

Quand on pousse le bouton...
Quand on éteint le bouton...
...LED ou écran à ON
...LED ou écran à OFF

Interprétation Digitale

%I100 change de 0 à 1
%I100 change de 1 à 0
%Q100 ou %Q0 est mis à 1 Set
%Q100 ou %Q0 est à mis 0 Reset

Ladder

Front montant
Front descendant

Tableau sur l'interprétation textuelle en interprétation digitale et Ladder

5.7.2 Créer un premier programme simple en C

On va créer le même programme que précédemment (même système et même cahier des charges) mais le code sera du C.

```
if (%I100 == 1)
{
    %Q100=1;
    %Q0=1;
    WriteText("LED is ON","Screen is ON",0,0);
}
else
{
    %Q100=0;
    %Q0=0;
}
```

```
WriteText("LED is OFF","Screen is OFF",2,2);
}
```

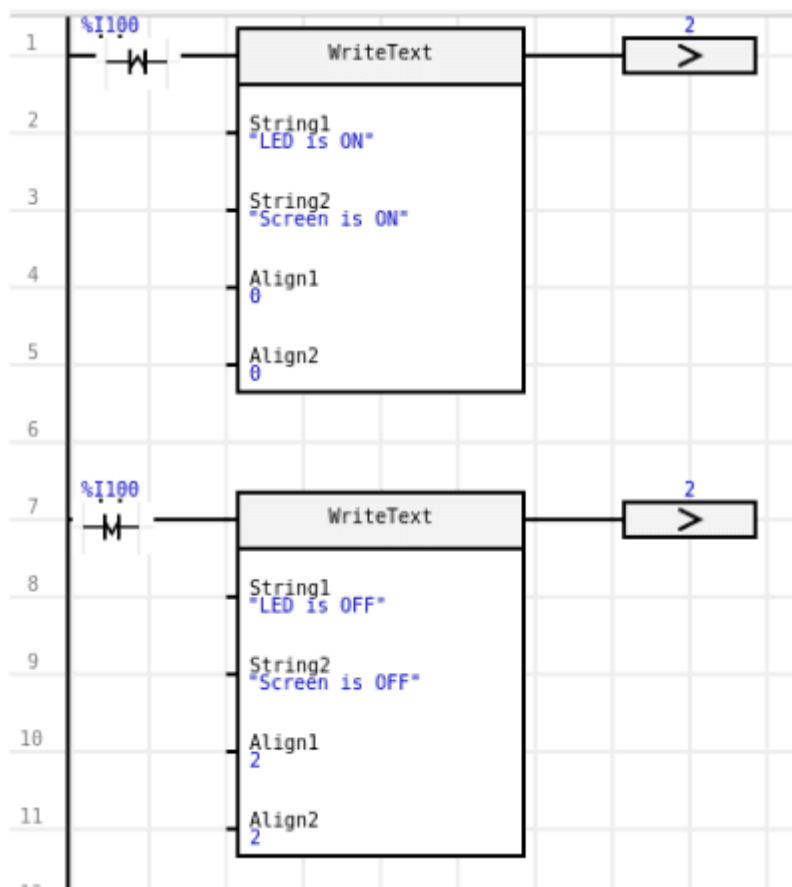
5.7.3 Créer un premier programme simple combinant Ladder et C

Comme nous l'avons déjà mentionné dans les sections précédentes, il est également possible de combiner un programme contenant à la fois du code Ladder et du code C en le structurant dans des pages différentes. Le même programme que celui expliqué précédemment est divisé en deux pages comme suit :

- Une première page comprend le programme qui affiche le texte à l'écran lorsque l'on appuie sur le bouton poussoir.
- Une deuxième page contient le programme qui active la LED et le rétro-éclairage de l'écran LED. Cela peut se faire de deux manières :
 - Première méthode: le programme principal est écrit en Ladder et appelle une page C.
 - Deuxième méthode: le programme principal est écrit en code C et il appelle une page Ladder.

5.7.3.1 Première méthode

Contenu de la page 1:



Capture d'écran du contenu Ladder de la fenêtre principale de MicroLADDER

Contenu de la page 2:

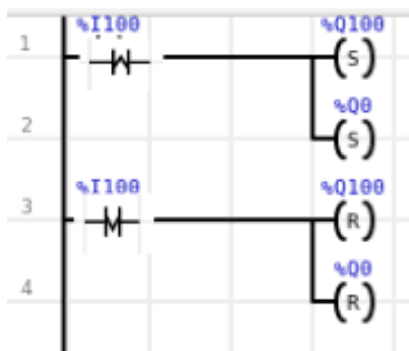
```
if (%I100==1)
{
    %Q100=1;
    %Q0=1;
}
else
{
    %Q100=0;
    %Q0=0;
}
```

5.7.3.2 Deuxième méthode

Contenu de la page 1:

```
if (%I100 == 1)
{
    WriteText("LED is ON","Screen is ON", 0,0);
    page_2();
}
else
{
    WriteText("LED
page_2();
}
```

Contenu de la page 2:



Capture d'écran du contenu Ladder de la fenêtre principale de MicroLADDER

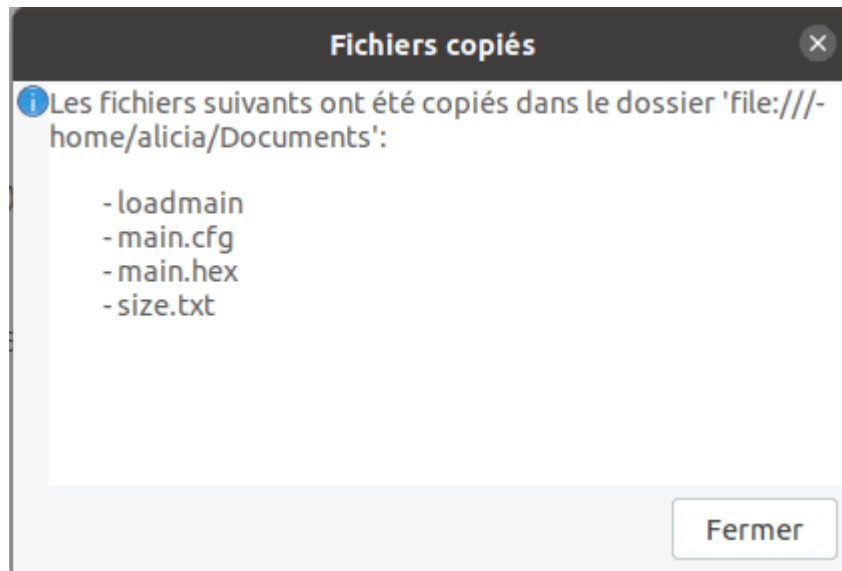
5.7.4 Compiler et charger un programme sur l'automate

Après avoir créé le programme, il faut le compiler avant de le charger dans l'automate.

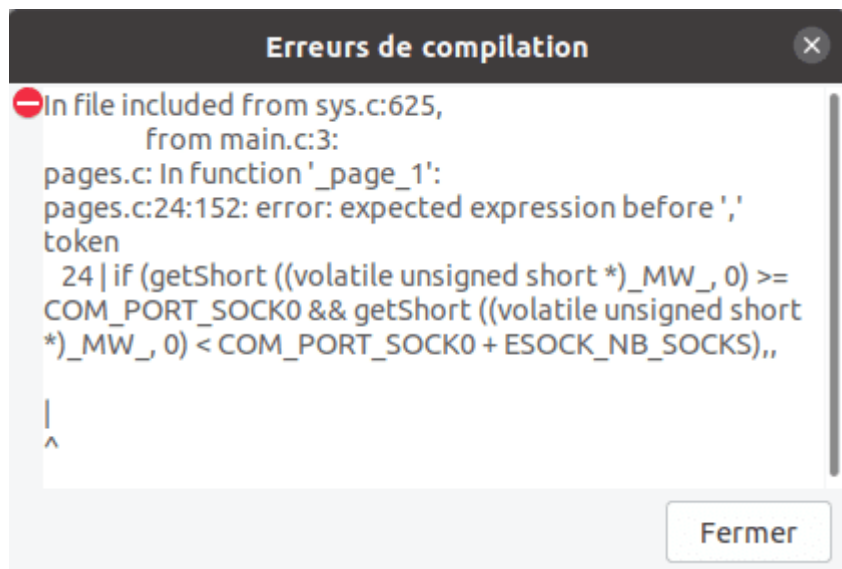
Il est conseillé de sauvegarder le projet avant de le compiler en cliquant sur l'onglet « Fichier » puis sur « Enregistrer » dans la barre de menu.

On peut ensuite compiler le programme en cliquant sur l'onglet « Programme » puis sur « Compiler » toujours dans la barre de menu. Une fenêtre *popup* apparaît, nous demandant de spécifier l'endroit où l'on souhaite sauvegarder ces nouveaux fichiers:

- Si l'automate est équipé d'un lecteur de carte SD et que l'on souhaite charger le programme par ce biais alors on copie directement les fichiers sur la carte SD. Le programme sera automatiquement chargé lors de l'insertion de la carte SD dans l'automate.
- Si on ne souhaite pas charger le programme avec la carte SD ou si l'automate ne permet pas cette option, on sélectionne simplement un dossier sur l'ordinateur. On a la possibilité de charger le fichier hexadécimal (.hex) sur l'ordinateur via MicroLADDER ou le logiciel MicroCONTROL.

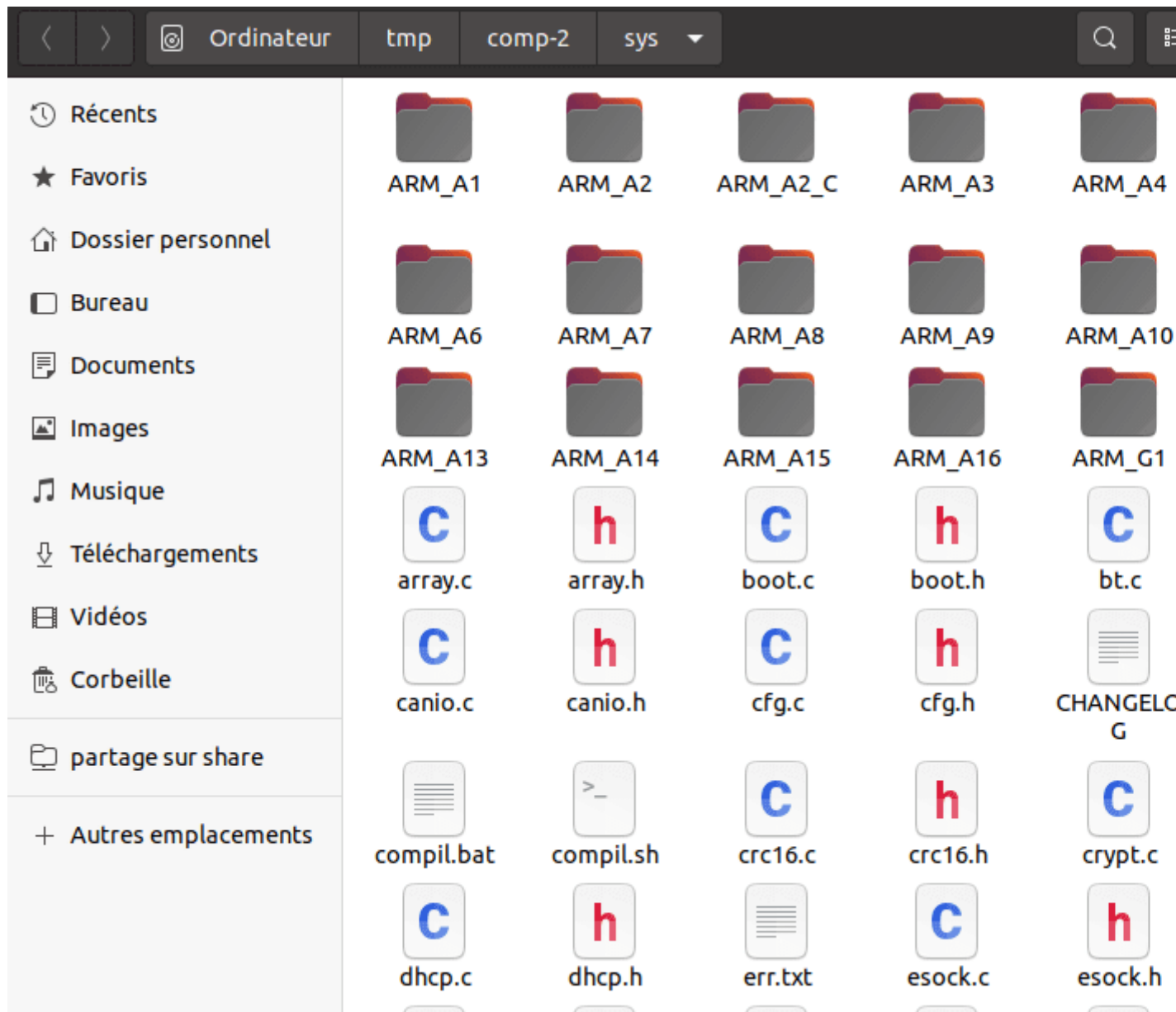


Si la compilation est réussie, les fichiers main.hex, main.cfg, loadmain et size.txt auront été créés et une seule nouvelle fenêtre « Fichiers copiés » apparaîtra pour confirmer que la compilation a réussi.



S'il y a des erreurs dans le programme, une fenêtre *popup* « Erreurs de compilation » apparaîtra ainsi qu'une deuxième fenêtre *popup* contenant tous les fichiers et dossiers nécessaires à la compilation. Ce

sont les fichiers et dossiers du code système. Tant qu'il y aura des erreurs dans le programme, ce dernier ne compilera et ces deux fenêtres s'ouvriront.



Capture d'écran de la fenêtre *popup* des fichiers et dossiers du code système lors d'une erreur de compilation

5.8 Créer un bloc fonction

Sous MicroLADDER, une fonction peut être créée de deux manières différentes:

- soit dans le même programme que l'application qui l'appelle. Dans ce cas, la fonction sera juste appelée comme dans les autres langages
- soit dans un programme « Bloc fonction ». Dans ce cas là, il va falloir l'importer avant qu'une application puisse l'utiliser. Cela peut être utile pour gérer des sous-parties identiques dans un

programme ou dans plusieurs programmes. Pour déclarer une fonction dans un bloc fonction, cliquez sur l'onglet « Programme » puis sélectionnez « Type de programme » puis « Bloc fonction ».

Une fonction peut également appeler d'autres fonctions.

Dans cette partie, on va se concentrer sur la création de fonction dans un bloc fonction.

5.8.1 Définition des variables

On a explicité précédemment que l'on peut déclarer automatiquement des variables. C'est toujours possible de faire cela dans un bloc fonction mais il est nécessaire de définir manuellement les variables d'entrée et de sortie de la fonction créée.

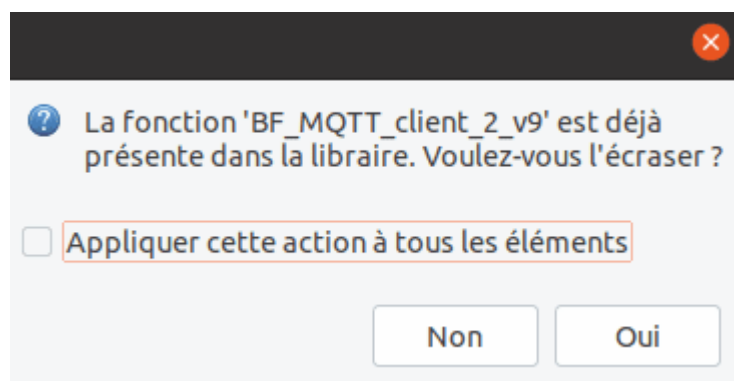
Si on choisit de coder une fonction dans un "Bloc fonction" alors quatre types de variables différentes pourront y être déclarées. Pour choisir le type d'une variable, allez dans l'éditeur de variable puis allez dans les « Propriétés de la variable » puis sur l'onglet « Programmation ». Le type de la variable est à sélectionner dans le menu déroulant « Paramètre »:

- **interne:** la variable n'est valable qu'à l'intérieur de la fonction. Elle est initialisée à chaque appel d'une fonction, sauf dans le cas où la propriété « Globale » est sélectionnée.
- **entrée:** la valeur donnée par l'application appelante est envoyée à l'appel de la fonction. La valeur sera visible dans le bloc de la fonction sur la page Ladder.
- **sortie:** la valeur est renvoyée à l'application appelante à la fin de l'exécution de la fonction. La valeur sera visible dans le bloc de la fonction sur la page Ladder.
- **externe:** à l'appel de la fonction, la variable prend la valeur que lui donne l'application appelante. A la fin de l'exécution de la fonction, la valeur est renvoyée à l'application appelante. Elle n'apparaît pas dans l'objet Ladder de la fonction sur une page Ladder.

5.8.2 Utilisation d'un bloc fonction

Le bloc fonction doit être importé à l'intérieur de l'application en cliquant sur l'onglet « Librairie » puis sur « Importer une fonction ». Une fenêtre s'ouvre permettant de sélectionner le fichier du bloc fonction voulu.

Une seule importation est suffisante, même si la fonction est utilisée plusieurs fois.



Cependant à chaque changement fait dans le bloc fonction, on est obligé d'importer à nouveau le bloc fonction. Après avoir sélectionné le fichier du bloc fonction, une fenêtre *popup* s'ouvrira. Cliquez sur « Oui » pour écraser le bloc fonction existant sinon cliquez sur « Non ».

5.8.2.1 Importer une fonction

Un objet « Fonction » doit être importé dans une page du programme en cliquant sur l'onglet « Librairie » puis sur « Importer une fonction ».

Pour les données de type binaire, les contacts et les bobines peuvent être connectés aux connexions de la fonction. Pour les données de type numérique, il faut double-cliquer sur la fonction et définir les variables échangées. Cette dernière méthode peut également être utilisée pour les données de type binaire.

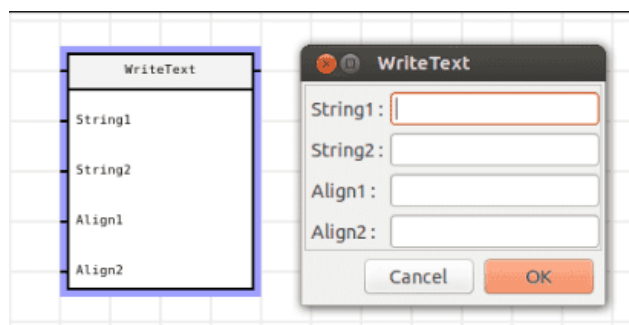
Information

Une fonction ne peut pas accéder aux bits des mots système (%S et %SW). Ils doivent être déclarés en externe. Chaque fois qu'une fonction est utilisée, elle organise son propre espace mémoire.

5.8.2.2 Utiliser une fonction en Ladder

En Ladder, pour utiliser une fonction que l'on a importée, il faut insérer un objet "Fonction", en faisant un clic-droit sur la page puis en sélectionnant "Fonction". Un menu déroulant, nous demande alors quelle fonction on souhaite importer. Cliquez sur celle que vous voulez utiliser.

Il faut maintenant passer à la déclaration des données en entrées et en sorties (si il y en a). Les données d'entrées de type binaire, les contacts et les bobines peuvent être reliés aux de la fonction. Pour les données de type numérique, il faut double-cliquer sur la fonction et définir les variables échangées. Cette dernière méthode peut également être utilisée pour les données de type binaire.



Capture d'écran d'une initialisation de variables d'une fonction

5.8.2.3 Utiliser une fonction en C

En C, la fonction doit être appelée par son nom et les variables doivent être introduites comme paramètres dans l'ordre où elles ont été déclarées : d'abord les variables d'entrée, puis les variables de sortie et enfin les variables externes. Cette démarche est similaire à l'appel d'une fonction en langage C classique, avec les variables de sortie et externes en plus.

```
string1="Good morning";
string2="Good night";
align1=1;
align2=1;
WriteText (string1, string2, align1, align2);
```

5.9 Créer des fonctions et des variables globales

Si sur une page, on souhaite utiliser des fonctions ou variables déclarées et définies sur une autre page C alors il y a une manipulation à réaliser avant de pouvoir les utiliser.
En effet, sur la page C qui contient les fonctions et variables globales, on doit englober ces dernières dans des balises global de cette manière:

```
<global>
```

```
(déclaration de variables et/ou déclaration et/ou définition de fonction)
```

```
</global>
```

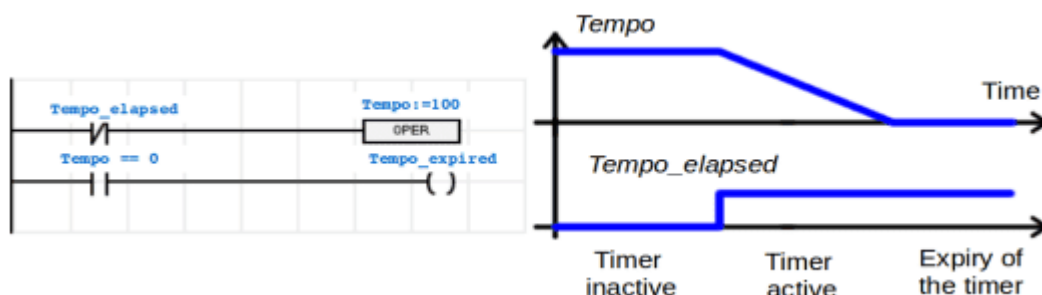
5.10 Utiliser les timers

5.10.1 C'est quoi une temporisation ?

Une temporisation a un statut parmi :

- **temporisation terminée:** lorsque la valeur du temporisation est égale à zéro.
- **temporisation active:** lorsque la valeur de la temporisation est supérieure à zéro et inférieure à la valeur de sa période de compte à rebours. Dans ce cas, sa valeur diminue avec la période de compte à rebours indiquée dans la variable.

5.10.2 Mieux comprendre les temporisations



Au premier état, il n'y a pas de temps écoulé (la variable binaire `Tempo_expired` est à 0), la valeur de la période de décompte (100) est chargée dans la variable `Tempo`. L'état de la temporisation est inactif. Tant que `Tempo_expired` est à 0, la variable `Tempo` reste à 100, donc aucun compte à rebours n'est effectué.

Au deuxième état, la variable binaire `Tempo_elapsed` est mise à 1. La valeur de la période de compte à rebours (100) n'est plus écrite dans la variable `Tempo` et la valeur de `Tempo` diminue. L'état de la temporisation est active.

Au troisième état, la valeur de `Tempo` tombe à 0 après le compte à rebours. La temporisation a expiré, elle est à nouveau inactive.

5.10.3 Comment utiliser une temporisation ?

Pour utiliser une temporisation, la propriété « `Timer(en ms)` » d'une variable doit être définie. Allez dans l'éditeur de variable puis double-cliquez sur la variable sur laquelle vous souhaitez mettre une minuterie. Une fenêtre *popup* s'ouvre et allez dans l'onglet « `Programmation` » puis remplissez les champs « `Valeur initiale` » par la valeur de la temporisation. Dans le champs « `Timer (en ms)` », écrivez le pas de décrémentation. Si on veut que notre minuterie soit décrémentée toutes les 1 s, écrivez alors 1000 dans ce champs.

La période de compte à rebours (en ms) du champs « `Timer (en ms)` » et la valeur de minuterie sera gérée par le système comme une temporisation.

L'utilisation d'une temporisation en Ladder ou en C est similaire pour les deux cas.

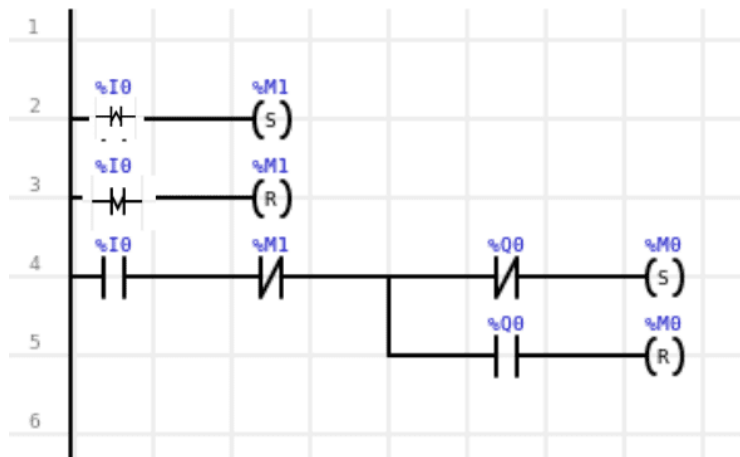
5.10.3.1 Exemple d'utilisation d'une temporisation

Pour mieux comprendre comment utiliser une minuterie on utilise l'exemple suivant:

En considérant à nouveau l'automate MicroARM-A1, on va se servir du programme décrit section [5.7 Créer un programme](#) pour lui rajouter de nouvelles fonctionnalités:

- En appuyant sur le bouton (%I0), un compteur est activé. Ce compteur ne compte que le temps pendant lequel il est poussé.
- En relâchant le bouton, le compteur se remet à zéro.
- Après trois secondes d'appui sur le bouton, le rétro-éclairage de l'écran LCD s'allume s'il était éteint, ou s'éteint s'il était allumé.

Ce programme peut être créé en Ladder comme le montre la capture d'écran suivante:



Capture d'écran d'un programme Ladder sur la fenêtre principale de MicroLADDER

Une nouvelle variable booléenne (%M1) indiquant si le bouton est appuyé(1) ou non(0) a été rajoutée. Pour établir la temporisation, écrivez trois dans le champs "Valeur initiale" et 1000 dans « Timer (en ms) » pour que le compteur soit décrémenté toutes les secondes.

6. Environnement logiciel

6.1 Architecture du système

6.1.1 Moniteur

Le moniteur est la couche logicielle de base. Il permet ensuite de charger l'application logicielle. L'utilisateur n'aura pas besoin de le modifier, sauf si un nouveau téléchargement de logiciel peut nécessiter le rechargement du logiciel moniteur.

Si une carte SD est disponible à la mise sous tension, les paramètres FORCE_MON et RUN du fichier « MAIN.CFG » disponible sur la carte SD détermineront le lancement de la boucle principale et de l'application.

Les paramètres sont sauvegardés en même temps sur la carte SD et sur l'EEPROM. Il est donc possible de fonctionner sans la carte SD. Le moniteur est chargé par Sirea.

6.1.1.1 Connaître l'état de fonctionnement de l'automate grâce une LED

Chaque automate possède une LED qui peut être soudée sur la carte ou accessible par un connecteur. On peut connaître le mode de fonctionnement de l'automate en fonction de comment va clignoter la LED.

6.1.1.1.1 Mode surveillance

La LED clignote lentement : 500 ms allumée/ 500 ms éteinte. L'automate est sous contrôle Modbus. Il peut être contrôlé par MicroCONTROL.

6.1.1.1.2 Transfert du programme de la carte SD vers la mémoire de l'API

La LED clignote rapidement : 100 ms allumée/ 100 ms éteinte.

Le transfert ne dure que quelques secondes. Il se produit lorsque MicroCONTROL a été transféré. On peut le transférer en appuyant sur le bouton ou en passant le paramètre de la carte SD.

6.1.1.1.3 Mode STOP

La LED clignote lentement et reste principalement éteinte : 100 ms allumée/ 900 ms éteinte. L'automate n'est plus sur le moniteur, la boucle principale commence, l'application ne s'exécute pas. Notre automate est stoppé.

6.1.1.1.4 Mode RUN

La LED clignote lentement et reste principalement allumée : 900 ms allumée/ 100 ms éteinte. L'application s'exécute. Notre automate est alors en fonctionnement.

6.1.1.1.5 Versions incompatibles entre le moniteur et l'application

La LED s'allume en rouge statique. Cela ne se produit que lorsque l'application démarre et que les structures de données enregistrées par le moniteur sont différentes des structures de données enregistrées par l'application. Cela peut créer des problèmes avec les variables enregistrées.

6.1.2 Fichier main.cfg

Ce fichier se trouve sur la carte SD. Il contient le paramétrage. Il est lu et réécrit lors du lancement de l'application ou de l'insertion de la carte SD.

Par conséquent, si le fichier est modifié, il est entièrement réécrit. Si des modifications doivent être apportées à ce fichier, il est important de procéder avec précaution.

Certaines caractéristiques de ce fichier peuvent être modifiées par des variables si nécessaire.

Variables du fichier main.cfg	Description
FORCE_MON	Si elle est à 1, elle impose de rester sur le moniteur et de ne pas lancer l'application
LOAD	Si elle vaut 1, indique qu'il faut charger le programme en mémoire. Elle est ensuite remise à 0 à la fin du chargement
FIRST_RUN	Si elle vaut 1, indique qu'il faut initialiser les variables. Il n'est pas nécessaire de la mettre à 1 lors d'un chargement de l'application car le système le fait tout seul. C'est la variable %S2
RUN	Si elle vaut 1, permet de passer de l'application en STOP à l'application en RUN

RESET_SW	Si elle vaut 1, indique qu'il y a eu un dépassement du temps de cycle (chien de garde) et bloque l'automate sur la boucle principale. C'est la variable %S21
RESET_CFG	Si elle vaut 1, permet de réinitialiser les variables du fichier « main.cfg »
LOAD_IO_CFG	Si elle vaut 1, permet de recharger le fichier « var.csv » et de réinitialiser l'historique des courbes, alarmes et événements. C'est la variable %S19
MODE_DST	Si elle vaut 1, permet de mémoriser la gestion de l'heure d'été. C'est la variable %SW11
TIME_OFF	Permet de régler le décalage par rapport à l'heure UTC en minute. Elle est utile pour les mise à l'heure automatique. C'est la variable %SW12
BOOT_VER	Indique la version du boot installé sur l'automate. C'est la variable %SW23
W_SSID	Permet de régler le SSID (nom du point d'accès) pour le Wi-Fi. La norme autorise jusqu'à 32 caractères
W_STYPE	Permet de régler le type de cryptage pour le Wi-Fi. Les valeurs possibles sont « » (chaîne de caractères vide utilise la détection automatique), OPEN (pas de cryptage), WEP, WPA, WPAAES, WPA2AES, WPA2TKIP, WPA2
W_SKEY	Permet de régler la clé de sécurité pour le Wi-Fi

Tableau des variables disponibles pour le fichier main.cfg

L'exemple suivant est le code d'un fichier main.cfg avec des valeurs par défaut. L'automate lancera la boucle principale de l'application présente dans la mémoire. S'il n'y a pas d'application, le système sera bloqué.

```
FORCE_MON=0
LOAD=0
RUN=0
```

6.1.3 Fichier Loadmain

Lorsque ce fichier est présent sur la carte SD, l'application est rechargée et le fichier est supprimé.

Cette méthode est équivalente à la mise à 1 de la variable LOAD, mais elle évite de modifier les autres variables du fichier main.cfg. Le contenu du fichier n'est pas important, seule sa présence compte. Il est donc possible de créer un fichier appelé « loadmain » sur la carte SD. L'automate détecte ce fichier et charge le fichier main.hex dans sa mémoire. La méthode la plus rapide est de laisser MicroLADDER sauvegarder les fichiers compilés directement sur la carte SD, puis de placer la carte SD dans l'automate.

6.1.4 Charger un programme

Il n'est pas obligatoire de posséder une carte SD. Si vous en avez une, elle doit être formatée en FAT32. Sinon on peut charger un programme via la réseau.

6.1.4.1 Via le réseau avec MicroLADDER

Le logiciel MicroLADDER permet de charger des fichiers hexadécimal (.hex), c'est-à-dire des fichiers déjà compilés.

Le transfert peut se faire par un port série ou par le port Ethernet. Il est donc possible de le charger à distance depuis le code de l'application. Pour cela, il faut aller sur l'onglet « Communication » puis cliquez sur « Chargement d'un programme ». Une fenêtre s'ouvrira nous permettant de sélectionner le fichier hexadécimal à transférer.

Le transfert du fichier s'effectue d'abord sur la carte SD, si une carte SD est disponible. Après le processus de transfert, le fichier sera chargé dans la mémoire de l'automate. L'application peut ensuite être lancée via MicroLADDER en allant sur l'onglet « Communication » puis en cliquant sur « Démarrage du programme ».

Dans le cas où l'on charge pour la première fois une application dans l'automate, il est nécessaire d'établir `FORCE_MON=1` dans le fichier `MAIN.CFG` de la carte SD. Dans le cas contraire, l'automate essaiera de lancer la boucle principale de l'application, même si elle n'existe pas.

6.1.4.2 Via Carte SD

Si l'automate comprend une carte SD, vous pouvez simplement copier le fichier compilé par MicroLADDER (`main.hex`) sur la carte SD et créer le fichier `main.cfg` qui contient les informations suivantes :

```
FORCE_MON=0  
LOAD=1  
RUN=1
```

La carte SD doit ensuite être insérée dans l'automate. La LED d'état de fonctionnement indique le transfert du programme de la carte SD vers la mémoire de l'automate par un clignotement rapide. Une fois le programme transféré, la LED d'état de fonctionnement indique l'exécution de l'application par un clignotement, principalement avec la LED d'état allumée.

6.1.5 Sauvegarde des variables

6.1.5.1 RAM Sauvegardée

Certains automates disposent d'une RAM sauvegardée. Il suffit de cocher l'option « Sauvegardée » dans les propriétés de la variables et le système gère de façon autonome la sauvegarde.

6.1.5.2 EEPROM ou FRAM

Si l'automate ne dispose pas de RAM sauvegardée mais d'une EEPROM ou FRAM, il est possible d'utiliser les variables sauvegardées. Lorsque %S18 passe à 1, le système lance une sauvegarde de toutes les variables qui ont la propriété « Sauvegardée » de cochée.

Avec cette version de MicroLADDER, il ne faut plus utiliser les instructions de sauvegarde manuelle. En effet, il y a trop de risque d'utiliser des zones mémoire utilisée par le système.

6.1.5.3 Mots systèmes

Les mots systèmes %SW15 à %SW22 sont à la disposition de l'utilisateur pour sauvegarder des mots dans la carte SD. Les sauvegardes ne sont effectuées que s'il y a une modification de valeur. Les sauvegardes peuvent avoir lieu à la fin de la communication Modbus avec la console de programmation (si forçage de valeur par la communication) et à la fin du tour de cycle automate (si modification de valeur par l'application).

Chaque sauvegarde prend de 180 à 200ms.

La lecture de ces mots est effectuée de façon transparente à l'initialisation.

Il y a une double sauvegarde sur la carte SD et dans la EEPROM ou FRAM.

6.2 Types de données

Il est important de noter que MicroLADDER gère les variables de l'API qui peuvent être utilisées dans les objets Ladder, c'est-à-dire celles qui sont accessibles à la surveillance en temps réel par le logiciel. Cependant, dans des pages C, les variables internes déclarées peuvent être d'un autre type que celles proposées par MicroLADDER.

6.2.1 Entrées TOR

Une entrée TOR vaut soit 1 ou 0. Le préfixe d'une entrée TOR est %I. Un exemple est %I0. Les données sont accessibles en Modbus standard en faisant un décalage (%I0 ? %M20000).

Le nombre d'entrées TOR varient en fonction de l'automate.

6.2.2 Entrées analogiques

La plage de valeur ainsi que le nombre de valeurs varient en fonction de l'automate utilisé. Le préfixe pour une entrée analogique est %IW. Un exemple d'entrées analogique est %IW100.

Sur certains automates, les entrées analogiques peuvent être configurées par des cavaliers. Il faut ensuite accorder la propriété « Configuration » de la variables analogiques à l'aide de MicroLADDER afin d'avoir la bonne plage de valeur et le bon étalonnage.

Propriété « Configuration »

Type d'entrée analogique

0

Configuration par défaut. Correspond à la configuration disponible sur l'automate ayant la valeur de la propriété de configuration la plus petite

1

0-20mA. Plage de valeur généralement 0-20000 points

2	0-10V. Plage de valeur généralement 0-10000 points
3	PT100. Valeur de température généralement en dixième de degrés Celsius

Tableau sur les types d'entrées analogiques

Les plages de valeurs peuvent être différentes suivant l'automate. Ces données sont accessibles en Modbus standard en faisant un décalage (%IWO ? %MW40000).

6.2.3 Sorties TOR

Une sortie TOR vaut soit 1 ou 0. Le préfixe d'une sortie TOR est %Q. Un exemple est %Q0. Le nombre de sorties TOR varient en fonction de l'automate.

Les données sont accessibles en Modbus standard en faisant un décalage (%Q0 ? %M21000).

6.2.4 Sorties analogiques

La plage de valeur ainsi que le nombre de valeurs varient en fonction de l'automate utilisé. Le préfixe pour une entrée analogique est %QW. Un exemple de sortie analogique est %QW0.

Les données sont accessibles en Modbus standard en faisant un décalage (%QW0 ? %M41000).

6.2.5 Sorties MLI (Modulation de Largeur d'Impulsions)

Tous les automates n'ont pas des sorties MLI (Modulation de Largeur d'Impulsions). Par exemple parmi les plus utilisés les automates MicroARM-12 et MicroARM-A2 ont des sorties MLI. Pour l'automate MicroARM-12, les deux sorties de MLI sont %Q100 et %QW101. Pour l'automate MicroARM-A2, les deux sorties sont %QW102 et %QW103. *Pour plus de renseignements, se référer à la fiche technique de chaque automate.*

La variable %SW26 définit la fréquence en Hz. Cette valeur doit être comprise entre 0 et 65535 Hz (en pratique l'électronique ne permet pas d'aller à une fréquence supérieure à 10000 Hz).

Une variable %QW par sortie MLI définit le rapport cyclique. Cette valeur doit être comprise entre 0 et 1000. Par exemple, pour obtenir un créneau de temps à 1 de 20% et 0 de 80%, il faut mettre %QW à 200. Le signal de sortie MLI varie entre 0V et Vcc (tension d'alim).

Pour créer une sortie en binaire, il faut paramétrer %SW26 à 0 et écrire 0 ou 1000 dans la variable %QW pour mettre la sortie TOR à 0 ou 1.

Par défaut, %SW26 vaut 0 et les %QW valent 0, ce qui correspond à une sortie toujours à 0V.

Lorsque %SW26 vaut 0, la fréquence est en réalité égale à la fréquence maximale dont peut aller le microcontrôleur de l'automate (cette fréquence est au moins égale à 100000 Hz et peut aller jusqu'à 250000 Hz). Ceci permet de réagir plus rapidement à un changement d'état en mode binaire car la période en cours se termine forcément avant d'appliquer le nouveau rapport cyclique.

Information

- Il est possible de paramétrer une fréquence allant jusqu'à 65535 Hz. Cependant
- La limite maximale est d'environ 10 kHz suivant la charge.
- Pour visualiser le signal à l'oscilloscope, il faut connecter une résistance

6.2.6 Bits

La valeur est soit 0 ou 1. Une variable bit aura le préfixe %M. Un exemple de bit est %M0.

Les bits sont des bits de caractères. De ce fait, 8 bits utilisent en mémoire 1 seul octet. L'accès à un bit est plus long que l'accès à un octet ou un mot

6.2.7 Entiers

La valeur est comprise entre 0 et 65535. Une variable entière aura le préfixe %MW (Main Word). Un exemple est %MW0.

6.2.8 Long

La valeur est comprise entre $-2\,147\,483\,648$ et $2\,147\,483\,647$. Une variable long aura le préfixe %MD. Un exemple est %MD0.

Les données sont accessibles en Modbus standard en faisant un décalage. Le calcul de l'adresse se fait grâce à l'opération suivante: Adresse = 20000 + (2 x Y). La valeur de Y étant le nombre après %MF dans la documentation. Par exemple, la documentation affiche une valeur lisible à l'adresse %MF21, l'adresse à lire grâce au protocole Modbus est 20042.

6.2.9 Float

La valeur est comprise entre 3.4×10^{-38} et 3.4×10^{38} . Une variable float aura la préfixe %MF. Un exemple est %MF0.

Les données sont accessibles en Modbus standard en faisant un décalage. Le calcul de l'adresse se fait grâce à l'opération suivante: Adresse = 30000 + (2 x Y). La valeur de Y étant le nombre après %MF dans la documentation. Par exemple, la documentation affiche une valeur lisible à l'adresse %MF31, l'adresse à lire grâce au protocole Modbus est 30062.

6.2.10 Chaînes de caractères

La longueur de la chaîne de caractères est indiquée dans le champs « Nombre de caractères ». Une variable de type chaînes de caractères aura un préfixe %MS (Main String). Un exemple de variable est %MS0.

Pour accéder au n ème caractères de la chaîne %MS0, il faut écrire %MS0[n-1]. Par exemple, on accède au 5ème caractères du tableau avec MS0[4].

6.2.11 Tableaux

Il nous ait souvent demander de créer des tableaux d'entiers, de flottants, de chaînes de caractères... . Dans ce cas, après avoir choisi le type de la variable (1er champs de l'onglet « Propriétés de la variable »), il faut remplir le champ « Adresse » avec la valeur construite de la manière suivante: *adresse[]*. Il faut ensuite remplir le champ « Nombre d'éléments » par la valeur de la taille du tableau. Par exemple, dans la capture d'écran ci-dessous, on a créé un tableau de 100 éléments dont le premier élément sera à l'adresse 100. Pour accéder au n ème élément, il faut écrire %MW100[n-1]. Par

exemple on accède au 5ème élément avec (%MW100)[4]

Propriétés de la variable

Général | Programmation | Entrée / Sortie | Communication | Historisation | Serveur distant | Affichage

Type : Entier non signé 16 bits (%MW)

Adresse : 100[]

Mnémonique : Buffer

Commentaire :

Nombre d'éléments : 100

Nombre de caractères :

Annuler Valider

Après avoir validé notre tableau, il faut créer toutes les variables qui composent les cases de notre tableau. Par exemple, si on a créé un tableau d'entiers de 100 éléments à l'adresse %MW100, on va devoir créer 100 variables de type entier. Dans la capture d'écran suivante, on a créé une première variable (1er case du tableau) à l'adresse %MW100, une deuxième variable (2ème case du tableau) à l'adresse %MW101...

BF_MQTT_client_2_v9 - Éditeur de variables

Fenêtre | Tables | Variables | Communication

Toutes les variables

Adresse	Mnémonique	Commentaire	Valeur	Valeur formatée	Forcée	Nombre
%MW100[]	Buffer				☒	
%MW100					☐	
%MW101					☐	
%MW102					☐	
%MW103					☐	
%MW104					☐	
%MW105					☐	
%MW106					☐	
%MW107					☐	
%MW108					☐	
%MW109					☐	
%MW110					☐	

6.2.12 Bits système

Ces données sont accessibles en Modbus standard en faisant un décalage (%S0 ? %M22000).

Pour la suite le terme « scrutation du programme » signifie que l'automate va exécuter de manière séquentielle le programme.

Bit	Mnémonique	Onglet	Description
%S0	MST	Contrôle du cycle	Bit mis à 1 au premier cycle du programme système qui suit après une mise sous tension ou un rechargement du programme. Bit rétabli à 0 automatiquement après l'exécution du premier cycle. Ce bit est légèrement différent de %S16 (DEM_C).
%S1	RUN	Contrôle du cycle	Bit à 1, est mis à l'état 0 par le programme utilisateur ou en mode écriture de données par la liaison série (commande RUN/STOP) Bit à 1 : scrutation du programme. Bit à 0 : arrêt du programme.
%S2	INIT	Gestion des variables	Bit à 0, est mis à l'état 1 par le système au démarrage du programme ou en mode écriture de données par la liaison série (commande INIT). Bit à 1 : chargement des valeurs initiales dans les valeurs de données. Pas de modification des variables sauvegardées. Est remis à 0 automatiquement à la fin de l'initialisation.
%S3			Réservé
%S4			Réservé
%S5	MSEC_10	Horodateur	Bits dont le changement d'état est cadencé par l'horloge interne. Ils sont asynchrones par rapport au cycle de scrutation du programme.
%S6	MSEC_100		
%S7	SEC_1		
%S8	MIN_1		
%S9			Réservé
%S10			Réservé
%S11	INIT_Q	Gestion des entrées/sorties	Bit à 0, est mis à l'état 1 dans le programme utilisateur ou en la forçant depuis l'éditeur de variables. Bit à 1 : provoque la mise à 0 des sorties. Bit à 0 : les sorties sont mises à jour par le programme utilisateur.

%S12	GEL_Q	Gestion des entrées/sorties	Bit à 0, est mis à l'état 1 dans le programme utilisateur ou en la forçant depuis l'éditeur de variables. Bit à 1 : provoque le figeage des sorties dans l'état. Bit à 0 : les sorties sont mises à jour par le programme utilisateur.
%S13	MAJ_H		Bit à 0, est mis à l'état 1 dans le programme utilisateur ou en la forçant depuis l'éditeur de variables. Bit à 1 : utilisé pour faire une mise à l'heure et à la date : copie les valeurs des %SW5 à %SW10 correspondant dans l'horloge système. Bit à 0 : les mots %SW5 à %SW10 sont rafraîchis par le système afin d'indiquer l'heure et la date en cours. Depuis le 29/06/12, ce bit ne sert plus et n'apparaît plus dans l'éditeur de variables. Il est possible de modifier %SW5 à %SW10 n'importe quand. La modification est prise à la fin du tour de cycle automate.
%S15	CDG	Chiens de garde	Bit à 0, est mis à l'état 1 par le système dès que le temps de cycle du programme devient supérieur au temps de cycle maximum défini dans le mot système SW1. Est remis automatiquement à 0 par une commande INIT ou à la mise sous tension (mode MST) ou par le programme utilisateur
%S16	DEM_C	Contrôle du cycle	Bit à 1 au premier tour de scrutation programme utilisateur quelle que soit la cause du redémarrage (mise sous tension, rechargement, démarrage par la console). Est remis à 0 automatiquement à la fin du 1° cycle. Ce bit est légèrement différent de %S0 (MST).
%S17			Réservé Bit à 0, est mis à l'état 1 dans le programme utilisateur ou en la forçant depuis l'éditeur de variables.
%S18	SAV_VARS	Gestion des variables	Provoque une sauvegarde dans la EEPROM ou la FRAM, des variables déclarées sauvegardées. Ce bit est remis à 0 à la fin de la sauvegarde. Voir la section 6.1.5 Sauvegarde des variables pour plus de détails. Attention il ne faut pas faire de sauvegarde en permanence car la EEPROM a un nombre d'écriture limité.

%S19	LOAD_IO_CFG		Bit à 0, est mis à l'état 1 dans le programme utilisateur ou en la forçant depuis l'éditeur de variables. Provoque la relecture du fichier « var.csv » et la réinitialisation de l'historique des alarmes et évènements.
%S20	RESET_HARD	Chiens de garde	Non géré
%S21	RESET_SOFT	Chiens de garde	Bit à 0, est mis à 1 par le système après un reset soft par le watchdog. Ce bit est remis à 0 lors du redémarrage du programme. Voir la section 6.3 Watchdog pour plus de détails.
%S22	RESET_ERR	Chiens de garde	Non géré
%S23			Réservé
%S24	ALM_BAT_RTC	Horodateur	Normalement à 0, est mis à 1 par le système lorsque la batterie de sauvegarde est faible
%S25	DHCP	Ethernet	Utiliser DHCP pour paramétrer le port Ethernet. Le socket 3 est utilisée pour la demande initiale et pour des demandes périodiques. Cela signifie que le DHCP prend temporairement la main sur le socket au détriment de l'utilisation prévue par l'application.
%S26	W_DHCP	Wifi	Utiliser DHCP pour paramétrer le port Wi-Fi. Contrairement à l'Ethernet, le module Wi-Fi utilise un socket interne qui ne perturbe pas l'application.
%S27			
à			Réservé
%S49			

Tableau sur la définition des bits systèmes

6.2.13 Mots systèmes

Ces données sont accessibles en MODBUS standard en faisant un décalage (%SW0 ? %MW42000).

6.2.13.1 Détail Modbus

Bit	Mnémonique	Onglet	Description
%SW0	T_CYC	Chiens de garde	Temps de cycle automate en ms. Accessible seulement en lecture.

%SW1	S_CYC	Chiens de garde	Surveillance du temps de cycle automate. Valeur de référence permettant au système de contrôler un dépassement de temps de scrutation (SW0 > SW1) et de positionner le bit %S15 (CDG). Il n'y a pas d'arrêt de l'application. Accessible en lecture et écriture. Valeur de timeout du temps de cycle automate. Valeur de référence permettant au système de contrôler un dépassement maximal du temps de scrutation. En cas de dépassement (SW0 > SW2), le système effectue une sauvegarde des variables, force les sorties TOR et analogiques à 0 et passe l'automate en mode STOP. Le bit %S21 (RESET_SOFT) est positionné à 1. Accessible en lecture et écriture.
%SW2	Mx_CYC	Chiens de garde	Réservé
%SW3			
%SW4	WD_HARD	Chiens de garde	Non géré
%SW5	SECONDE	Horodateur	Fonction date et heure.
%SW6	MINUTE		Mots contenant les valeurs courantes de la date et l'heure.
%SW7	HEURE		Depuis le 29/06/12, ces mots sont accessibles en lecture et écriture en permanence, sans tenir compte de l'état de %S13 (MAJ_H).
%SW8	JOUR		
%SW9	MOIS		
%SW10	ANNEE		

%SW11	MODE_DST	Horodateur	Gestion automatique de l'heure d'été (valable depuis le 16/07/12).
			• Valeur 0 : le système gère automatiquement le passage à l'heure d'été d'après les conventions européennes. • Valeur 1 : le système gère automatiquement le passage à l'heure d'été d'après les conventions américaines et canadiennes (valable depuis le 02/02/21 et le code système 14.10). • Valeur 2 à 65534 : réservé pour de futurs modes. • Valeur 65535 : pas de gestion automatique de l'heure d'été.
%SW12	TIME_OFF	Horodateur	Le composant matériel (RTC) reste toujours à l'heure d'hiver. C'est la valeur utilisable dans les mots système (%SW5 à %SW10) qui tient compte du passage à l'heure d'été Décalage par rapport à l'heure UTC (en minutes). Pour la France, il faut mettre 60.
%SW13	VERSION_SYS	Informations système	Version du code système. Seulement en lecture.
%SW14	VERSION_APP	Informations système	Version de l'application. Ce mot est à la disposition de l'utilisateur pour stocker le numéro de version de son application.
%SW15	SAV1	Gestion des variables	Ces 8 mots sont à la disposition de l'utilisateur pour sauvegarder des mots. Voir la section 6.1.5 Sauvegarde des variables pour plus de détails.
%SW16	SAV2		
%SW17	SAV3		
%SW18	SAV4		
%SW19	SAV5		
%SW20	SAV6		
%SW21	SAV7		
%SW22	SAV8		
%SW23	VERSION_BOOT	Informations système	Version du boot. Valable depuis le code système V8.0 du 26/07/16.
%SW24	VERSION_MLADDER	Informations système	Version de MicroLADDER qui a servi à la compilation. Valable depuis MicroLADDER V11.0 du 16/08/16.

			Période d'appel des fonctions interruptives en μs. Les fonctions qui doivent être appelés doivent avoir la propriété « Appeler sur interruption » de validée. Valeur 0 : pas d'appel.
%SW25	FREQ_TIMER	Interruption	Valeur différente de 0 : période d'appel en μs. Cette valeur n'est pas sauvegardée. Elle doit être initialisé par le programme. Au cours du programme, il est possible de la modifier à volonté. Une valeur trop faible bloque l'automate.
%SW26	FREQ_PWM	Gestion des entrées/sorties	Fréquence de la sortie MLI. Masque de lecture. Il permet d'empêcher le rafraîchissement des entrées. Chaque entrée est associée à un bit. 1^{er} et 2nd octets d'entrée 3^{ème} et 4^{ème} octet d'entrée 5^{ème} et 6^{ème} octet d'entrée 7^{ème} et 8^{ème} octet d'entrée 9^{ème} et 10^{ème} octet d'entrée 11^{ème} et 12^{ème} octet d'entrée Remarque : une carte d'entrée de 16 bits peut être à cheval sur 2 mots système. Ces variables n'existent plus depuis MicroLADDER V16.0 du 14/03/19 et ne sont plus visibles dans l'éditeur de variables. Il est maintenant possible de forcer directement des variables.
%SW27	FET1		
%SW28	FET2		
%SW29	FET3		
%SW30	FET4		
%SW31	FET5		
%SW32	FET6		
%SW33			Réservé

		Configuration du port
		Numéro d'esclave
		Résultat de l'échange
		Code de vitesse
		Format de communication
		Pour le Modbus maître, valeur de temps maximale en ms entre l'envoi d'une trame et la réception de la réponse. La valeur par défaut est 5000 ms. Voir la section 6.2.13.1 Détail Modbus pour plus d'informations
		Configuration du port
		Numéro d'esclave
		Résultat de l'échange
		Code de vitesse
		Format de communication
		Pour le Modbus maître, valeur de temps maximale en ms entre l'envoi d'une trame et la réception de la réponse. La valeur par défaut est 3000 ms. Voir la section 6.2.13.1 Détail Modbus pour plus d'informations
		Configuration du port
		Numéro d'esclave
		Résultat de l'échange
		Code de vitesse
		Format de communication
		Pour le Modbus maître, valeur de temps maximale en ms entre l'envoi d'une trame et la réception de la réponse. La valeur par défaut est 3000 ms. Voir la section 6.2.13.1 Détail Modbus pour plus d'informations
		Configuration du port
		Numéro d'esclave
		Résultat de l'échange
		Code de vitesse
		Format de communication
		Pour le Modbus maître, valeur de temps maximale en ms entre l'envoi d'une trame et la réception de la réponse. La valeur par défaut est 3000 ms. Voir la section 6.2.13.1 Détail Modbus pour plus d'informations
		Configuration du port
		Numéro d'esclave
		Résultat de l'échange
		Code de vitesse
		Format de communication
		Pour le Modbus maître, valeur de temps maximale en ms entre l'envoi d'une trame et la réception de la réponse. La valeur par défaut est 3000 ms. Voir la section 6.2.13.1 Détail Modbus pour plus d'informations

		Configuration du port	
		Numéro d'esclave	
		Résultat de l'échange	
		Code de vitesse	
		Format de communication	
%SW58	SER4	Ports séries	Pour le Modbus maître, valeur de temps maximale en ms entre l'envoi d'une trame et la réception de la réponse. La valeur par défaut est 3000 ms. Voir la section 6.2.13.1 Détail Modbus pour plus d'informations
%SW59	NESC_SER4		
%SW60	RES_SER4		
%SW61	SPD_SER4		
%SW62	FOR_SER4		
%SW63	TIMEOUT_SER4		
		Configuration du port	
		Numéro d'esclave	
		Résultat de l'échange	
		Code de vitesse	
		Format de communication	
%SW64	SER5	Ports séries	Pour le Modbus maître, valeur de temps maximale en ms entre l'envoi d'une trame et la réception de la réponse. La valeur par défaut est 3000 ms. Voir la section 6.2.13.1 Détail Modbus pour plus d'informations
%SW65	NESC_SER5		
%SW66	RES_SER5		
%SW67	SPD_SER5		
%SW68	FOR_SER5		
%SW69	TIMEOUT_SER5		
		Configuration du port	
		Numéro d'esclave	
		Résultat de l'échange	
		Code de vitesse	
		Format de communication	
%SW70	SER6	Ports séries	Pour le Modbus maître, valeur de temps maximale en ms entre l'envoi d'une trame et la réception de la réponse. La valeur par défaut est 3000 ms. Voir 6.2.13.1 Détail Modbus pour plus d'informations
%SW71	NESC_SER6		
%SW72	RES_SER6		
%SW73	SPD_SER6		
%SW74	FOR_SER6		
%SW75	TIMEOUT_SER6		
		Protocole de communication du socket (voir 6.2.13.1 Détail Modbus pour plus d'informations).	
		Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)	
%SW76	SER7	Ports séries	Numéro d'esclave du socket
%SW77	NESC_SER7		Résultat de l'échange du socket
%SW78	RES_SER7		Timeout du socket
%SW79	SPD_SER7		
%SW80	FOR_SER7		
%SW81	TIMEOUT_SER7		

%SW91 W_DNS_ADDR1	Wifi	Adresse du serveur DNS pour le port Wi-Fi (valable depuis MicroLADDER 13.0 et le code système 10.0 du 11/09/17).
%SW92 W_DNS_ADDR2		
%SW93 W_DNS_ADDR3		
%SW94 W_DNS_ADDR4		
%SW95 W_MODE	Wifi	Mode de fonctionnement du port Wi-Fi.
%SW96 DNS_ADDR1	Ethernet	Adresse du serveur DNS pour le port Ethernet et pour le port Wi-Fi. A partir de MicroLADDER 13.0 et du code système 10.0 du 11/09/17, ces mots ne sont valables que pour l'Ethernet puisqu'il y a maintenant des mots pour le Wi-Fi.
%SW97 DNS_ADDR2		
%SW98 DNS_ADDR3		
%SW99 DNS_ADDR4		
%SW100 MAC_ADDR1	Ethernet	Adresse MAC du port Ethernet.
%SW101 MAC_ADDR2		
%SW102 MAC_ADDR3		
%SW103 MAC_ADDR4		
%SW104 MAC_ADDR5		
%SW105 MAC_ADDR6		
%SW106 IP_ADDR1	Ethernet	Adresse IP du port Ethernet.
%SW107 IP_ADDR2		
%SW108 IP_ADDR3		
%SW109 IP_ADDR4		
%SW110 SUBNET_MASK1	Ethernet	Masque de sous réseau du port Ethernet.
%SW111 SUBNET_MASK2		
%SW112 SUBNET_MASK3		
%SW113 SUBNET_MASK4		
%SW114 GATEWAY1	Ethernet	Passerelle du port Ethernet
%SW115 GATEWAY2		
%SW116 GATEWAY3		
%SW117 GATEWAY4		
%SW118 SOCK0	Ethernet	Protocole de communication du socket (voir 6.2.13.1 Détail Modbus pour plus d'informations).
%SW119 PORT_SOCK0		Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)
%SW120 NESC_SOCK0		Numéro d'esclave du socket
%SW121 RES_SOCK0		Résultat de l'échange du socket
%SW122 TIMEOUT_SOCK0		Timeout du socket

%SW123 SOCK1		Protocole de communication du socket (voir 6.2.13.1 Détail Modbus pour plus d'informations).
%SW124 PORT SOCK1	Ethernet	Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)
%SW125 NESC SOCK1		Numéro d'esclave du socket
%SW126 RES SOCK1		Résultat de l'échange du socket
%SW127 TIMEOUT SOCK1		Timeout du socket
%SW128 SOCK2		Protocole de communication du socket (voir 6.2.13.1 Détail Modbus pour plus d'informations).
%SW129 PORT SOCK2	Ethernet	Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)
%SW130 NESC SOCK2		Numéro d'esclave du socket
%SW131 RES SOCK2		Résultat de l'échange du socket
%SW132 TIMEOUT SOCK2		Timeout du socket
%SW133 SOCK3		Protocole de communication du socket (voir 6.2.13.1 Détail Modbus pour plus d'informations).
%SW134 PORT SOCK3	Ethernet	Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)
%SW135 NESC SOCK3		Numéro d'esclave du socket
%SW136 RES SOCK3		Résultat de l'échange du socket
%SW137 TIMEOUT SOCK3		Timeout du socket
%SW138 W_MAC_ADDR1		
%SW139 W_MAC_ADDR2		
%SW140 W_MAC_ADDR3		
%SW141 W_MAC_ADDR4	Wifi	Adresse MAC du port Wi-Fi.
%SW142 W_MAC_ADDR5		
%SW143 W_MAC_ADDR6		
%SW144 W_IP_ADDR1		
%SW145 W_IP_ADDR2		
%SW146 W_IP_ADDR3	Wifi	Adresse IP du port Wi-Fi.
%SW147 W_IP_ADDR4		

%SW148 W_SUBNET_MASK1	Wifi	Masque de sous réseau du port Wi-Fi.
%SW149 W_SUBNET_MASK2		
%SW150 W_SUBNET_MASK3		
%SW151 W_SUBNET_MASK4		
%SW152 W_GATEWAY1	Wifi	Passerelle du port Wi-Fi.
%SW153 W_GATEWAY2		
%SW154 W_GATEWAY3		
%SW155 W_GATEWAY4		
%SW156 WSOCK0	Wifi	Protocole de communication du socket (voir 6.2.13.1 Détail Modbus pour plus d'informations).
%SW157 PORT_WSOCK0		Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)
%SW158 NESC_WSOCK0		Numéro d'esclave du socket
%SW159 RES_WSOCK0		Résultat de l'échange du socket
%SW160 TIMEOUT_WSOCK0	Wifi	Timeout du socket
%SW161 WSOCK1		Protocole de communication du socket (voir 6.2.13.1 Détail Modbus pour plus d'informations).
%SW162 PORT_WSOCK1		Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)
%SW163 NESC_WSOCK1		Numéro d'esclave du socket
%SW164 RES_WSOCK1	Wifi	Résultat de l'échange du socket
%SW165 TIMEOUT_WSOCK1		Timeout du socket
%SW166 WSOCK2		Protocole de communication du socket (voir 6.2.13.1 Détail Modbus pour plus d'informations).
%SW167 PORT_WSOCK2		Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)
%SW168 NESC_WSOCK2	Wifi	Numéro d'esclave du socket
%SW169 RES_WSOCK2		Résultat de l'échange du socket
%SW170 TIMEOUT_WSOCK2		Timeout du socket

%SW171 WSOCK3		Configuration du port LoRa (voir 6.2.13.1 Détail Modbus pour plus d'informations). Numéro d'esclave (cette variable est sauvegardée dans la carte SD) Résultat de l'échange (voir 6.2.13.1 Détail Modbus pour plus d'informations)
%SW172 PORT_WSOCK3	Wifi	Canal de fréquence de réception (voir 6.2.13.2 Détail Radio Fréquence) Canal fréquence d'envoi (voir 6.2.13.2 Détail Radio Fréquence)
%SW173 NESC_WSOCK3		Pour le Modbus maître, valeur de temps maximale en ms entre
%SW174 RES_WSOCK3		l'envoi d'une trame et la réception de la réponse. La valeur par défaut est 3000ms.
%SW175 TIMEOUT_WSOCK3		Valeur du RSSI Wi-Fi en Décibel milliwatt (ou dBm). Correspond à -WsockLastRSSI qui est en dBm.
%SW189 W_RSSI		Valeur du RSSI du premier port radio en -dBm. Correspond à -RFLastRSSI[0] qui est en dBm.
%SW190 RSSI_RF0	Wifi / Radio	Valeur du RSSI du second port radio en -dBm. Correspond à -RFLastRSSI[1] qui est en dBm.
%SW191 RSSI_RF1		Ces variables sont mise à jour toutes les minutes. Elles sont valables depuis MicroLADDER V14.15 du 05/06/18 et le code système V11.9 du 05/06/18
%SW200 RF0		Configuration du port LoRa (voir 6.2.13.1 Détail Modbus pour plus d'informations). Numéro d'esclave (cette variable est sauvegardée dans la carte SD) Résultat de l'échange (voir 6.2.13.1 Détail Modbus pour plus d'informations)
%SW201 NESC_RF0		Canal de fréquence de réception (voir 6.2.13.2 Détail Radio Fréquence)
%SW202 RES_RF0	Radio	Canal fréquence d'envoi (voir 6.2.13.2 Détail Radio Fréquence)
%SW203 FREQ_RX_RF0		Pour le Modbus maître, valeur de temps maximale en ms entre
%SW204 FREQ_TX_RF0		l'envoi d'une trame et la réception de la réponse. La valeur par défaut est 3000ms.
%SW205 TIMEOUT_RF0		

%SW206 RF1

%SW207 NESC_RF1

%SW208 RES_RF1

Radio

%SW209 FREQ_RX_RF1

%SW210 FREQ_TX_RF1

%SW211 TIMEOUT_RF1

%SW214 BT_MAC_ADDR1

%SW215 BT_MAC_ADDR2

%SW216 BT_MAC_ADDR3

%SW217 BT_MAC_ADDR4

%SW218 BT_MAC_ADDR5

%SW219 BT_MAC_ADDR6

%SW220 BT0

%SW221 NESC_BT0

%SW222 RES_BT0

%SW223 GAP_BT0

%SW224 TIMEOUT_BT0

%SW225 MAC_ADDR1_2

%SW226 MAC_ADDR2_2

%SW227 MAC_ADDR3_2

%SW228 MAC_ADDR4_2

%SW229 MAC_ADDR5_2

%SW230 MAC_ADDR6_2

%SW231 IP_ADDR1_2

%SW232 IP_ADDR2_2

%SW233 IP_ADDR3_2

%SW234 IP_ADDR4_2

Protocole de communication du socket (voir [6.2.13.1 Détail Modbus](#) pour plus d'informations) du port Ethernet 2.

Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)

Numéro d'esclave du socket

Résultat de l'échange du socket

Timeout du socket

Adresse MAC du port Bluetooth.

Ces variables sont valables depuis MicroLADDER V15.0 du 20/11/18.

Mode de fonctionnement du port Bluetooth
0

Numéro d'esclave du port Bluetooth

Etat du port Bluetooth 0

Mode GAP du port Bluetooth 0 (0 = déconnecté, bluetooth inactif ; 1

= advertising (connexion possible) ; 2 = connecté)

Timeout du port Bluetooth 0

Ces variables sont valables depuis MicroLADDER V15.0 du 20/11/18.

Adresse MAC du port Ethernet 2. Valable depuis MicroLADDER V18.0 du 25/11/21.

Adresse IP du port Ethernet 2. Valable depuis MicroLADDER V18.0 du 25/11/21.

%SW235 SUBNET_MASK1_2	
%SW236 SUBNET_MASK2_2	
%SW237 SUBNET_MASK3_2	
%SW238 SUBNET_MASK4_2	Masque du sous réseau du port Ethernet 2. Valable depuis MicroLADDER V18.0 du 25/11/21
%SW239 GATEWAY1_2	
%SW240 GATEWAY2_2	
%SW241 GATEWAY3_2	
%SW242 GATEWAY4_2	Passerelle du port Ethernet 2. Valable depuis MicroLADDER V18.0 du 25/11/21.
%SW243 SOCK4	
	Protocole de communication du socket (voir 6.2.13.1 Détail Modbus pour plus d'informations) du port Ethernet 2.
%SW244 PORT SOCK4	Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)
%SW245 NESC SOCK4	Numéro d'esclave du socket
%SW246 RES SOCK4	Résultat de l'échange du socket
%SW247 TIMEOUT SOCK4	Timeout du socket
%SW248 SOCK5	
	Protocole de communication du socket (voir 6.2.13.1 Détail Modbus pour plus d'informations) du port Ethernet 2.
%SW249 PORT SOCK5	Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)
%SW250 NESC SOCK5	Numéro d'esclave du socket
%SW251 RES SOCK5	Résultat de l'échange du socket
%SW252 TIMEOUT SOCK5	Timeout du socket
%SW253 SOCK6	
	Protocole de communication du socket (voir 6.2.13.1 Détail Modbus pour plus d'informations) du port Ethernet 2.
%SW254 PORT SOCK6	Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)
%SW255 NESC SOCK6	Numéro d'esclave du socket
%SW256 RES SOCK6	Résultat de l'échange du socket
%SW257 TIMEOUT SOCK6	Timeout du socket

%SW258 SOCK7

Protocole de communication du socket (voir [6.2.13.1 Détail Modbus](#) pour plus d'informations) du port Ethernet 2.

%SW259 PORT_SOCKET

Numéro de port en mode esclave du socket (502 pour Modbus, 10000 pour VNC, 80 pour HTTP, 161 pour SNMP)

%SW260 NESC_SOCKET

Numéro d'esclave du socket

%SW261 RES_SOCKET

Résultat de l'échange du socket

%SW262 TIMEOUT_SOCKET

Timeout du socket

Tableau sur le détail des variables systèmes

SER0 à SER7

SOCK0 à SOCK3

WSOCK0 à

WSOCK3

RF0 à RF1

Variables de configuration du port. Ces variables ne sont pas sauvegardées.

Nom de la valeur	Valeur de la variable	Description
COM_PROTOCOL_NONE	0	Aucun protocole ou Modbus maître
COM_PROTOCOL_TCP		
COM_PROTOCOL_TCP_CLIENT		
COM_PROTOCOL_IOPUS	1	Protocole IOBus
COM_PROTOCOL_MODBUS	2	Modbus esclave
COM_PROTOCOL_MODBUS_TCP	3	Modbus TCP esclave
COM_PROTOCOL_GFX	4	Protocole Gfx (pilotage écran à distance)
COM_PROTOCOL_HTTP	5	Protocole HTTP
COM_PROTOCOL_UDP	6	Protocole maître UDP
COM_PROTOCOL_UDP_CLIENT		Protocole client UDP
COM_PROTOCOL_DATA	7	TBC
COM_PROTOCOL_TCP_SERVER	8	TBC
COM_PROTOCOL_UDP_SERVER	9	TBC
COM_PROTOCOL_HTTP_QUERY	10	Protocole HTTP pour les commandes (valable seulement depuis le 23/01/17)
COM_PROTOCOL_RF	11	Protocole radio (valable seulement depuis le code système 12.0)
COM_PROTOCOL_BT	12	Protocole Bluetooth (valable seulement depuis le code système 12.0)

COM_PROTOCOL_MODBUS_RO	13	Modbus esclave Read Only (valable seulement depuis le 15/01/20 code système 11.25 et code système 13.16)
COM_PROTOCOL_MODBUS_TCP_RO	14	Modbus TCP esclave Read Only (valable seulement depuis le 15/01/20 code système 11.25 et code système 13.16)

Tableau des valeurs des protocoles avec leurs descriptions

Information

La valeur par défaut est COM_PROTOCOL_MODBUS pour les ports série. Elle est CO
Il est vivement recommandé d'utiliser le nom de la valeur pour initialiser le

NESC_SER0 à NESC_SER7
NESC_SOCK0 à NESC_SOCK3
NESC_WSOCK0 à NESC_WSOCK3

Numéro d'esclave pour le mode Modbus esclave. Ces variables ne doivent pas obligatoirement être à 0 pour faire du Modbus maître ou de la communication sans protocole. Ces variables sont sauvegardées sur la carte SD

Nom de la valeur	Valeur de la variable	Description
	0	Seulement Modbus maître
	1 à 255	Numéro d'esclave Modbus

La valeur pas défaut est 1.

RES_SER0 à RES_SER7
RES_SOCK0 à RES_SOCK3
RES_WSOCK0 à RES_WSOCK3
RES_RF0 à RES_RF1

Variables du résultat de l'échange. Elles fonctionnent en émission ou réception.

Nom de la valeur	Valeur de la variable	Description
COM_STATE_READY	0	Communication terminée
COM_STATE_WAIT	1	Temporisation avant envoi de la trame
COM_STATE_SEND	2	Communication en cours (émission ou réception)
COM_STATE_ERROR	240	Limite à partir de laquelle commence la zone des défauts

COM_STATE_ERROR + MODBUS_ERROR_BAD_RESPONSE	253	Mauvaise réponse
COM_STATE_ERROR + MODBUS_ERROR_INCOMPLETE_RESPONSE	254	Réponse incomplète
COM_STATE_ERROR + MODBUS_ERROR_NO_RESPONSE	255	Time out : pas de réponse

Tableau des résultats des échanges avec leurs descriptions

SPD_SER0 à SPD_SER7 **Variables sur la vitesse d'échange. Ces variables sont sauvegardées dans la carte SD.**

Nom de la valeur	Valeur de la variable	Description
COM_SPEED_NONE	0	
COM_SPEED_300	1	300 bauds
COM_SPEED_1200	2	1200 bauds
COM_SPEED_2400	3	2400 bauds
COM_SPEED_4800	4	4800 bauds
COM_SPEED_9600	5	9600 bauds
COM_SPEED_19200	6	19200 bauds
COM_SPEED_38400	7	38400 bauds
COM_SPEED_57600	8	57600 bauds
COM_SPEED_76800	9	76800 bauds
COM_SPEED_115200	10	115200 bauds

Tableau des vitesses d'échange avec leurs descriptions

FOR_SER0 à FOR_SER7 **Variables sur les paramètres du format d'échange. Ces variables sont sauvegardées dans la carte SD.**

Nom de la valeur	Valeur de la variable	Description
COM_FORMAT_NONE	0	
COM_FORMAT_8N2	1	8 bits, pas de parité, 2 bits de stop,
COM_FORMAT_8N1	2	8 bits, pas de parité, 1 bit de stop,
COM_FORMAT_8E1	3	8 bits, parité paire, 1 bit de stop,
COM_FORMAT_8O1	4	8 bits, parité impaire, 1 bit de stop,
COM_FORMAT_7E2	5	7 bits, parité paire, 2 bits de stop,
COM_FORMAT_7O2	6	7 bits, parité impaire, 2 bits de stop,
COM_FORMAT_7E1	7	7 bits, parité paire, 1 bit de stop,
COM_FORMAT_7O1	8	7 bits, parité impaire, 1 bit de stop,

COM_FORMAT_7R2	9	7 bits, parité forcée à 0, 2 bits de stop,
COM_FORMAT_7R1	10	7 bits, parité forcée à 0, 1 bit de stop,
COM_FORMAT_7S2	11	7 bits, parité forcée à 1, 2 bits de stop,
COM_FORMAT_7S1	12	7 bits, parité forcée à 1, 1 bit de stop,

Tableau sur les paramètres de format d'échange avec leurs descriptions

La valeur par défaut est COM_FORMAT_8N1 (2).

6.2.13.2 Détail Radio Fréquence

FREQ_RX_RF0 à FREQ_RX_RF1 FREQ_TX_RF0 à FREQ_TX_RF1	Variables sur le numéro de canal de fréquence du port radio. Ces variables sont sauvegardées sur la carte SD.
--	--

Nom de la valeur	Valeur de la variable	Description
RF_FREQ_NONE	0	868,125 MHz
RF_FREQ_0		
RF_FREQ_1	1	868,475 MHz
RF_FREQ_2	2	868,950 MHz
RF_FREQ_3	3	869,525 MHz

Tableau sur les fréquences avec leurs descriptions

6.3 Watchdog

Un watchdog ou chien de garde va permettre à l'automate de ne pas se bloquer dans une étape particulière.

A chaque cycle, l'automate va lancer une temporisation. Si cette temporisation dépasse une temporisation de référence, alors le watchdog se déclenche. Le watchdog va permettre d'effectuer un reset du système ou bien de redémarrer le système.

6.3.1 Dépassement du temps de cycle

La valeur de temporisation de référence du système est à paramétrer dans la variable système %SW1 en ms.

Lorsque le temps de cycle (%SW0) est supérieur à la valeur de référence, %SW1, le système met la variable booléenne %S15 à 1. Il n'y a pas de blocage du cycle, le programme va continuer à tourner. On va seulement avoir une indication si il y a eu un dépassement du cycle en observant la variable %S15. Ainsi, si la variable %S15 est à 1, elle doit être remis volontairement à 0.

Si %SW1 vaut 0, il n'y a pas de contrôle du temps de cycle.

6.3.2 Watchdog soft par interruption

La valeur de temporisation de référence du système est à paramétrer dans la variable système %SW2 en ms.

Lorsque le temps de cycle (%SW0) est supérieur à la valeur de référence, %SW2, le système effectue une sauvegarde des variables, force les sorties TOR et analogiques à 0, passe l'automate en mode STOP, met %S21 à 1 et enfin écrit 1 dans le paramètre RESET_SW sur la carte SD.

La vérification que le temps de cycle n'est pas supérieur à la valeur de référence est levée par interruption.

Si %SW2 vaut 0, le système utilise la valeur par défaut (3000 ms).

6.4 Configuration du code système

6.4.1 Optimiser la taille du code

Cette option est valable depuis la version 14.0 de MicroLADDER et la version 11.0 du code système.

Lorsque cette option est à 1, la taille du code est réduite, mais le temps d'exécution est augmenté.

Par défaut, l'option est à 0.

6.4.2 AUTO_STOP

Lorsque cette option est à 0, lors d'un watchdog soft, l'application redémarre seule. %S21 est mis à 1 et l'application doit la remettre à 0.

Lorsque cette option est à 1, lors d'un watchdog soft, l'application ne redémarre pas et l'automate reste sur la boucle principale. %S21 est mis à 1 et il est remis à 0 si l'application est redémarrée.

Par défaut cette option est à 1.

6.4.3 DHCP

Lorsque cette option est à 1, elle permet d'avoir la possibilité d'utiliser la fonction DHCP (attribution d'adresse IP automatiquement par un serveur DHCP).

Lorsque cette option est à 0, elle permet de gagner en taille de code.

Par défaut, l'option est à 1.

6.4.4 DNS

Lorsque cette option est à 1, elle permet d'avoir la possibilité d'utiliser la fonction DNS (conversion d'une adresse web littérale en adresse IP par un serveur DNS).

Lorsque cette option est à 0, elle permet de gagner en taille de code.

Par défaut, l'option est à 1.

6.4.5 GFX

Lorsque cette option est à 1, elle permet de prendre la main à distance sur l'IHM à l'aide du logiciel microPAD et de la liaison Ethernet. Il faut donc un socket du port Ethernet de disponible pour cette fonctionnalité.

Lorsque cette option est à 0, elle permet de gagner en taille de code.

Par défaut, l'option est à 0.

6.4.6 HTTP

Lorsque cette option est à 1, elle permet d'avoir accès aux serveurs web. Il faut donc un socket du port Ethernet de disponible. Si l'application dispose d'un IHM développé par MicroHMI, ce sera l'application MicroHMI qui sera visualisée (il faut alors 2 sockets), si l'application ne dispose pas d'IHM, ce sera un accès

au répertoire www de la carte SD. Il est aussi possible de faire de la communication ModBus à travers le protocole HTTP.

Lorsque cette option est à 0, elle permet de gagner en taille de code.

Par défaut, l'option est à 0.

6.4.7 LCD

Cette option est valable depuis la version 10.14 de MicroLADDER du 09/06/16.

Lorsque cette option est à 1, elle permet de gérer l'écran LCD de l'automate.

Lorsque cette option est à 0, elle permet de gagner jusqu'à 50 ko de code.

Par défaut, l'option est à 1.

6.4.8 RF

Cette option est valable depuis la version 14.0 de MicroLADDER et la version 11.0 du code système. Elle permet d'activer la gestion du module Radio Fréquence LoRa sur un port série pré-attribué sur chaque automate.

Par défaut, l'option est à 0.

6.5 Fonctions utiles

6.5.1 Communication sans protocole

Pour paramétrer une communication sans protocole, il faut mettre le mot système SERx, SOCKx ou WSOCKx à la valeur COM_PROTOCOL_NONE (valeur 0).

6.5.1.1 Paramétrage

void ComSetCharTimeout (unsigned char comPort, unsigned short timeout);

Cette fonction permet de définir le timeout inter-caractères. Cela est utile avec une communication radio où les caractères peuvent arriver avec un peu plus de retard. Cette fonction est valable depuis le code système 8.0 du 01/06/16.

- **unsigned char comPort** : numéro du port de communication. Il est préférable d'utiliser les variables prédéfinies:

COM_PORT_SER0, COM_PORT_SER1, COM_PORT_SER2, COM_PORT_SER3, COM_PORT_SER4, COM

- **unsigned short timeout**: valeur du timeout en ms.

Exemple:

```
ComSetCharTimeout (COM_PORT_SER0, 100);
```

6.5.1.2 Emission

unsigned short ComPush (unsigned char comPort, unsigned char *str, int l);

Cette fonction écrit une chaîne de caractères dans un buffer temporaire. Elle est utilisée avant d'envoyer une chaîne de caractères à l'automate.

- **unsigned char comPort** : numéro du port de communication. Il est préférable d'utiliser les variables prédéfinies

COM_PORT_SER0, COM_PORT_SER1, COM_PORT_SER2, COM_PORT_SER3, COM_PORT_SER4,
COM_PORT_SER5, COM_PORT_SER6, COM_PORT_SER7, COM_PORT SOCK0, COM_PORT SOCK1,
COM_PORT SOCK2, COM_PORT SOCK3, COM_PORT_WSOCK0, COM_PORT_WSOCK1,
COM_PORT_WSOCK2, COM_PORT_WSOCK3.

- **unsigned char *str** : chaîne de caractères de type unsigned char qui va servir de buffer d'envoi.
- **int l** : nombre de caractères à envoyer de type unsigned short.

Exemple :

```
char s[] = "Test 123";  
ComPush (COM_PORT_SER0, s, 8);
```

unsigned char ComPushByte (unsigned char comPort, unsigned char b);

Cette fonction écrit un octet dans un buffer temporaire à chaque fois qu'elle est exécutée. Cette fonction est une alternative à la fonction *ComPush* et elle est utilisée avant d'envoyer une chaîne de caractères à l'automate.

- **unsigned char comPort** : numéro du port de communication. Il est préférable d'utiliser les variables prédéfinies

COM_PORT_SER0, COM_PORT_SER1, COM_PORT_SER2, COM_PORT_SER3, COM_PORT_SER4, COM

- **unsigned char b**: caractères de type unsigned char que l'on envoie dans un buffer.

Exemple :

```
ComPushByte (COM_PORT_SER0, 'T');  
ComPushByte (COM_PORT_SER0, 'e');  
ComPushByte (COM_PORT_SER0, 's');  
ComPushByte (COM_PORT_SER0, 't');  
ComPushByte (COM_PORT_SER0, ' ');  
ComPushByte (COM_PORT_SER0, '1');  
ComPushByte (COM_PORT_SER0, '2');
```

```
ComPushByte (COM_PORT_SER0, '3');
```

void ComSend (unsigned char comPort);

Cette fonction écrit une chaîne de caractères, préalablement écrite dans un buffer, sur le port série, le port Ethernet ou Wi-Fi. La fonction *ComPush* ou *ComPushByte* doit être exécutée avant afin de réaliser le stockage de la chaîne de caractères à envoyer.

Lors de l'émission sur un port Ethernet ou Wi-Fi, la fonction ne rend la main qu'à la fin de l'émission. Lors de l'émission sur un port série, la fonction rend la main immédiatement et l'émission se fait en différée sous interruption. Ainsi, pour connaître la fin de l'émission, il faut regarder le mot système RES_SERx ou utiliser la fonction *ComFlushOutput*. Le mot RES_SERx est mis à jour à la fin du tour de cycle automate. Il ne faut donc pas bloquer le programme en attendant un changement de son état.

- **unsigned char comPort** : numéro du port de communication. Il est préférable d'utiliser les variables prédéfinies

```
COM_PORT_SER0, COM_PORT_SER1, COM_PORT_SER2, COM_PORT_SER3, COM_PORT_SER4,
COM_PORT_SER5, COM_PORT_SER6, COM_PORT_SER7, COM_PORT SOCK0, COM_PORT SOCK1,
COM_PORT SOCK2, COM_PORT SOCK3, COM_PORT_WSOCK0, COM_PORT_WSOCK1,
COM_PORT_WSOCK2, COM_PORT_WSOCK3.
```

Exemple 1 :

```
char s[] = "Test 123";
ComPush (COM_PORT_SER0, s, 8);
ComSend (COM_PORT_SER0);
```

Exemple 2 :

```
ComPushByte (COM_PORT_SER0, 'T');
ComPushByte (COM_PORT_SER0, 'e');
ComPushByte (COM_PORT_SER0, 's');
ComPushByte (COM_PORT_SER0, 't');
ComPushByte (COM_PORT_SER0, ' ');
ComPushByte (COM_PORT_SER0, '1');
ComPushByte (COM_PORT_SER0, '2');
ComPushByte (COM_PORT_SER0, '3');
ComSend (COM_PORT_SER0);
```

void ComRealFlushOutput (unsigned char comPort);

Cette fonction attend la fin de l'émission. Elle est donc bloquante.

- **unsigned char comPort** : numéro du port de communication. Il est préférable d'utiliser les variables prédéfinies

```
COM_PORT_SER0, COM_PORT_SER1, COM_PORT_SER2, COM_PORT_SER3, COM_PORT_SER4,
COM_PORT_SER5, COM_PORT_SER6, COM_PORT_SER7, COM_PORT SOCK0, COM_PORT SOCK1,
COM_PORT SOCK2, COM_PORT SOCK3, COM_PORT_WSOCK0, COM_PORT_WSOCK1,
COM_PORT_WSOCK2, COM_PORT_WSOCK3.
```

Exemple :

ComFlushOutput (COM_PORT_SER1);

6.5.1.3 Réception

unsigned short ComGetFrameLength (unsigned char comPort);

Cette fonction permet de savoir combien de caractères ont été reçus. La réception se fait dans un buffer cyclique de 256 caractères. Lorsque le buffer est plein, la réception repart au début.

- **unsigned short** (valeur de retour) : variable de retour de type unsigned short. Elle indique le nombre de caractères reçu dans le buffer. Au départ elle vaut 0. Elle s'incrémente jusqu'à 255, passe à 256 et repasse à 1. De ce fait, elle pointe vers l'emplacement du prochain caractère sauf lorsqu'elle indique 256.
- **unsigned char comPort** : numéro du port de communication. Il est préférable d'utiliser les variables prédéfinies

COM_PORT_SER0, COM_PORT_SER1, COM_PORT_SER2, COM_PORT_SER3, COM_PORT_SER4,
COM_PORT_SER5, COM_PORT_SER6, COM_PORT_SER7, COM_PORT SOCK0, COM_PORT SOCK1,
COM_PORT SOCK2, COM_PORT SOCK3, COM_PORT_WSOCK0, COM_PORT_WSOCK1,
COM_PORT_WSOCK2, COM_PORT_WSOCK3.

Exemple:

Long = ComGetFrameLength (COM_PORT_SER1);

unsigned char *ComGetFrame (unsigned char comPort);

Cette fonction permet de récupérer les caractères reçus. Elle renvoie un pointeur sur le début de la chaîne de réception. Après avoir traité les caractères reçus, il faut utiliser la fonction *ComFlushInput* pour que la prochaine réception commence au début du buffer.

- **unsigned char *** (valeur de retour) : variable de retour pointant sur la chaîne de caractères reçus par l'automate
- **unsigned char comPort** : numéro du port de communication. Il est préférable d'utiliser les variables prédéfinies suivantes:

COM_PORT_SER0, COM_PORT_SER1, COM_PORT_SER2, COM_PORT_SER3, COM_PORT_SER4,
COM_PORT_SER5, COM_PORT_SER6, COM_PORT_SER7, COM_PORT SOCK0, COM_PORT SOCK1,
COM_PORT SOCK2, COM_PORT SOCK3, COM_PORT_WSOCK0, COM_PORT_WSOCK1,
COM_PORT_WSOCK2, COM_PORT_WSOCK3.

Exemple :

```
char *Chaine;
Chaine = ComGetFrame(COM_PORT_SER1);
%MW20:=Chaine[0];
%MW21:=Chaine[1];
%MW22:=Chaine[2];
```

void ComFlushInput (unsigned char comPort);

Cette fonction remet à 0 le nombre de caractères reçus. De plus, pour les ports série, elle efface la chaîne de caractères de réception.

- **unsigned char comPort** : numéro du port de communication. Il est préférable d'utiliser les variables prédéfinies suivantes:

COM_PORT_SER0, COM_PORT_SER1, COM_PORT_SER2, COM_PORT_SER3, COM_PORT_SER4,
COM_PORT_SER5, COM_PORT_SER6, COM_PORT_SER7, COM_PORT SOCK0, COM_PORT SOCK1,
COM_PORT SOCK2, COM_PORT SOCK3, COM_PORT_WSOCK0, COM_PORT_WSOCK1,
COM_PORT_WSOCK2, COM_PORT_WSOCK3.

Exemple :

```
ComFlushInput (COM_PORT_SER1);
```

void ComClearFrame (unsigned char comPort, int l);

Cette fonction efface du tampon de réception le nombre de caractères demandé et déplace les autres caractères vers le début. Elle est donc très utile pour effacer uniquement les caractères qui ont été traités.

- **unsigned char comPort** : numéro du port de communication. Il est préférable d'utiliser les variables prédéfinies

COM_PORT_SER0, COM_PORT_SER1, COM_PORT_SER2, COM_PORT_SER3, COM_PORT_SER4,
COM_PORT_SER5, COM_PORT_SER6, COM_PORT_SER7, COM_PORT SOCK0, COM_PORT SOCK1,
COM_PORT SOCK2, COM_PORT SOCK3, COM_PORT_WSOCK0, COM_PORT_WSOCK1,
COM_PORT_WSOCK2, COM_PORT_WSOCK3.

- **int l** : nombre de caractères à effacer.

Exemple :

```
ComClearFrame (COM_PORT_SER1, 10);
```

6.5.2 Communication avec le protocole Modbus

Pour plus de détails, se référer à la section [6.2.13.1 Détail Modbus](#) avec notamment les variables %SW34 à %SW175. Il est possible de faire du Modbus standard ou du Modbus étendu (Modbus Sirea) sur tous les ports. Cela signifie qu'il est possible de faire le chargement d'une application par tous les ports. Par défaut, tous les ports sont paramétrés en Modbus Sirea esclave n°1 pour les ports série et Modbus TCP esclave n°1 pour les ports Ethernet et Wi-Fi.

6.5.2.1 Modbus esclave et Modbus TCP esclave

Il suffit de renseigner NESC_SERx, NESC SOCKx ou NESC_WSOCKx avec le numéro d'esclave et SERx, SOCKx ou WSOCKx avec le protocole (variable COM_PROTOCOL_MODBUS pour du Modbus standard ou COM_PROTOCOL_MODBUS_TCP pour du Modbus TCP). Pour le Modbus TCP, il faut aussi que PORT SOCKx ou PORT_WSOCKx soit à 502. Le système se charge de répondre aux

requêtes reçues.

6.5.2.2 Modbus maître

Il suffit de renseigner SERx, SOCKx ou WSOCKx sans protocole (variable COM_PROTOCOL_NONE). NESC_SERx ne doit pas forcément être à 0.

La gestion des trames de lecture et d'écriture est à l'initiative de l'application.

unsigned char ModbusRead (unsigned char comPort, unsigned char tcp, unsigned char slave, unsigned char type, unsigned short masterAdd, unsigned short nbValues, unsigned short slaveAdd);

Cette fonction permet la lecture de mots ou bits sur un esclave Modbus. La vitesse et le format de communication sont paramétrés dans les mots systèmes. La fonction doit être lancée sur front. Elle va ensuite prendre un certain temps pour s'exécuter. Il est possible de connaître son état d'évolution par la variable RES_SERx, RES SOCKx ou RES_WSOCKx dans les mots systèmes.

- **unsigned char** (valeur de retour) : variable de retour de type unsigned char. Elle vaut 1 si la fonction est lancée. Elle vaut 0 s'il y a une erreur de paramètre qui empêche de lancer la fonction.
- **unsigned char comPort** : numéro de port. Il est préférable d'utiliser les variables prédéfinies :

COM_PORT_SER0, COM_PORT_SER1, COM_PORT_SER2, COM_PORT_SER3, COM_PORT_SER4, COM

- **unsigned char tcp**: type de Modbus (0 : Modbus classique et 1 : Modbus TCP).
- **unsigned char slave** : numéro d'esclave.
- **unsigned char type** : type de variable à lire (bit ou mot, mémoire ou entrée). Il est préférable d'utiliser les variables prédéfinies:

MODBUS_TYPE_MW, MODBUS_TYPE_M, MODBUS_TYPE_IW, MODBUS_TYPE_I

- **unsigned short masterAdd** : adresse de la première donnée dans la mémoire du maître pour le stockage.
- **unsigned short nbValues** : nombre de données à lire.
- **unsigned short slaveAdd** : adresse de la première donnée dans l'esclave.

unsigned char ModbusWrite (unsigned char comPort, unsigned char tcp, unsigned char slave, unsigned char type, unsigned short masterAdd, unsigned short nbValues, unsigned short slaveAdd, unsigned char forceMultFunc);

Cette fonction permet l'écriture de mots ou bits sur un esclave Modbus. La vitesse et le format de communication sont paramétrés dans les mots systèmes. La fonction doit être lancée sur front. Elle va ensuite prendre un certain temps pour s'exécuter. Il est possible de connaître son état d'évolution par la variable RES_SERx, RES SOCKx ou RES_WSOCKx dans les mots systèmes.

- **unsigned char** (valeur de retour) : variable de retour de type unsigned char. Elle vaut 1 si la fonction est lancée. Elle vaut 0 s'il y a une erreur de paramètre qui empêche de lancer la fonction.
- **unsigned char comPort** : numéro de port. Il est préférable d'utiliser les variables prédéfinies :

COM_PORT_SER0, COM_PORT_SER1, COM_PORT_SER2, COM_PORT_SER3, COM_PORT_SER4, COM

- **unsigned char tcp** : type de Modbus (0 : Modbus classique, 1 : Modbus TCP).
- **unsigned char slave** : numéro d'esclave.
- **unsigned char type** : type de variable à écrire (bit ou mot). Il est préférable d'utiliser les variables prédéfinies :

MODBUS_TYPE_MW, MODBUS_TYPE_M.

- **unsigned short masterAdd** : adresse de la première donnée dans la mémoire du maître pour l'émission.
- **unsigned short nbValues** : nombre de donnée à écrire.
- **unsigned short slaveAdd** : adresse de la première donnée dans l'esclave.
- **unsigned char forceMultFunc** : valeur à mettre à 1 pour forcer l'utilisation de la fonction d'écriture de plusieurs mots (code fonction 10h au lieu de 06h) ou plusieurs bits (code fonction 0Fh au lieu de 05h). Mettre 0 dans le cas contraire et le système sélectionne le code fonction en fonction du nombre d'éléments à écrire.

6.5.2.3 Modbus TCP maître

Il faut gérer une ouverture et une fermeture de socket. La gestion des trames est identique au Modbus sur port série. Il faut faire attention à bien renseigner les deux premiers paramètres des fonctions *ModbusRead* et *ModbusWrite*. Avant chaque lecture ou écriture, il faut recontrôler l'état du socket qui peut avoir perdu sa connexion. Depuis le 02/10/13, il n'est plus nécessaire de régler le port à 502 dans les variables `PORT_SOCKET*` et `PORT_WSOCKET*` (cela est fait dans la nouvelle fonction).

void ComConnect (unsigned char comPort, char *host, unsigned short port);

Cette fonction ouvre le socket. Il faut ensuite tester l'état du socket avant de générer des trames.

- **unsigned char comPort** : numéro du port de communication. Mettre `COM_PORT_SOCKET0`, `COM_PORT_SOCKET1`, `COM_PORT_SOCKET2`, `COM_PORT_SOCKET3` pour les socket Ethernet et mettre `COM_PORT_WSOCKET0`, `COM_PORT_WSOCKET1`, `COM_PORT_WSOCKET2`, `COM_PORT_WSOCKET3` pour les sockets Wi-Fi. Cette valeur doit être en accord avec le premier paramètre des fonctions *ModbusRead* et *ModbusWrite*.
- **char *host**: chaîne de caractères contenant l'adresse IP ou l'adresse si le DNS est activé.
- **unsigned short port** : numéro du port du client ou du serveur auquel il faut se connecter. Par exemple 502 pour du Modbus.

void ComClose (unsigned char comPort);

Cette fonction ferme le socket.

- **unsigned char comPort**: numéro du port de communication. Mettre `COM_PORT_SOCKET0`, `COM_PORT_SOCKET1`, `COM_PORT_SOCKET2`, `COM_PORT_SOCKET3` pour les socket Ethernet et mettre `COM_PORT_WSOCKET0`, `COM_PORT_WSOCKET1`, `COM_PORT_WSOCKET2`, `COM_PORT_WSOCKET3` pour les sockets Wi-Fi. Cette valeur doit être en accord avec le premier paramètre des fonctions *ModbusRead* et *ModbusWrite*.

unsigned char ComGetSockState (unsigned char comPort);

Cette fonction lit l'état du socket.

- **unsigned char** (valeur de retour) : variable de retour de type unsigned char. Elle vaut SOCK_STATE_CLOSED (valeur 0) si le socket est fermé. Elle vaut SOCK_STATE_READY (valeur 1) si le socket est connecté (pour une fonction maître ou esclave). Elle vaut SOCK_STATE_CONNECT (valeur 2) si le socket est en cours de connexion. Elle vaut SOCK_STATE_CLOSE (valeur 3) si le socket est en cours de fermeture.
- **unsigned char comPort** : numéro du port de communication. Mettre COM_PORT_SOCKET0, COM_PORT_SOCKET1, COM_PORT_SOCKET2, COM_PORT_SOCKET3 pour les socket Ethernet et mettre COM_PORT_WSOCKET0, COM_PORT_WSOCKET1, COM_PORT_WSOCKET2, COM_PORT_WSOCKET3 pour les sockets Wi-Fi. Cette valeur doit être en accord avec le premier paramètre des fonctions *ModbusRead* et *ModbusWrite*.

Remarque : La tentative de connexion d'un socket peut ne pas fonctionner. Il faut donc que l'application gère un timeout et gère la non connexion.

6.5.3 Gestion des fichiers

La gestion des fichiers sur les automates peut se faire sur la carte SD mais aussi sur une clé USB. Ces deux supports doivent être formatés en FAT32.

Information

Seuls certains automates ont des ports USB nous permettant de brancher une clé

Les opérations de manipulation de fichiers sont bloquantes. Elles prennent du temps, surtout lorsqu'il faut parcourir la carte ou la clé pour trouver les secteurs libres. Elles ne peuvent pas être traitées dans une interruption. Ainsi, la plupart des fonctions de gestion des fichiers génèrent des ResetWatchdog afin de ne pas provoquer de

dépassement de watchdog. Un ResetWatchdog va permettre de mettre la valeur de la temporisation d'exécution à 0 pour ne pas qu'un watchdog soit levé, bloquant le programme.

La carte ou la clé est divisée en clusters. Chaque cluster est à nouveau divisé en 4 secteurs de 512 octets. Pour écrire des données sur un secteur, il faut lire tout le secteur, modifier les octets et réécrire les 512 octets.

L'insertion de la carte SD à chaud exécute un cFatInit qui dure 1255ms. Les autres opérations prennent de 30 à 100ms.

6.5.3.1 Structure des noms de fichier

Il faut indiquer le type de support (SD pour carte SD ou USB pour une clé USB) suivi de « : », le nom du fichier (maximum 8 caractères) et l'extension (maximum 3 caractères). Le système ne fait pas la différence entre les majuscules et les minuscules.

Exemple :

```
char *achTmpFileName = "USB:LOG.TMP";
char *achTmpFileName = "SD:LOG.TMP";
```

6.5.3.2 Structure « FSFile »

La structure « FSFile » est une structure qui contient des informations durant le travail sur un fichier (de son ouverture à sa fermeture). Pour des raisons d'inutilité, on ne décrira pas cette structure dans cette documentation car il n'y a pas d'action particulière à faire avec cette structure, ce sont les fonctions que l'on va appeler pour travailler sur le fichier qui vont impacter cette variable.

S'il s'écoule plusieurs tours de cycle entre l'ouverture et la fermeture du fichier, il faut que cette variable soit de type global.

unsigned char FSOpen (FSFile *fd, char *name, unsigned char type);

Cette fonction permet d'ouvrir le fichier avant de faire une lecture ou une écriture.

- **unsigned char** (valeur de retour) variable de type unsigned char. Elle vaut 1 si l'action est correcte. Elle vaut 0 en cas d'erreur.
- **FSFile *fd** : adresse d'une variable de type FSFile.
- **char *name** : chaîne de caractères contenant le nom du fichier.
- **unsigned char type** : indique l'ouverture du fichier pour une lecture (FA_READ), pour un ajout (FA_APPEND) ou pour une écriture au début du fichier (valeur FA_WRITE).

Exemple :

```
FSFile fd;  
unsigned char chResult;  
chResult = FSOpen (&fd, "SD:SIREA.TXT", FA_WRITE);
```

void FSClose (FSFile *fd);

Cette fonction permet de fermer le fichier lorsque les écritures sont terminées. Il n'est pas nécessaire d'appeler cette fonction si le fichier a été ouvert pour une lecture.

- **FSFile *fd** : adresse d'une variable de type FSFile.

Exemple :

```
FSFile fd;  
FSClose (&fd);
```

unsigned char FSSeek (FSFile *fd, unsigned long position);

Cette fonction permet de se positionner dans un fichier. Il faut exécuter la fonction *FSOpen* avant.

- **unsigned char** (valeur de retour): variable de type unsigned char. Elle vaut 1 si l'action est correcte. Elle vaut 0 si la position demandée est en dehors du fichier.
- **FSFile *fd** : adresse d'une variable de type FSFile.
- **unsigned long position** : octet à atteindre.

Exemple :

```
FSFile fd;  
unsigned char chResult;  
chResult = FSSeek (&fd, 10);
```

unsigned char FSDelete (char *name);

Cette fonction permet d'effacer un fichier.

- **unsigned char** (valeur de retour): variable de type unsigned char. Elle vaut 1 si l'action est correcte. Elle vaut 0 en cas d'erreur.
- **char *name** : chaîne de caractères contenant le nom du fichier.

Exemple :

```
unsigned char chResult;  
chResult = FSDelete ("SD:LOG.TMP");
```

unsigned char FSCopy (char *src, char *dst);

Cette fonction permet de copier un fichier.

- **unsigned char** (valeur de retour): variable de type unsigned char. Elle vaut 1 si l'action est correcte. Elle vaut 0 en cas d'erreur.
- **char *src**: chaîne de caractères contenant le nom du fichier source.
- **char *dst** : chaîne de caractères contenant le nom du fichier destination.

Exemple :

```
unsigned char chResult;  
chResult = FSCopy ("SD:LOG.TMP", "SD:LOG2.TMP");
```

unsigned char FSMove (char *src, char *dst);

Cette fonction permet de renommer un fichier. Elle permet aussi de faire un déplacement si le dossier source est différent du dossier destination. Les dossiers source et destination doivent être sur le même support (clé USB ou carte SD).

- **unsigned char** (valeur de retour): variable de type unsigned char. Elle vaut 1 si l'action est correcte. Elle vaut 0 en cas d'erreur.
- **char *src** : chaîne de caractères contenant le nom du fichier source.
- **char *dst** : chaîne de caractères contenant le nom du fichier destination.

Exemple :

```
unsigned char chResult;  
chResult = FSMove ("SD:LOG.TMP", "SD:LOG2.TMP");
```

unsigned long FSRead (FSFile *fd, char *data, unsigned long size);

Cette fonction permet de lire une quantité de caractères. Il faut exécuter la fonction *FSOpen* avant.

- **unsigned long** (valeur de retour) : variable de type unsigned long. Elle contient le nombre de caractères lus.
- **FSFile *fd** : adresse d'une variable de type FSFile.

- **char *data** : chaîne de caractères qui contiendra les caractères lus.
- **unsigned long size** : nombre de caractères à lire.

Exemple :

```
SFile fd;
char strChaine[20];
unsigned long lTaille;
lTaille = FSRead (&fd, strChaine, 10);
```

unsigned char FSWrite (FSFile *fd, char *data, long size);

Cette fonction permet d'écrire une quantité de caractères dans le fichier. Il faut exécuter la fonction *FSOpen* avant et la fonction *FSClose* après.

- **unsigned char** (valeur de retour): variable de type unsigned char. Elle vaut 1 si l'action est correcte. Elle vaut 0 en cas d'erreur.
- **FSFile *fd** : adresse d'une variable de type FSFile.
- **char *data** : chaîne de caractères qui contient les caractères à écrire.
- **long size** : nombre de caractères à écrire. Ce paramètre peut être remplacé par une valeur négative pour que la fonction calcule elle même la longueur.

Exemple :

```
FSFile fd;
char *strChaine = "Bonjour";
unsigned char chResult;
chResult = FSWrite (&fd, strChaine, 7);
```

unsigned char FSReadLine (FSFile *fd, char *data, unsigned short maxLen, unsigned char *broken);

Cette fonction permet de lire une ligne du fichier. Il faut exécuter la fonction *FSOpen* avant.

- **unsigned char** (valeur de retour): variable de type unsigned char. Elle vaut 1 si l'action est correcte. Elle vaut 0 en cas d'erreur.
- **FSFile *fd** : adresse d'une variable de type FSFile.
- **char *data** : chaîne de caractères qui contiendra les caractères lus.
- **unsigned short maxLen** : nombre maximum de caractères à lire.
- **unsigned char *broken** : adresse d'une variable de type unsigned char. La fonction renverra dans cette variable 0 si la ligne retournée est complète et renvoie 1 si la ligne retournée est incomplète.

Exemple :

```
FSFile fd;
char strChaine[20];
unsigned char chRetour;
unsigned char chResult;
chResult = FSReadLine (&fd, strChaine, 20, &chRetour);
```

unsigned char FSReadCSVRow (FSFile *fd, unsigned short *nbCols, char *data, unsigned short maxLen);

Cette fonction permet de lire une ligne de donnée contenant plusieurs colonnes. Il faut faire exécuter la fonction *FSOpen* avant.

- **unsigned char** (valeur de retour) : variable de type unsigned char. Elle vaut 1 si l'action est correcte. Elle vaut 0 en cas d'erreur.
- **FSFile *fd** : adresse d'une variable de type FSFile.
- **unsigned short *nbCols** : adresse d'une variable de type unsigned short. La fonction renverra dans cette variable le nombre de colonnes trouvées.
- **char *data** : chaîne de caractères qui contiendra les caractères lus. Les différentes colonnes sont séparées par 0.
- **unsigned short maxLen** : nombre maximum de caractères à lire.

Exemple :

```
FSFile fd;
char strChaine[20];
unsigned short nbCol;
unsigned char chResult;
chResult = FSReadCSVRow (&fd, &nbCol, strChaine, 20);
```

unsigned char FSWriteCSVRow (FSFile *fd, unsigned short nbCols, char *data);

Cette fonction permet d'écrire une ligne de donnée contenant plusieurs colonnes. Il faut exécuter la fonction *FSOpen* avant et la fonction *FSClose* après.

- **unsigned char** (valeur de retour): variable de type unsigned char. Elle vaut 1 si l'action est correcte. Elle vaut 0 en cas d'erreur.
- **FSFile *fd**: adresse d'une variable de type FSFile.
- **unsigned short nbCols** : nombre de colonnes à écrire.
- **char *data** : chaîne de caractères contenant les caractères à écrire. Les différentes colonnes sont séparées par 0.

Exemple :

```
FSFile fd;
char *strChaine = "Col1-Col2";
chaine[4] = 0;
unsigned char chResult;
chResult = FSWriteCSVRow (&fd, 2, strChaine);
```

unsigned char FSCreateFolder (char *name);

Cette fonction permet de créer un répertoire. Elle n'est valable que depuis le code système 8.14 ou 9.14.

- **unsigned char** (valeur de retour) : variable de type unsigned char. Elle vaut 1 si l'action est correcte. Elle vaut 0 en cas d'erreur (répertoire déjà existant, problème d'écriture sur le support,

...).

- **char *name** : chaîne de caractères contenant le nom du répertoire et son chemin.

Exemple :

```
%M0 := FSCreateFolder ("test1");
%M1 := FSCreateFolder ("test1/test2");
%M2 := FSCreateFolder ("test1/test2/test3");
```

6.5.4 Gestion des journaux

6.5.4.1 Format timestamp

C'est une variable de type « long » contenant le nombre de secondes écoulées depuis le 1er janvier 1970. Une variable de type timestamp peut être convertie sous format « Date ». Les fonctions *dateToTime* et *timeToDate* sont disponibles pour effectuer ces conversions.

L'horodatage est en temps UTC. La date est en heure locale. Elle est donc influencée par le décalage été/hiver et le fuseau horaire.

6.5.4.2 Structure « Date »

Une structure Date est constituée de plusieurs éléments :

```
int sec   Secondes [0, 59]
int min   Minutes [0, 59]
int hour  Heure commençant à minuit [0, 23]
int mday  Jour du mois [1, 31]
int mon   Mois, commençant par janvier [1, 12]
int year  Année
int wday  Jour commençant par le dimanche [0, 6]
int yday  Jour commençant au 1er janvier [0, 365]
int isdst  Heure d'été ou d'hiver
Tableau sur la structure « Date »
```

6.5.4.3 Fonctions de conversions Date/timestamp

unsigned long dateToTime (Date d);

Cette fonction convertit une structure « Date » en timestamp. Elle fonctionne du 01/01/1970 00:00:00 (valeur de retour = 0) au 01/19/2038 03:14:07 (valeur de retour = 2147483647). Sinon, le retour est -1.

- **unsigned long** (valeur de retour) : variable de type long
- **Date d** : variable de structure « Date ».

Exemple :

```
long LTSDate ;
Date d ;
d.mday = 25 ;
d.mon = 12 ;
d.year = 2010 ;
LTSDate = dateToTime(d) ;
```

Date timeToDate (unsigned long time);

Cette fonction convertit un timestamp en structure « Date ».

- La valeur 2147483647 renvoie 01/19/2038 03:14:07.
- La valeur 0 renvoie 01/01/1970 00:00:00.
- La valeur – 2147483648 envoie le 19/01/2038 03:14:08.
- La valeur -1 envoie le 02/07/2106 06:28:15.
- **Date** (valeur de retour) : variable de structure « Date ».
- **unsigned long time** : timestamp que l'on souhaite convertir.

Exemple :

```
long LTSDate ;
Date d ;
d = timeToDate(LTSDate) ;
Heure = d.hour ;
```

6.5.4.4 Format LogValue

Cette structure permet de récupérer les données enregistrées. Elle comprend les éléments suivants :

unsigned long time;	Horloge de l'enregistrement des données.
LogVariable *var;	Pointeur sur les données historiques
char *string;	chaîne de caractères de l'événement ou de l'alarme. Cette chaîne est NULL si elle est inexistante.
unsigned char type;	Lors de la récupération d'événements, il s'agit de LOG_ENTRY_TYPE_EVENT (valeur 1) dans le cas d'un message, LOG_ENTRY_TYPE_ALARM_ON (valeur 2) dans le cas de l'apparition d'une erreur et LOG_ENTRY_ALARM_OFF (valeur 3) dans le cas d'une disparition d'erreur.
double value;	Valeur de la variable enregistrée. Pour la récupération d'un événement, la valeur de ce champ est la valeur de la variable.

Tableau sur la structure LogValue

6.5.4.5 Fonctions de gestions des données historisées

unsigned short LogAlQuery (unsigned long minTime, unsigned long maxTime, unsigned char order);

Cette fonction ouvre la base de données pour récupérer les défauts présents. La fonction *LogFetch* doit ensuite être exécutée dans le même cycle pour la récupération des données.

- **unsigned short** (valeur de retour) : variable de type unsigned short. Elle correspond au nombre d'alarmes récupérées pour la période demandée.
- **unsigned long minTime** : date de début de recherche au format timestamp. Elle est définie à 0 pour une date illimitée.
- **unsigned long maxTime** : date de fin de recherche au format timestamp. La valeur 0 correspond à l'illimitée.
- **unsigned char order** : ordre de tri. Il faut utiliser LOG_QUERY_ORDER_ASC pour un ordre croissant des données (le registre le plus ancien en premier) ou LOG_QUERY_ORDER_DESC pour un ordre descendant (le registre le plus ancien en dernier).

Exemple :

```
long tsStart ;
long tsEnd ;
unsigned short nbrErr ;
nbrErr = LogAlQuery (tsStart, tsEnd, LOG_QUERY_ORDER_DESC) ;
```

unsigned short LogEvQuery (unsigned long minTime, unsigned long maxTime, unsigned char order);

Cette fonction ouvre la base de données pour récupérer le journal de tous les défauts qui sont apparus, qui sont disparus et de toutes les messages qui sont apparus.

La fonction *LogFetch* doit ensuite être exécutée dans le même cycle pour la récupération des données. C'est dans le champ « Type » de la variable de structure LogValue qui peut faire la différence entre les trois différents types de données.

- **unsigned short** (valeur de retour) : variable de type unsigned short. Elle correspond au nombre d'événements récupérés pour la période demandée.
- **unsigned long minTime** : date de début de recherche au format timestamp. Elle est définie à 0 pour une date illimitée.
- **unsigned long maxTime** : date de fin de recherche au format timestamp. La valeur 0 correspond à l'illimitée.
- **unsigned char order** : ordre de tri. Il faut utiliser LOG_QUERY_ORDER_ASC pour avoir par ordre croissant les données (le registre le plus ancien en première position) ou LOG_QUERY_ORDER_DESC pour les avoir dans un ordre décroissant (le registre le plus ancien en dernière position).

Exemple :

```
long lTSDebut ;
long lTSFin ;
unsigned short nNbrDef ;
nNbrEv = LogEvQuery (lTSDebut, lTSFin, LOG_QUERY_ORDER_DESC) ;
```

unsigned short LogTrQuery (LogVariable *var, unsigned long minTime, unsigned long maxTime, unsigned char order);

Cette fonction ouvre la base de données pour récupérer une valeur. La fonction *LogFetch* doit ensuite être exécutée dans le même cycle pour la récupération des données.

- **unsigned short** (valeur de retour) : variable de type unsigned short. Nombre de valeurs récupérées pour la période demandée.
- **LogVariable *var** : variable de type unsigned short. Index de la variable à extraire.
- **unsigned long minTime** : date de début de recherche au format horodateur. La valeur 0 correspond à une recherche illimitée.
- **unsigned long maxTime** : date de fin de recherche au format horodateur. La valeur 0 est illimitée.
- **unsigned char order** : ordre de tri. Il faut utiliser LOG_QUERY_ORDER_ASC pour avoir par ordre croissant les données (le registre le plus récent le plus ancien en première position) ou LOG_QUERY_ORDER_DESC pour les avoir dans un ordre décroissant (le registre le plus ancien en dernière position).

Exemple :

```
long lTSDebut ;
long lTSTFin ;
unsigned short nNbrDef ;
LogVariable *var = LogFindVariable("%MW20") ;
nNbrTr = LogEvQuery (var, lTSDebut, lTSTFin, LOG_QUERY_ORDER_DESC) ;
```

unsigned char LogFetch (LogValue *val);

Cette fonction récupère une donnée. Il faut faire plusieurs appels à cette fonction pour récupérer plusieurs données. Il faut utiliser les fonctions *LogAlQuery*, *LogEvQuery* ou *LogTrQuery* juste avant de récupérer un paquet de données. Tous le paquet de données doit être récupérés dans le même tour de cycle.

- **unsigned char** (valeur de retour) : variable de type unsigned char. Elle vaut 0 lorsqu'un problème survient.
- **LogValue *val** : structure de type LogValue. Elle contiendra les données.

Exemple :

```
LogValue stDef ;
LogFetch (&stDef) ;
```

unsigned char LogAlSave (char *fileName, unsigned long minTime, unsigned long maxTime);

Cette fonction permet de sauvegarder les alarmes sur une carte SD ou une clé USB.

- **unsigned char** (valeur de retour) : variable de type unsigned char. Elle vaut 0 lorsqu'un problème survient.
- **char *fileName** : chaîne de caractères contenant le nom du fichier.

- **unsigned long minTime** : date de début de recherche au format timestamp. Il faut mettre cette variable à 0 pour une recherche illimitée.
- **unsigned long maxTime** : date de fin de recherche au format timestamp. La valeur 0 correspond à une date illimitée.

Exemple :

```
long LTSDebut ;
long LTSFin ;
unsigned short ret ;
ret = LogAlSave ("SD:Alarme.csv", LTSDebut, LTSFin) ;
```

unsigned char LogEvSave (char *fileName, unsigned long minTime, unsigned long maxTime);

Cette fonction permet de sauvegarder les événements sur une carte SD ou une clé USB.

- **unsigned char** (valeur de retour) : variable de type unsigned char. Elle vaut 0 lorsqu'un problème survient.
- **char *fileName** : chaîne de caractères contenant le nom du fichier.
- **unsigned long minTime** : date de début de recherche au format timestamp. Il faut mettre cette variable à 0 pour une recherche illimitée.
- **unsigned long maxTime** : date de fin de recherche au format timestamp. La valeur 0 correspond à une date illimitée.

Exemple :

```
long LTSDebut ;
long LTSFin ;
unsigned short ret ;
ret = LogEvSave ("SD:Evenem.csv", LTSDebut, LTSFin) ;
```

unsigned char LogTrSave (char *fileName, LogVariable *var, unsigned long minTime, unsigned long maxTime);

Cette fonction permet de sauvegarder les valeurs sur une carte SD ou une clé USB.

Il faut utiliser la fonction *LogFindVariable* avant d'appeler cette fonction.

- **unsigned char** (valeur de retour) : variable de type unsigned char. Elle vaut 0 lorsqu'un problème survient.
- **char *fileName** : chaîne de caractères contenant le nom du fichier.
- **LogVariable *var** : numéro de la variable enregistrée.
- **unsigned long minTime** : date de début de recherche au format timestamp. Il faut mettre cette variable à 0 pour une recherche illimitée.
- **unsigned long maxTime** : date de fin de recherche au format timestamp. La valeur 0 correspond à une date illimitée.

Exemple :

```
long LTSDebut ;
long LTSFin ;
```

```
unsigned short ret ;
LogVariable *var = LogFindVariable("%MW20") ;
if (var) {ret = LogTrSave ("SD:Valeur.csv", var, lTSDebut, lTSFin);}
```

unsigned char LogSave (FSFile *fd);

Cette fonction est utilisée pour exporter le résultat d'une requête d'historique vers un fichier déjà ouvert. Cette fonction est valable depuis le code système 8.0 du 11/07/16.

- **unsigned char** (valeur de retour) : variable de type unsigned char. Elle vaut 0 s'il y a un problème.
- **FSFile *fd** : adresse d'une variable de type FSFile.

Exemple :

```
FSFile fd ;
char chRet
long lTSDebut ;
long lTSFin ;
unsigned short nNbrDef ;
LogVariable *var = LogFindVariable("%MW20") ;
nNbrTr = LogTrQuery (var, lTSDebut, lTSFin, LOG_QUERY_ORDER_DESC) ;
if (FSOpen (&fd, "SD:Extract.csv", FA_WRITE))
{
    chRet = LogSave (&fd) ;
    FSClose (&fd) ;
}
```

void LogPurge (void);

Cette fonction supprime tous les événements et les courbes de toutes les variables. Elle ne prend aucun paramètre en entrée ni en sortie.

Exemple :

```
LogPurge () ;
```

void LogEvPurge (void);

Cette fonction supprime tous les événements. Elle ne prend aucun paramètre en entrée ni en sortie.

Exemple :

```
LogEvPurge () ;
```

void LogTrPurge (LogVariable *var);

Cette fonction supprime le log complet d'une variable. Il faut utiliser la fonction *LogFindVariable* avant d'appeler cette fonction.

- **LogVariable *var** : numéro de la variable journalisée.

Exemple :

```
LogVariable *var = LogFindVariable("%MW20") ;
ret = LogTrPurge (var) ;
```

6.5.5 Chaînes de caractères

double StrToNum (char *s);

Cette fonction convertit une chaîne de caractères en une valeur numérique avec virgule. Elle ne traite que les premiers caractères numériques.

- **double** (valeur de retour) : variable de type double.
- **char *s** : La chaîne de caractères à convertir.

Exemple :

```
int nEntier = StrToNum ( "-200 " ) ;
float fFlottant = StrToNum ( "3.14 " ) ;
```

int sprintf(char *str, const char *string,...);

Cette fonction est similaire à la fonction standard du langage C *sprintf*. Elle est utilisée pour remplir des chaînes de caractères avec des valeurs numériques et/ou des caractères de chaîne.

Cette fonction peut être remplacée par *StrSet*, qui permet en outre de contrôler la longueur des chaînes de caractères.

- **int** (valeur de retour) : code de valeur de retour
- **char *str** : pointeur sur la chaîne de caractères à remplir.
- **const char *string** : structure de la chaîne de caractères à créer avec les éléments constants et le type des éléments variables. Un exemple de type est:
 - % c (caractère)
 - % l ou % d (entier)
 - % s (chaîne de caractères)
 - % f (floatant).
- **Paramètre 3 et suivants** : liste des éléments variables à placer dans la chaîne de caractères.

Exemple :

```
char * strUnit = "Bar" ;
Char strValeur [20] ;
int nMesure ;
sprintf (strValeur, "measure =% i% s ", nMesure, strUnit) ;
```

void StrToUpper (char *buf);

Cette fonction remplace les caractères minuscules d'une chaîne de caractères par des caractères majuscules.

Cette fonction peut-être utilisée à partir de la version 8.6 du 27/12/16 et les versions 9 du code système sinon elle est à remplacer par la fonction **void upper (char *buf);**

char *buf : pointeur sur la chaîne de caractères

Exemple :

```
char * strTexte = "Hello " ;  
Supérieur (StrTexte) ;
```

void StrToLower (char *buf);

Cette fonction remplace les caractères majuscules d'une chaîne de caractères par des caractères minuscules.

Cette fonction peut-être utilisée à partir de la version 8.6 du 27/12/16 et les versions 9 du code système sinon elle est à remplacer par la fonction **void lower (char *buf);**

char *buf : pointeur sur la chaîne de caractères

Exemple :

```
char * strTexte = "Hello " ;  
Supérieur (strTexte) ;
```

int StrSet (char *s, const char *format, ...);

Cette fonction est identique à *sprintf* mais elle effectue des vérifications sur la taille des chaînes de caractères. Il est donc préférable d'utiliser cette fonction.

int (valeur de retour) : code de valeur de retour

char *str : pointeur sur la chaîne de caractères à remplir.

const char *string : structure de la chaîne de caractères à créer avec les éléments constants et le type des éléments variables. Un exemple de type est:

- % c (caractère)
- % l ou % d (entier)
- % s (chaîne de caractères)
- % f (floatant).

Paramètre 3 et suivants : liste des éléments variables à placer dans la chaîne de caractères.

char StrGetChar (char *s, unsigned long pos);

Cette fonction est utilisée pour récupérer un caractère dans une chaîne de caractères.

char (valeur de retour) : variable char contenant le caractère récupéré ou 0 si la position demandée est en dehors de la chaîne de caractères.

char *s : pointeur sur la chaîne de caractères à examiner.

unsigned long pos : variable de type unsigned long indiquant l'emplacement du caractère recherché dans la chaîne de caractères.

Exemple :

```
Char ChCar ;  
chCar = StrGetChar(%MS20, 10) ;
```

unsigned char StrSetChar (char *s, unsigned long pos, char c);

Cette fonction permet de modifier un caractère dans une chaîne de caractères. Elle va vérifier si on essaye de changer un caractère en dehors de la chaîne de caractères. Elle est valable depuis la version 9.0 du 26/09/16 du code système.

unsigned char (valeur de retour) : variable de type unsigned char. Elle vaut 1 si l'action est correcte. Elle vaut 0 si l'action n'a pas pu être effectuée.

char *s : pointeur sur la chaîne de caractères à examiner.

unsigned long pos : une variable de type unsigned long indiquant la place du caractère à modifier dans la chaîne de caractères.

char c : caractère à modifier.

Exemple :

```
char ChCar = ' C ' ;  
unsigned char chRep = StrSetChar (%MS20, 10, ChCar) ;
```

7. IO Bus

Le Bus IO s'appuie sur le protocole Modbus pour communiquer entre un maître et des esclaves.

Le principe est d'établir la connexion entre les variables du maître et celles de l'esclave et le système gère automatiquement les trames de communication pour mettre à jour ces variables. Tout se passe du côté du maître. L'esclave répond simplement aux trames en suivant la norme Modbus.

7.1 Déclaration des équipements esclaves

Pour définir un esclave, cliquez sur l'onglet « Programme » dans la barre de menu de la fenêtre principale. Cliquez ensuite sur « Equipements » puis sur « Ajouter un équipements distants ». La fenêtre *popup* suivante s'ouvre:

Propriétés de l'équipement

Index : 1

Libellé : Nom

Port local : Série 0

Configuration série : 38400 8N1

Protocole : Modbus

Numéro d'esclave : 1

Délai (en ms) : 50

Timeout (en ms) : 500

Longueur maximale d'une trame :

☐ Regrouper les variables

Annuler Valider

Capture d'écran de la fenêtre *popup* « Propriétés de l'équipement »

Les champs suivants sont disponibles:

- **Index** : numéro de l'esclave. Chaque équipement est repéré par un numéro unique.
- **Libellé** : texte servant uniquement d'aide à la compréhension et qui n'a pas d'incidence dans le fonctionnement.
- **Port** : type de port. Il faut choisir entre Série, Ethernet, Wi-Fi ou Radio. Pour un port série, il faut préciser le numéro du port, la vitesse et le format de communication. Pour un port Ethernet ou Wi-Fi, il faut préciser l'adresse IP de l'esclave. Pour un port radio, il faut préciser le canal de fréquence.
- **Protocole** : protocole pour communiquer. Il faut choisir entre Modbus (c'est le ModBus standard), Sirea (c'est le ModBus étendu qui n'est utilisable qu'avec les appareil Sirea), Modbus (mots inversés) (les mots qui constituent des entiers longs et des flottants sont inversés par rapport au Modbus standard), Modbus (écritures multiples uniquement)(utilisé pour l'esclave 0 pour émettre des trames en broadcast à tous les esclaves) ou Modbus (mots inversés, écritures multiples uniquement).
- **Numéro d'esclave**: numéro d'esclave ModBus.
- **Délai** : délai intertrame en ms de la communication.
- **Timeout** : temps entre le début de l'attente d'une réponse et le passage en erreur afin de continuer la communication.
- **Longueur maximale d'une trame** : nombre d'octets maximal dans une trame, c'est à dire le nombre de mots multiplié par 2 ou le nombre de bits divisé par 8.

- **Regrouper les variables** : case à cocher qui permet lors de la lecture de données de créer des trames plus longues mais moins nombreuses en faisant la lecture de données non intéressantes.

7.2 Déclaration des variables

Dans la Déclaration des variables, dans l'onglet « Communication », le champ « Adresse distante » doit être rempli avec l'adresse de l'esclave suivie d'un point et du numéro de l'équipement.

Par exemple si on écrit %MW16.1 alors cela signifie que la variable maître sera liée à la variable %MW16 de l'équipement 1.

De plus, dans l'onglet « Communication », il est possible de mettre à l'échelle la valeur.

7.3 État de la communication avec les équipements

7.3.1 Structure statistique

Il existe des informations sur l'état de la communication avec les équipements. Elles sont contenues dans une structure « LogStats ». Pour pouvoir lire cette structure, il faut d'abord obtenir un pointeur sur l'équipement et ensuite obtenir un pointeur sur la structure stockant les statistique de l'équipement. Sirea a mis à disposition toute une bibliothèque de bloc fonction permettant d'obtenir simplement l'état de la communication.

Exemple pour obtenir la structure des statistique de l'équipement 1 :

```
LogDevice *dev = LogGetDevice (1) ;
LogStats stats = LogGetStats (dev) ;
```

7.3.2 Structure LogStats

unsigned long lastTimeOK	Horodatage de la dernière trame OK
unsigned long lastTimeERR	Erreur sur l'horodatage de la dernière trame
unsigned char lastState	Etat de la dernière trame (0 : pas de communication, 1 : communication OK, plus de 250 : erreur)
unsigned long nbFramesOK	Nombre de trames OK
unsigned long nbFramesERR	Erreur sur le nombre de trames

Tableau sur la structure LogStats

8. Protocole HTTP

Pour un automate avec un port Ethernet, il est possible de développer un serveur Web avec MicroHMI. Deux sockets doivent être utilisés. Le premier est configuré en mode COM_PROTOCOL_HTTP avec

son port 80 et le suivant est configuré en mode COM_PROTOCOL_HTTP_QUERY avec son port 81. L'utilisation de deux sockets permet d'avoir un affichage plus fluide et de charger correctement les images à être affichées.

Si l'automate dispose d'un port Ethernet, que l'option HTTP du code système est activée, et que l'application n'a pas d'IHM, le serveur Web pourra récupérer les fichiers dans le répertoire WWW de la carte SD. Il suffit d'indiquer dans un navigateur l'adresse IP de l'automate et le nom du fichier.

9. Wi-Fi

9.1 Mode

Il existe le mode Ad Hoc pour faire un point d'accès Wi-Fi à partir d'un automate et le mode client. C'est la variable %SW95 (W_MODE) qui est utilisée pour le gérer.

Constantes	Valeur	Description
WSOCK_MODE_OFF	0	Pas de connections WiFi
WSOCK_MODE_STA	1	Connexion en mode client
WSOCK_MODE_AP	2	Connexion en mode Ad Hoc

Tableau sur les modes de connexion Wi-Fi avec leur description

9.2 Fonctions utiles

void WSocketSetSSID (char *ssid);

Cette fonction permet d'initialiser le nom de la station Wi-Fi, aussi bien en mode client qu'en mode Ad Hoc.

Cette propriété peut également être définie avec le paramètre « W_SSID » du fichier main.cfg. La norme autorise jusqu'à 32 caractères.

- **char *ssid** : une chaîne de caractères contenant le SSID du réseau.

Exemple :

```
WSocketSetSSID (TP-LINK _ 9B4144) ;
```

void WSocketSetSecKey (char *key);

Cette fonction permet de définir la clé de sécurité, aussi bien en mode client qu'en mode Ad Hoc. Cette propriété peut également être définie avec le paramètre « W_SKEY » du fichier main.cfg.

- **char *key** : chaîne de caractères contenant la clé de sécurité

Exemple :

```
WSocketSetSecKey ("123456") ;
```

void WSocketSetSecType (char *type);

Cette fonction est utilisée pour régler le type de cryptage. Elle est particulièrement utile en mode Ad Hoc car en mode client il suffit de ne pas l'appeler pour utiliser la détection automatique. Cette propriété peut également être définie avec le paramètre « W_STYPE » du fichier main.cfg.

- **char *type** : chaîne de caractères contenant le type de cryptage. Les valeurs possibles sont « » (une chaîne de caractères vide utilise la détection automatique), OPEN (pas de cryptage), WEP, WPA, WPAAES, WPA2AES, WPA2TKIP ou WPA2.

Exemple :

```
WSockSetSecType ( "" ) ;
```

```
void WSockSetKey (char *type, char *key);
```

Cette fonction permet de définir le type de cryptage et la clé de sécurité. C'est l'équivalent des fonctions *WSockSetSecKey* et *WSockSetSecType*.

- **char *type** : chaîne de caractères contenant le type de cryptage. Les valeurs possibles sont « » (une chaîne de caractères vide utilise la détection automatique), OPEN (pas de cryptage), WEP, WPA, WPAAES, WPA2AES, WPA2TKIP ou WPA2.
- **char *key** : chaîne de caractères contenant la clé de sécurité

Exemple :

```
WSockSetKey ( "", "123456 " ) ;
```

10. Connexion à un serveur

Il est très facile d'établir une connexion avec un MicroSERVER afin de remonter des données. Les données à remonter doivent être configurées dans les onglets « Historisation » et « Serveur distant » dans l'onglet « Propriétés de la variable » dans l'éditeur de variable. Cette communication peut se faire par un port série s'il y a un modem GPRS derrière.

10.1 Réglage

La fenêtre de paramétrage est accessible dans MicroLADDER dans l'onglet « Programme » de la barre de menu de la fenêtre principale puis dans « Equipement » puis « Equipement de base ».

Capture d'écran de la fenêtre « Propriétés de l'équipement »

Les paramètres sont identiques à l'utilisation de la fonction *LogSetRemoteDevice* décrite dans la section [10.2 Réglage par programmation](#).

Par contre il est à noter que le champs « Taille de la mémoire (en octets) » doit être spécifié pour les automates ne disposant pas de RAM sauvegardée. On doit y rentrer la taille de la RAM allouée à la mémorisation des historiques. Pour les automates disposant d'une RAM sauvegardée, toute la RAM est utilisée.

Il n'est pas nécessaire de préciser le port du serveur si c'est le port par défaut (13214) qui est utilisé.

10.2 Réglage par programmation

void LogSetRemoteDevice (long index, char *type, char *label, unsigned char comPort, unsigned short speed, unsigned short format, char *host, unsigned short sockPort, unsigned short freq, unsigned long timeout, char *password);

Cette fonction est utilisée si il y a des paramètres de connexion qui évoluent pendant l'exécution du programme. Cette fonction est active sur front. Elle n'est appelée qu'une seule fois ou si les paramètres de communication changent.

- **long index** : entier de type long indiquant l'index du dispositif qui veut se connecter au serveur.
- **char *type** : chaîne de caractères contenant le type de dispositif. Cette information est uniquement utilisée pour réaliser l'affichage sur le serveur.
- **char *label** : chaîne de caractères contenant l'étiquette ou l'emplacement. Cette information est uniquement utilisée pour réaliser l'affichage sur le serveur.
- **unsigned char comPort** : identification du port de communication utilisé pour la communication.

- **unsigned short speed** : indication de la vitesse de communication. Pour une communication en Ethernet ou Wi-Fi, il est nécessaire de mettre COM_SPEED_NONE.
- **unsigned short format** : format de communication. Pour une communication en Ethernet ou en Wi-Fi, il est nécessaire de mettre COM_FORMAT_NONE.
- **char *host** : chaîne de caractères contenant l'adresse du serveur.
- **unsigned short sockPort** : port du serveur distant. Par défaut, c'est le port 13214.
- **unsigned short freq** : identification de la fréquence radio en cas d'utilisation de lora.
- **unsigned long timeout** : entier de type long non signé contenant le délai d'attente en secondes.
- **char *password** : chaîne de caractères contenant le mot de passe.

Exemple :

```
LogSetRemoteDevice (850, "Gestionnaire", "Castres", COM_PORT_SOCK3,
COM_SPEED_NONE, COM_FORMAT_NONE, "194.117.213.206", 13214,
RF_FREQ_NONE, 60, "mdp");
LogSetRemoteDevice (850, "Gestionnaire", "Castres", COM_PORT_SER2,
COM_SPEED_9600, COM_FORMAT_8N1, "194.117.213.206", 13214, RF_FREQ_NONE, 60, "m
```

Il n'est pas nécessaire de paramétrer le protocole du socket Ethernet ou Wi-Fi car la fonction le fait elle-même.

Pour connaître l'état de la connexion, il faut tester la variable « LogDataPortStep ». Elle est supérieure ou égale à « LOG_DATA_PORT_STEP_WAIT_CMD » (valeur 6) lorsque tout va bien.

Les paramètres de connexion sont sauvegardés. Il n'est pas nécessaire de relancer la fonction *LogSetRemoteDevice* à chaque redémarrage de l'application, mais seulement après un chargement.

11. Gestion des historiques

Sur tous les automates, il est possible de gérer les historiques. S'il n'y a pas de RAM sauvegardée, les données seront perdues en cas de coupure de courant.

Il est possible d'avoir des événements (mémorisation de toutes les dates d'apparition et de disparition des pannes, mémorisation de toutes les dates d'apparition des messages), alarmes (récupération des défauts présents et de la date d'apparition) et des traces (avec dates et valeurs d'une donnée).

Ces informations sont stockées en mémoire. Il est ensuite possible de lire ces données et de les sauvegarder sur la carte SD ou la clé USB. Il est également possible de récupérer ces informations sur un système externe à l'aide d'un protocole spécifique.

11.1 Paramétrage de l'historique

Les paramètres se trouvent dans l'onglet « Historisation » dans la fenêtre « Propriétés de la variable » d'une variable.

11.2 Alarme

On peut créer une alarme pour cette variable en remplissant les champs suivants :

- **Condition de l'alarme** : menu déroulant permettant de choisir l'opérateur logique de la condition de l'alarme (<, <=, >, >=, != et ==). Si on choisit « Aucune » alors il n'y aura pas d'alarmes pour

cette variable.

- **Seuil de l'alarme** : valeur de la condition avec laquelle l'alarme est activée. Une alarme est un état, et un événement se produit et est créé chaque fois qu'un seuil est dépassé.
- **Libellé d'apparition de l'alarme** : texte qui apparaîtra lorsque la condition d'alarme est présente.
- **Libellé de disparition de l'alarme** : texte qui apparaîtra lorsque la condition d'alarme disparaîtra.

11.3 Événement

On peut créer une événement pour cette variable en remplissant les champs suivants :

- **Condition de l'événement** : menu déroulant permettant de choisir l'opérateur logique de la condition de l'événement (<, <=, >, >=, != et ==). Si on choisit « Aucune » alors il n'y aura pas d'événement pour cette variable.
- **Seuil de l'événement** : valeur de la condition avec laquelle un événement sera activé.
- **Libellé de l'événement** : texte qui apparaîtra lorsque la condition de l'événement est présente.

11.4 Courbe

On peut pour cette variable en remplissant les champs suivants :

- **Type d'historisation** : menu déroulant permettant de choisir comment on souhaite archiver les courbes de valeur dans le temps. Il y a trois types d'historisations :
 - « Aucune » : aucune historisation effectuée.
 - « Standard » : prend une valeur avec un intervalle régulier
 - « Moyennée » : prend une valeur moyenne sur une période de temps donnée
- **Délai d'historisation (en secondes)** : délai d'archivage. Il dépend du type de journalisation parmi deux possibles :
 - « Standard » : temps spécifié (1 minute par exemple)
 - « Moyennée » : période de temps spécifiée
- **Seuil d'historisation** : seuil au-delà duquel il enregistrera une valeur.

Pour qu'une donnée de courbe soit mémorisée, il faut que le temps écoulé soit supérieur ou égal au temps paramétré et que le delta de valeur soit supérieur ou égal au seuil paramétré.

Dans le cas d'une valeur non moyennée, à la fin de la période, si le delta de variation dépasse l'hystérésis, il y a enregistrement de cette valeur. Dans le cas d'une valeur moyennée, une moyenne est réalisée pendant la période, à la fin de la période, si le delta de variation dépasse l'hystérésis, il y a enregistrement de cette valeur.

LogVariable *LogFindVariable (char *address);

Cette fonction permet de pointer sur la structure « LogVariable » d'une variable historique. Elle doit être utilisée avant d'effectuer une opération sur une variable historique.

- **LogVariable *** (valeur de retour): variable de type pointeur sur une structure LogVariable.
- **char *address** : chaîne de caractères contenant la variable.

Exemple :

```
LogVariable * var = LogFindVariable ("%MW20 ") ;
```

LogVariable *LogGetVariable (unsigned short index);

Cette fonction fait la même chose que la fonction *LogFindVariable* mais à partir de l'indice de la variable historique. Les index pour utiliser cette fonction doivent être connus, mais les index sont assignés à la compilation et donc inconnus lors de l'écriture du programme.

- **LogVariable *** (valeur de retour) : variable de type pointeur sur une structure LogVariable.
- **unsigned short index** : index de la variable historique.

Exemple :

```
LogVariable * var = LogGetVariable (4) ;
```

12. Informations diverses

12.1 Taille mémoire disponible

Chaque variable déclarée occupe de la place en RAM. Si en plus cette variable est initialisée, elle occupe de la place dans la mémoire flash.

Pour connaître la quantité de mémoire utilisée, il faut regarder le fichier « size.txt » qui est généré lors de la compilation. Toutes les valeurs sont exprimées en octet.

Nom de la colonne Signification

text	Taille du code programme
data	Taille des variables initialisées
bss	Taille des variables non initialisées
dec	Taille mémoire (programme + variables) en décimal
hex	Taille mémoire (programme + variables) en hexadécimal
filename	Nom du fichier

Tableau sur les noms des colonnes d'un fichier size.txt et de leur signification

La somme des colonnes data et bss ne doit pas dépasser la taille de la RAM.

La somme des colonnes text et data ne doit pas dépasser la taille de la mémoire flash sachant que 64 ko sont déjà occupés par le moniteur.

Exemple :

text	data	bss	dec	hex	filename
60380	16	4644	65040	fe10	main.elf

Capture d'écran du contenu d'un fichier size.txt

Lorsque l'automate dispose d'une RAM sauvegardée, les variables automate (%M,%MW, ...) sont stockées dans cette RAM sauvegardée et le compilateur ne contrôle pas la taille. Il peut potentiellement y avoir un débordement de mémoire, mais les RAM sauvegardées des automates sont tout de même

assez grandes.

Pour savoir s'il reste de la place en RAM sauvegardée, une solution consiste à déclarer une variable sauvegardée et à regarder le nombre de points qu'il est possible d'historiser (voir la section [11. Gestion des historiques](#) pour plus de détails).