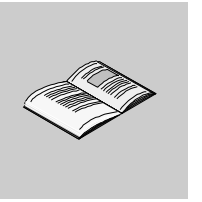
Guide de démarrage pour Unity Pro
Mise en oeuvre d'une application

UNY USE 40010V20F

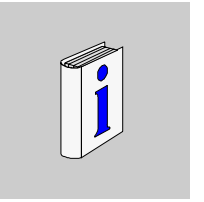
fre Septembre 2004

35008403.00

Table des matières



A propos de ce manuel



Présentation

Objectif du document

Ce manuel décrit la mise en oeuvre d'une application basée sur l'utilisation des différents types de variables, de langages de programmation et d'un écran d'exploitation décrivant le fonctionnement de l'application.

Champ d'application

L'application présentée dans ce manuel a été développée à partir de la version V2.0 du logiciel Unity Pro.

Document à consulter

Titre	Référence
Aide en ligne Unity Pro	
Application disponible dans le CD de documentation	gestion_cuve.XEF

Commentaires utilisateur

Envoyez vos commentaires à l'adresse e-mail TECHCOMM@modicon.com

Description de l'application

1

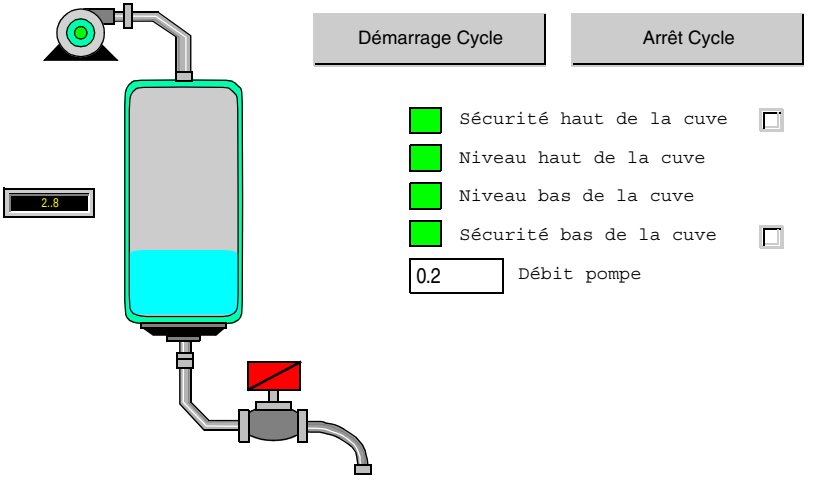
Présentation de l'application

Présentation

L'application décrite dans ce document consiste à gérer le niveau d'un liquide dans une cuve. Le remplissage de la cuve se fait par l'intermédiaire d'une pompe et la vidange est gérée par une vanne.
Les différents niveaux de la cuve sont mesurés par des capteurs disposés sur la cuve.
Le volume de la cuve est donné par un afficheur numérique.
Les moyens de contrôle du fonctionnement de l'application sont basés sur un écran d'exploitation qui doit fournir l'état des différents capteurs, actionneurs et le volume de la cuve.
Suivant l'état du niveau de la cuve et de l'application il faut avertir l'utilisateur par des alarmes et archiver les informations nécessaires à chaque déclenchement.

Illustration

Voici l'écran d'exploitation final de l'application :



Mode de marche

Le mode de marche est le suivant :

- un bouton **Démarrage cycle** permet de lancer les cycles de remplissage,
- lorsque le niveau haut de la cuve est atteint la pompe s'arrête et la vanne s’ouvre. Lorsque le niveau bas de la cuve est atteint, la vanne se ferme et la pompe se met en marche jusqu'a atteindre le niveau haut.
- un bouton **Arrêt cycle** permet d’interrompre les cycles de remplissage. Une action sur ce bouton permet de mettre le système en sécurité. La pompe s’arrête, la vanne s’ouvre jusqu’à atteindre le niveau "Sécurité bas" (cuve vide). La vanne se ferme et le cycle s’arrête.
- la pompe a un débit variable, la valeur de ce débit pourra être accessible par l’écran d’exploitation. Le débit de la vanne est égal à celui de la pompe.
- des sécurités doivent être mises en place :
  - perte du niveau haut de la cuve : un autre niveau dit "Sécurité haut" se déclenche, le système se met en sécurité. Dans ce cas, la pompe s’arrête, la vanne s’ouvre jusqu’à atteindre le niveau "Sécurité bas" (cuve vide). La vanne se ferme et le cycle s’arrête.
  - perte du niveau bas de la cuve : un autre niveau dit "Sécurité bas" se déclenche, le système se met en sécurité. Dans ce cas, la vanne se ferme et le cycle s’arrête.
- pour les deux sécurités, il faut afficher un message de défaut.
- les temps d’ouverture et de fermeture de la vanne sont surveillés, un message de défaut est affiché en cas de dépassement.

Présentation du logiciel Unity Pro

2

Présentation du logiciel Unity Pro

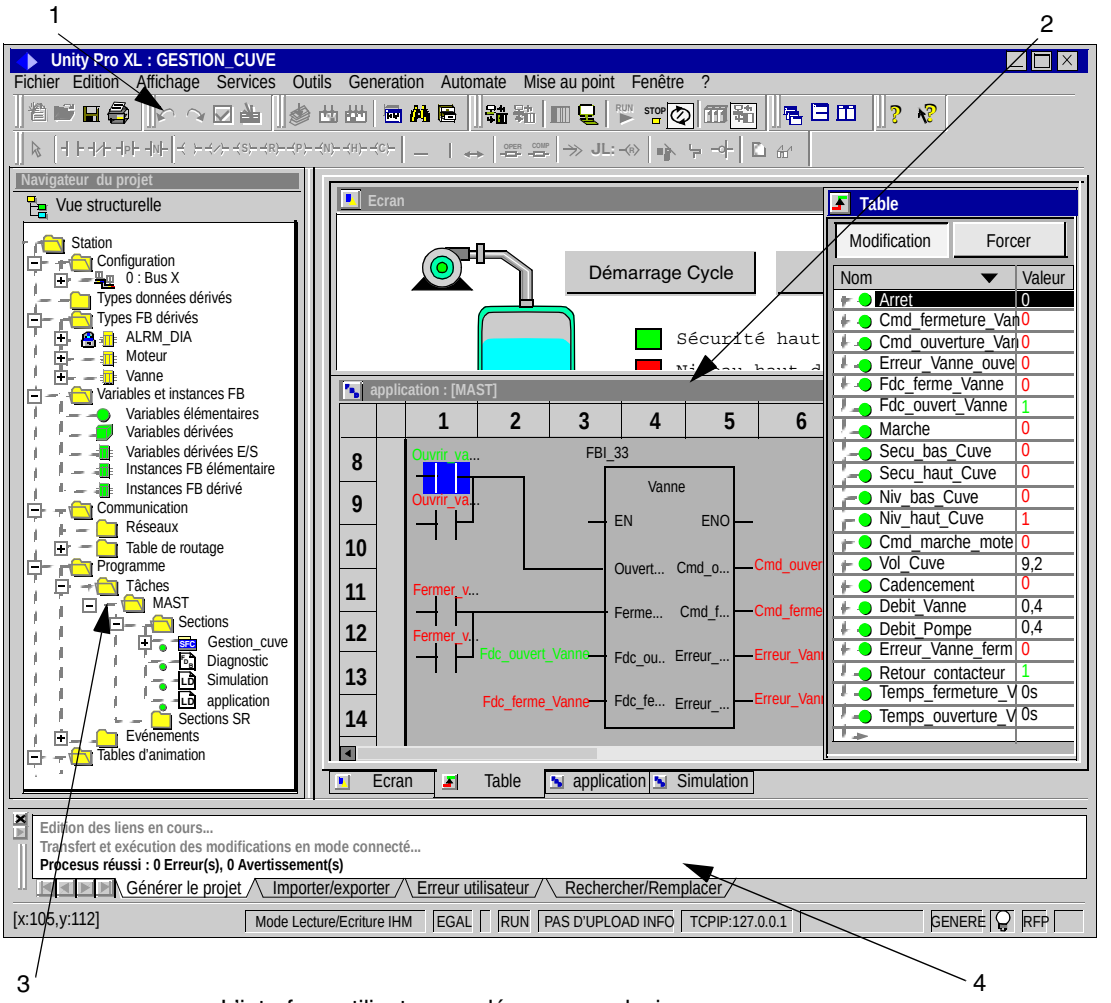
Présentation

Le logiciel Unity Pro est un atelier logiciel destiné à programmer les automates Telemecanique Modicon Premium, Modicon Quantum et Modicon Atrium.
Nous allons décrire brièvement les blocs d'Unity Pro nécessaires au développement de l'application.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro.

Interface utilisateur

L'écran ci-dessous présente l'interface utilisateur d'Unity Pro :



L'interface utilisateur se découpe en plusieurs zones :

Zone	Description
1	Barre d'outils Unity Pro.
2	Fenêtre de l'éditeur (éditeurs de langages, éditeur de données, etc.).
3	Navigateur de projet.
4	Fenêtre d'information (donne des informations sur les erreurs survenues, le suivi des signaux, les fonctions d'importation, etc.).

Navigateur de projet

Le navigateur de projet permet d'accéder aisément aux différents éditeurs (Voir *Les différentes étapes du process dans Unity Pro, p. 18*) utilisés par l'application.

- Configuration (Voir *Configuration, p. 11*),
- Type FB dérivés (Voir *Editeur de DFB, p. 13*),
- Variables et instances FB (Voir *Editeur de données, p. 12*),
- Programmes (Voir *Editeur de programmes, p. 12*) ,
- Diagnostic (Voir *Visualisateur de Diagnostic, p. 13*),
- Ecrans d'exploitation (Voir *Ecrans d'exploitation, p. 14*).

Configuration

L'outil de configuration permet de :

- créer\modifier\sauvegarder les éléments constituant la configuration de la station automate,
- paramétrer les modules métiers constituant la station,
- diagnostiquer les modules configurés dans la station,
- faire le bilan du courant consommé à partir des tensions délivrées par le module alimentation déclaré dans la configuration,
- contrôler le nombre de voies métiers configurées par rapport aux capacités du processeur déclaré dans la configuration,
- faire un bilan sur l'occupation mémoire du processeur.

Note : La configuration peut être effectuée avant, ou après la programmation du projet, cela présente l'avantage de pouvoir créer des projets génériques sans se préoccuper dans un premier temps de la configuration.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Modes opératoires` et `Configuration du projet`).

Editeur de données

L'éditeur de données propose les fonctions suivantes :
- déclaration d'instances de variable,
- définition de types de données dérivés (DDT), accessible directement par Type données dérivés,
- déclaration d'instance de blocs fonctions élémentaires et dérivés (EFB/DFB),
- définition des paramètres de blocs fonctions dérivés (DFB), accessible directement par Type FB dérivés (Voir *Editeur de DFB, p. 13*).

Pour accéder à l'Editeur de données, il suffit de double-cliquer sur Variables et instances FB dans le navigateur de projet.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur Unity, puis Logiciel Unity Pro, puis Modes opératoires et Editeur de données).

Editeur de programmes

L'éditeur de programme permet de développer les différentes tâches de l'automate en utilisant différents types de langage, notamment :
- FBD (langage en blocs fonctionnels),
- LD (langage à contacts),
- SFC (diagramme fonctionnel en séquence), disponible uniquement pour la tâche MAST,
- IL (liste d'instructions),
- ST (littéral structuré).

Pour accéder à l'Editeur de programmes, il suffit de double-cliquer sur Programme dans le navigateur de projet et de choisir une Tâche ou un Evénement.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur Unity, puis Logiciel Unity Pro, puis Modes opératoires et Programmation).

Editeur de DFB

Le logiciel Unity Pro permet de créer des blocs fonction utilisateur DFB, en utilisant les langages d'automatismes. Un DFB est un bloc de programme que vous développez afin de répondre aux spécificités de votre application. Il comprend :

- des paramètres d'entrées/sorties,
- des variables internes publiques ou privées.
- une ou plusieurs sections écrites en langage à contacts (LD), en liste d'instructions (IL), en littéral structuré (ST) ou en langage à blocs fonctionnels (FBD),

Pour accéder à l'`Editeur de DFB`, il suffit de double-cliquer sur `Type FB dérivés` dans le navigateur de projet.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Références langages` et `Bloc fonction utilisateur`).

Visualisateur de Diagnostic

Unity Pro dispose d'un outil de diagnostic du système et des projets.
Dans le cas où des erreurs se produisent, celles-ci s'affichent dans une fenêtre de diagnostic.

Pour accéder à l'`Editeur de DFB`, il suffit de double-cliquer sur `Type FB dérivés` dans le navigateur de projet.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis puis `Modes opératoires` et `Diagnostic`).

Ecrans d'exploitation

Les écrans d'exploitation intégrés sont destinés à faciliter l'exploitation d'un procédé automatisé. Ils utilisent dans le logiciel Unity Pro :

- le navigateur projet qui permet de naviguer dans les écrans et lancer les différents outils (l'éditeur graphique, l'éditeur de variables, l'éditeur de messages, ...),
- l'éditeur graphique qui permet de créer ou modifier les écrans. En mode connecté, il permet également de visualiser les écrans animés et de conduire le procédé,
- la bibliothèque d'objets qui présente des objets constructeur et permet de les insérer dans les écrans. Elle permet aussi de créer ses propres objets et de les insérer dans une famille de la bibliothèque.

Pour accéder aux `Ecrans d'exploitation`, il suffit de faire un clic droit sur `Ecrans d'exploitation` dans le navigateur de projet et de choisir un nouvel écran.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis puis `Modes opératoires` et `Ecrans d'exploitation`).

Simulateur

Le simulateur d'automate permet la simulation d'un projet sans connexion à un véritable automate.
Toutes les tâches du projet (Mast, Fast, AUX et Evénements) sont également disponibles dans le simulateur. La différence par rapport à une véritable API réside dans l'absence de modules E/S et de réseaux de communication.

Pour accéder au `Simulateur`, il suffit de choisir `Mode simulation` dans le menu Automate et de se connecter à l'API.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis puis `Modes opératoires`, puis `Mise au point et réglage` et `Simulateur de l'automate`).

Mise oeuvre de l'application avec Unity Pro

3

Présentation

Objet de ce chapitre

Ce chapitre donne la marche à suivre pour créer l'application décrite. Il donne de manière générale et détaillée les étapes pour créer les différents composants de l'application.

Contenu de ce chapitre

Ce chapitre contient les sous-chapitres suivants :

3.1 Présentation de la solution retenue

Présentation

Objet ce sous-chapitre

Ce sous-chapitre présente la solution retenue pour développer l'application. Il explique les choix technologiques et donne la chronologie de création de l'application.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Les choix technologiques retenus

Présentation

Il existe plusieurs manières d'écrire une application avec Unity Pro. Celle proposée permet de structurer l'application de façon à en faciliter sa réalisation et sa mise au point.

Choix technologiques

Le tableau ci-dessous donne les choix technologiques retenus pour l'application :

Objets	Choix retenus
Utilisation de la pompe	Création d'un bloc fonction utilisateur (DFB) afin de faciliter la gestion de la pompe en terme de saisie de programme et de rapidité de mise au point. Le langage de programmation utilisé pour élaborer ce DFB est un langage graphique à base de blocs fonctionnels (FBD).
Utilisation de la Vanne	Création d'un bloc fonction utilisateur (DFB) afin de faciliter la gestion de la vanne en terme de saisie de programme et de rapidité de mise au point. Le langage de programmation utilisé pour élaborer ce DFB est un langage graphique à base de blocs fonctionnels (FBD).
Ecran supervision	Utilisation des éléments de la bibliothèque et création de nouveaux objets.
Programme principal supervision	Ce programme est développé à partir d'un diagramme fonctionnel en séquence (SFC) aussi connu sous le nom de GRAFCET. Les différentes sections sont réalisées en langage à contacts (LD) et utilisent les différents DFB créés.
Affichage des défauts	Utilisation du DFB ALRM_DIA pour contrôler l'état des variables liées aux défauts.

Note : L'utilisation d'un bloc fonction DFB dans une application vous permet de:
- simplifier la conception et la saisie du programme,
- accroître la lisibilité du programme,
- faciliter la mise au point de l'application,
- diminuer le volume de code généré.

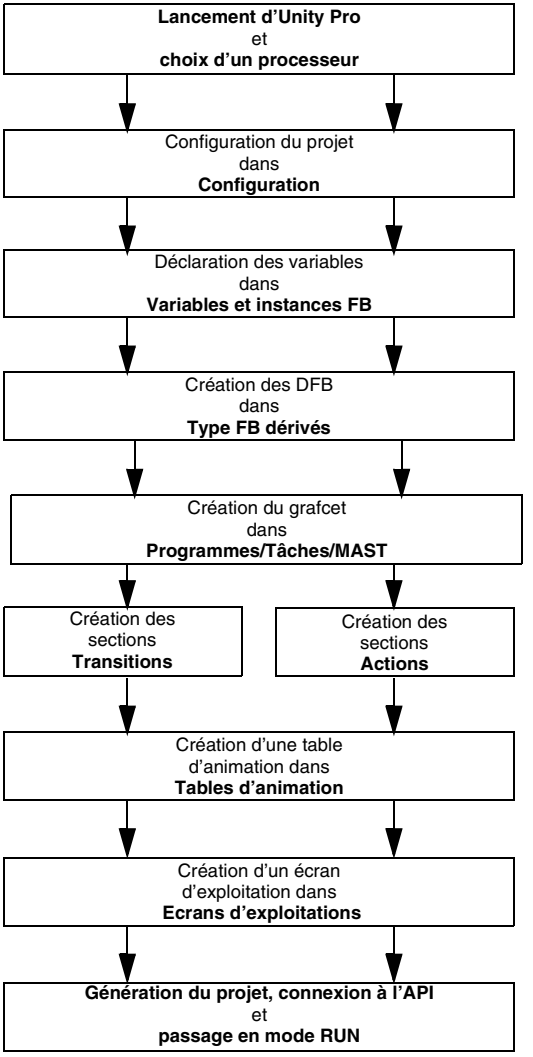
Les différentes étapes du process dans Unity Pro

Présentation

Le logigramme ci-dessous est destiné à donner les différentes étapes à suivre pour créer l'application. Un ordre chronologique doit être respecté afin de définir correctement tous les éléments de l'application.

Description

Description des différentes étapes :



3.2 Développement de l’application

Présentation

Objet ce sous-chapitre

Ce sous-chapitre décrit pas à pas la réalisation de l’application à l’aide du logiciel Unity Pro.

Contenu de ce sous-chapitre

Ce sous-chapitre contient les sujets suivants :

Création du projet

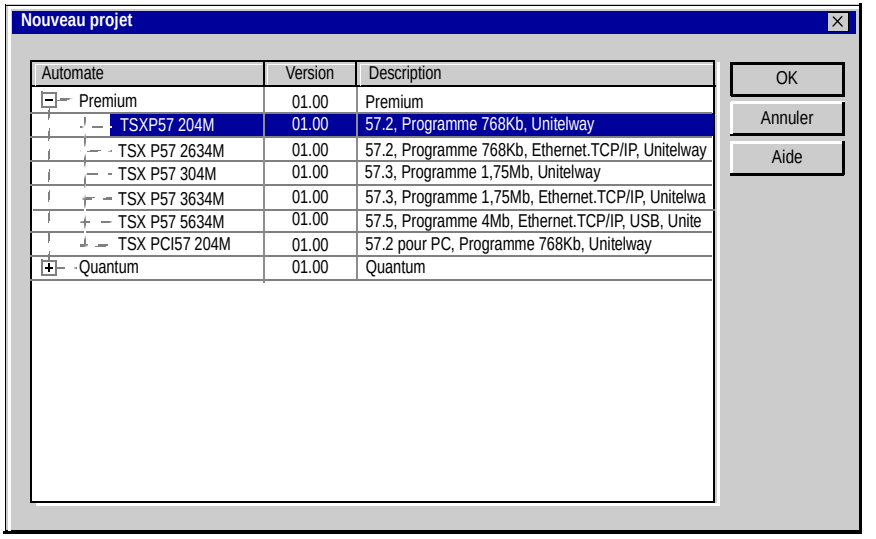
Présentation

Le développement d'une application sous Unity Pro passe par la création d'un projet associé à un automate.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Modes opératoires` et `Configuration du projet`).

Marche à suivre pour créer un projet

Le tableau ci-dessous présente la marche à suivre pour créer le projet sous Unity Pro :

Etape	Action
1	Lancez le logiciel Unity Pro,
2	Cliquez sur `Fichier` puis `Nouveau` puis choisissez un automate, 
3	Insérez un module (Voir *Configuration matérielle de l'application, p. 51*) ou un réseau afin de terminer votre configuration.
4	Validez par `OK`, vous pouvez à présent développer votre application dans Unity Pro.

Déclaration des variables

Présentation

Toutes les variables utilisées dans les différentes sections du programme doivent être déclarées.
Les variables non déclarées ne pourront être utilisées dans le programme.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Modes opératoires` et `Editeur de données`).

Marche à suivre pour déclarer les variables

Le tableau ci-dessous présente la marche à suivre pour déclarer les variables de l'application :

Etape	Action
1	Dans le `Navigateur de projet\Variables et instances FB`, double-cliquez sur `Variables élémentaires`.
2	Dans la fenêtre `Editeur de données` sélectionnez la case dans la colonne `Nom` puis entrez le nom de votre première variable.
3	Ensuite, sélectionnez le Type de cette variable.
4	Lorsque toutes vos variables sont déclarées, vous pouvez fermer la fenêtre.

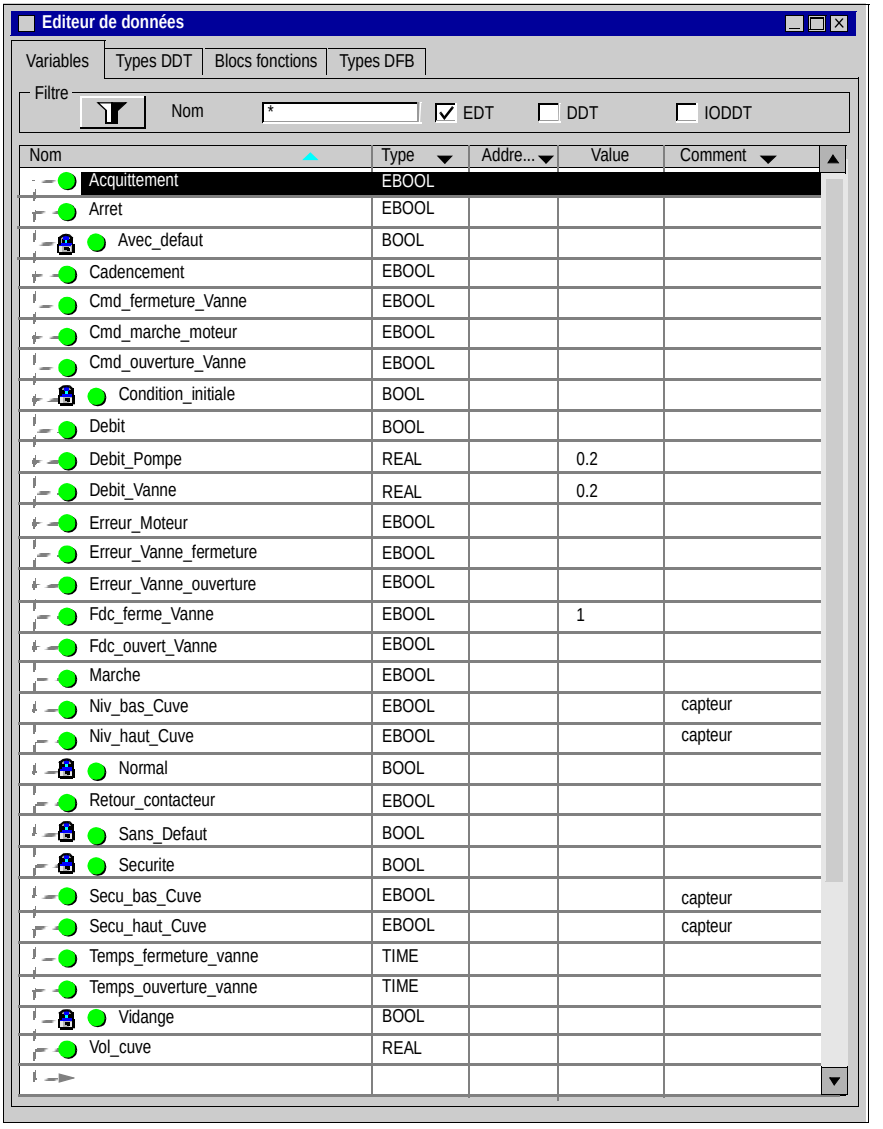
Variables utilisées pour l'application

Le tableau ci-dessous recense le détail des variables utilisées dans l'application :

Variable	Type	Définition
Acquittement	EBOOL	acquittement d'un défaut (Etat 1).
Arret	EBOOL	arrêt de cycle en fin de vidange (Etat 1).
Marche	EBOOL	demande de démarrage des cycles de remplissage (Etat 1).
Cmd_marche_moteur	EBOOL	démarrage des cycles de remplissage (Etat 1).
Erreur_moteur	EBOOL	erreur remontée par le moteur.
Retour_contacteur	EBOOL	erreur remontée par le contacteur en cas d'erreur sur le moteur.
Debit_Pompe	REAL	valeur du débit de la pompe.
Debit	BOOL	variable intermédiaire servant pour la simulation de l'application.
Cadencement	EBOOL	variable utilisée pour le calcul du volume de la cuve (elle est l'image de %S6 dans notre projet). Cette variable est utilisée pour la simulation du projet, il faut donc la supprimer dans le "cas réel".
Cmd_ouverture_vanne	EBOOL	ouverture de la vanne (Etat 1).
Cmd_fermeture_vanne	EBOOL	fermeture de la vanne (Etat 1).
Erreur_vanne_ouverture	EBOOL	erreur remontée par la vanne lors de l'ouverture.
Erreur_vanne_fermeture	EBOOL	erreur remontée par la vanne lors de la fermeture.
Fdc_ouvert_vanne	EBOOL	vanne en position ouverte (Etat 1).
Fdc_ferme_vanne	EBOOL	vanne en position fermée (Etat 1).
Temps_fermeture_vanne	TIME	temps de fermeture de la vanne.
Temps_ouverture_vanne	TIME	temps d'ouverture de la vanne.
Niv_bas_cuve	EBOOL	volume cuve au niveau bas (Etat 1).
Niv_haut_cuve	EBOOL	volume cuve au niveau haut (Etat 1).
Secu_bas_cuve	EBOOL	volume cuve au niveau sécurité bas (Etat 1).
Secu_haut_cuve	EBOOL	volume cuve au niveau sécurité haut (Etat 1).
Vol_cuve	REAL	variable utilisée pour le calcul du volume de la cuve. Cette variable est utilisée pour la simulation du projet, il faut donc la supprimer dans le "cas réel".

Note : Le type EBOOL peut être utilisé pour des modules d'E/S contrairement au type BOOL.

L'écran ci-dessous représente les variables de l'application créées à l'aide de l'éditeur de données :



Nom	Type	Addre...	Value	Comment
Acquittement	EBOOL			
Arret	EBOOL			
Avec_defaut	BOOL			
Cadencement	EBOOL			
Cmd_fermeture_Vanne	EBOOL			
Cmd_marche_moteur	EBOOL			
Cmd_ouverture_Vanne	EBOOL			
Condition_initiale	BOOL			
Debit	BOOL			
Debit_Pompe	REAL		0.2	
Debit_Vanne	REAL		0.2	
Erreur_Moteur	EBOOL			
Erreur_Vanne_fermeture	EBOOL			
Erreur_Vanne_ouverture	EBOOL			
Fdc_ferme_Vanne	EBOOL		1	
Fdc_ouvert_Vanne	EBOOL			
Marche	EBOOL			
Niv_bas_Cuve	EBOOL			capteur
Niv_haut_Cuve	EBOOL			capteur
Normal	BOOL			
Retour_contacteur	EBOOL			
Sans_Defaut	BOOL			
Securite	BOOL			
Secu_bas_Cuve	EBOOL			capteur
Secu_haut_Cuve	EBOOL			capteur
Temps_fermeture_vanne	TIME			
Temps_ouverture_vanne	TIME			
Vidange	BOOL			
Vol_cuve	REAL			

Création et utilisation des DFB

Présentation

Les types DFB sont des blocs fonction programmables par l'utilisateur en langage ST, IL, LD ou FBD. Notre application doit utiliser un DFB moteur et un DFB Vanne. Nous allons utiliser également des DFB existants dans la bibliothèque pour surveiller des variables. Notamment, les variables "sécurité" pour le niveau de la cuve et les variables "erreur" remontées par la vanne. L'état de ces variables sera visible dans `Visualisation du diagnostic`.

Note : les blocs fonction vous permettent de structurer et d'optimiser votre application. Vous pouvez les utiliser dès qu'une séquence de programme est répétée plusieurs fois dans votre application ou pour figer une programmation standard (par exemple, l'algorithme de commande d'un moteur).
Après avoir créé le type de DFB, vous pouvez définir une instance de ce DFB en utilisant l'éditeur de variables ou lors de l'appel de la fonction dans l'éditeur de programme.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Références langages` et `Bloc fonction utilisateur`).

Marche à suivre pour créer un DFB

Le tableau ci-dessous présente la marche à suivre pour créer les DFB de l'application :

Etape	Action
1	Dans le Navigateur de projet, faîtes un clic droit sur Types FB dérivés puis choisissez Ouvrir.
2	Dans la fenêtre Editeur de données sélectionnez la case dans la colonne Nom puis entrez le nom de votre DFB et validez par Entrée. Le nom de votre DFB apparaît avec le signe "Travaux" (DFB non analysé).
3	Ouvrez la structure de votre DFB (voir la figure ci-dessous) puis ajoutez les entrées, sorties et les autres variables propres à votre DFB.
4	Lorsque vos variables du DFB sont déclarées, analysez votre DFB (le signe "Travaux" doit disparaitre). Pour analyser votre DFB, sélectionnez le DFB et cliquez dans le menu Génération puis sur Analyser. Vous venez de créer les variables de votre DFB, il faut maintenant créer la section associée.
5	Dans le Navigateur de projet, double-cliquez sur Types FB dérivés puis sur votre DFB.Sous le nom de votre DFB le champ Sections apparaît.
6	Faîtes un clic droit sur Sections puis choisissez Nouvelle section .
7	Donnez un nom à votre section puis choisissez le type de langage et validez par OK. Editez votre section en utilisant les variables déclarées dans l'étape 3. Votre DFB peut être maintenant utilisable par le programme (Instance de DFB).

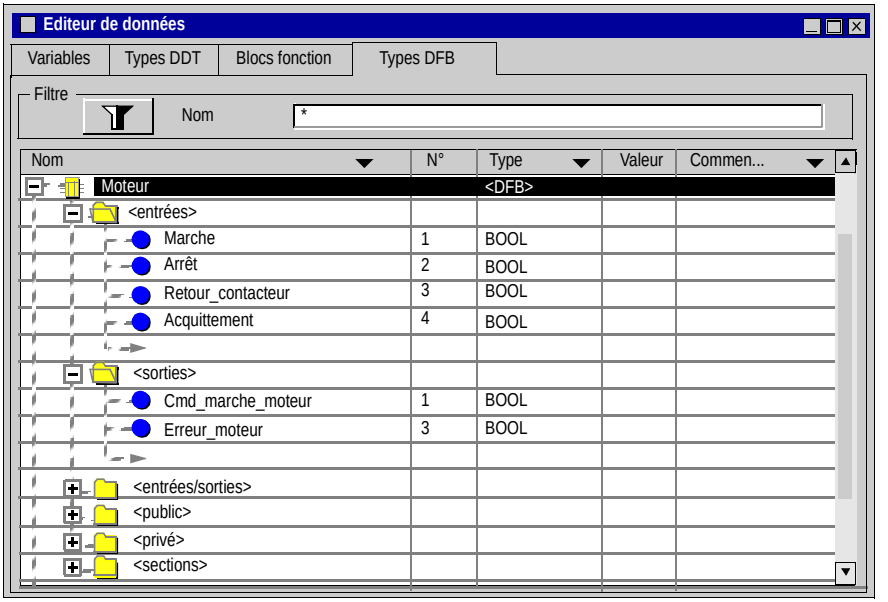
Variables utilisées par le DFB Moteur

Le tableau ci-dessous liste les variables utilisées par le DFB Moteur :

Variable	Type	Definition
Marche	Entrée	commande du démarrage du moteur.
Arret	Entrée	commande de l'arrêt du moteur.
Retour_contacteur	Entrée	retour du contacteur en cas de problème de démarrage du moteur.
Acquittement	Entrée	acquittement de la variable de sortie Erreur_moteur.
Cmd_marche_moteur	Sortie	démarrage du moteur.
Erreur_moteur	Sortie	affichage dans la fenêtre "Visualisation du diagnostic" d'une alarme liée à un problème sur le moteur.

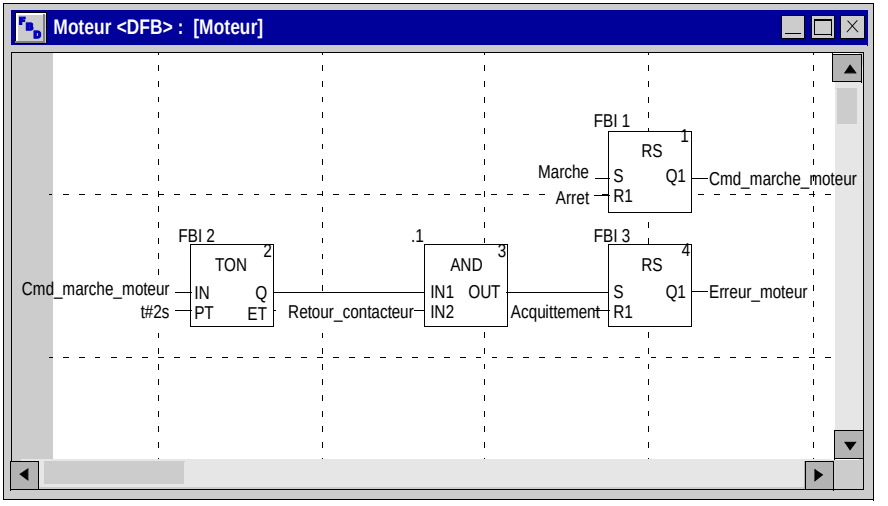
Illustration des variables du DFB Moteur déclarées dans l'éditeur de données

L'écran ci-dessous représente les variables du DFB Moteur utilisées dans cette application pour commander le moteur :



Principe de fonctionnement du DFB Moteur

L'écran ci-dessous représente le programme du DFB Moteur réalisé en langage FBD par l'application pour commander le moteur :



Lorsque Marche = 1et Arrêt = 0, on peut commander le moteur (Cmd_marche_moteur = 1). L'autre partie surveille la variable Retour_contacteur. Si Retour_contacteur n'est pas à "1" avant les deux secondes décompter par le compteur TON, la sortie Erreur_moteur passe à "1".

Note : Pour plus d'information sur la création de la section, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Modes opératoires` et `Programmation` et choisissez le langage désiré).

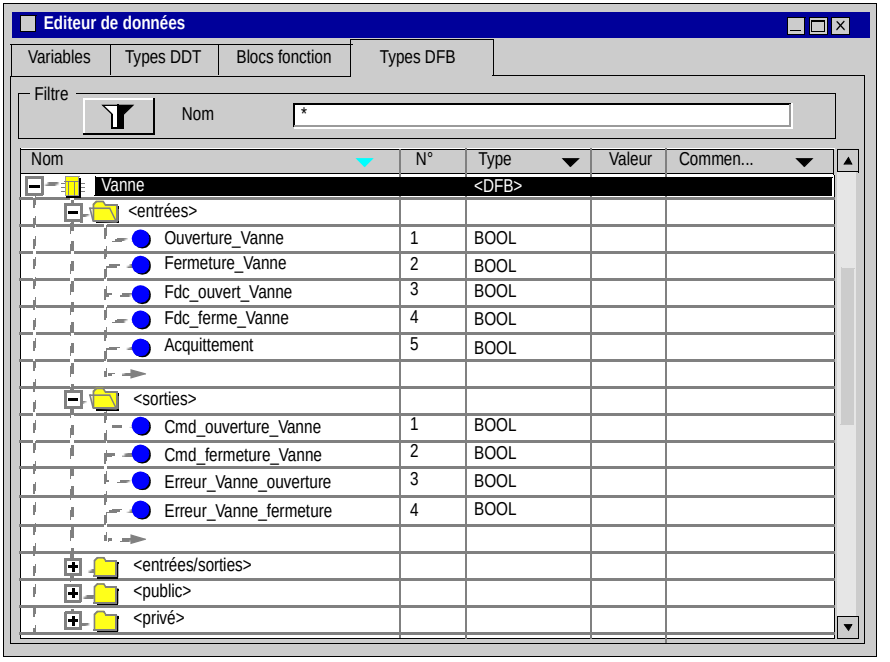
Variables utilisées par le DFB Vanne

Le tableau ci-dessous liste les variables utilisées par le DFB Vanne :

Variable	Type	Definition
Ouverture_vanne	Entrée	commande de l'ouverture de la vanne.
Fermeture_vanne	Entrée	commande de la fermeture de la vanne.
Fdc_ouvert_vanne	Entrée	état du fin de course de la vanne.
Fdc_ferme_vanne	Entrée	état du fin de course de la vanne.
Acquittement	Entrée	acquittement des variables Erreur_fermeture_vanne ou Erreu_ouverture_vanne.
Cmd_ouverture_vanne	Sortie	ouverture de la vanne.
Cmd_fermeture_vanne	Sortie	fermeture de la vanne.
Erreur_vanne_ouverture	Sortie	affichage dans la fenêtre "Visualisation du diagnostic" d'une alarme liée à un problème sur l'ouverture de la vanne.
Erreur_vanne_fermeture	Sortie	affichage dans la fenêtre "Visualisation du diagnostic" d'une alarme liée à un problème sur la fermeture de la vanne.

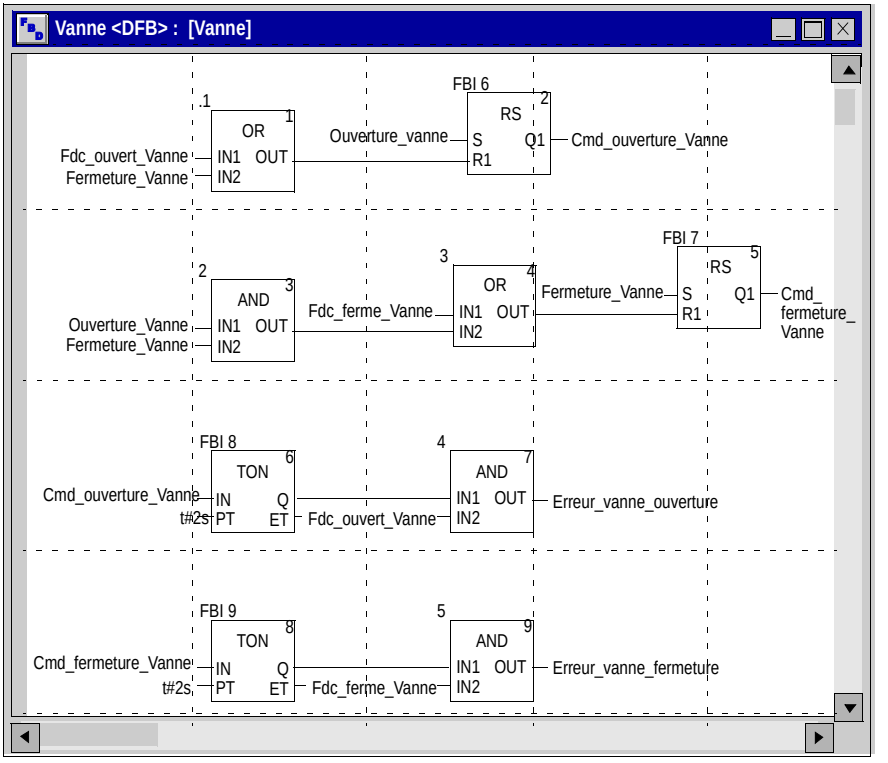
Illustration des variables du DFB Vanne déclarées dans l'éditeur de données

L'écran ci-dessous représente les variables du DFB Vanne utilisées dans cette application pour commander la vanne :



Principe de fonctionnement du DFB Vanne

L'écran ci-dessous représente le DFB Vanne réalisé en langage FBD :



Ce DFB autorise la commande de l'ouverture de la vanne (Cmd_ouverture_vanne) lorsque les entrées Fermeture_vanne et Fdc_ouvert_vanne sont à "0". Le principe est le même pour la fermeture avec une sécurité supplémentaires si on demande la fermeture et l'ouverture de la vanne en même temps (priorité sur l'ouverture).
Afin de surveiller le temps d'ouverture et de fermeture on utilise le temporisateur TON pour retarder le déclenchement d'un défaut. Dés que l'ouverture de la vanne est validée (Cmd_ouverture_vanne = 1) le temporisateur se déclenche. Si dans les deux secondes Fdc_ouvert_vanne n'est pas à "1" la variable de sortie Erreur_vanne_ouverture monte à "1". Dans ce cas un message sera affiché (Voir *Viewer de diagnostic, p. 53*).

Note : Le temps PT doit être réglé en fonction de votre matériel.

Note : Pour plus d'information sur la création de la section, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Modes opératoires` et `Programmation` et choisissez le langage désiré).

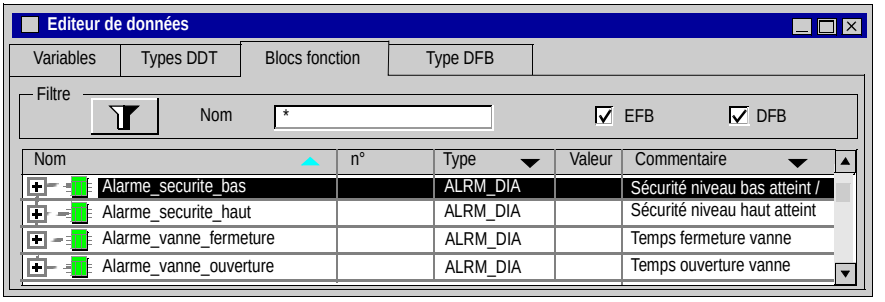
Marche à suivre pour personnaliser un DFB existant à partir d'un DFB de la bibliothèque

Le tableau ci-dessous présente la marche à suivre pour utiliser les DFB ALRM_DIA de la bibliothèque :

Etape	Action
1	Dans le `Navigateur de projet`, double-cliquez sur `Variables élémentaires`, puis choisissez l'onglet `Blocs Fonction`.
2	Dans la fenêtre `Editeur de données` sélectionnez la cellule dans la colonne `Nom` puis entrez le nom de votre Bloc fonction puis validez par `Entrée`.
3	La fenêtre de sélection de type FB apparaît, dans `Bibliothèques/ Familles` choisissez `Bibliothèques` puis `Diagnostic` et cliquez sur `ALRM_DIA` puis validez par `Entrée`.
4	Dans la fenêtre `Editeur de données`, ajoutez des commentaires dans le champ `Commentaire` afin de les visualiser dans `Viewer de diagnostic`. Votre Bloc fonction peut être maintenant utilisable par le programme (Instance de DFB).

Illustration des blocs fonction utilisés par l'application

L'écran ci-dessous représente les différents Blocs fonction ALRM_DIA utilisés dans l'application pour afficher des informations dans la fenêtre du `Viewer de diagnostic` :



Création du programme en SFC pour la gestion de la cuve

Présentation

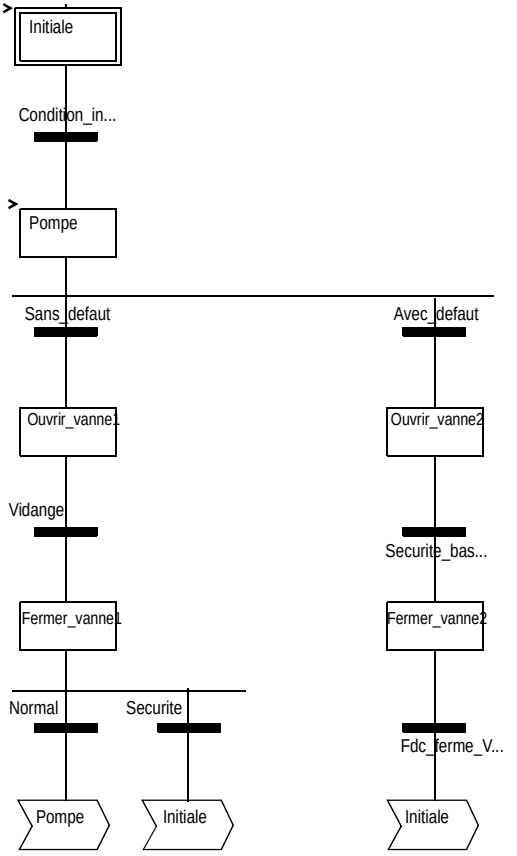
Le programme principal est écrit en SFC (Grafcet). Les différentes sections des étapes et transitions du grafcet sont écrites en LD. Ce programme est déclaré dans une tâche MAST, il sera dépendant de l'état d'une variable booléenne.
Le principal avantage du langage SFC est de pouvoir suivre en temps réel l'exécution de l'application grâce à son animation graphique.
Plusieurs sections sont déclarées dans la tâche MAST :

- la section **Gestion_cuve** (Voir *Illustration de la section Gestion_cuve, p. 33*) écrite en SFC décrivant le mode opératoire,
- la section **Application** (Voir *Création du programme en LD pour l'exécution de l'application, p. 36*) écrite en LD exécutant le démarrage de la pompe en utilisant le DFB moteur, la fermeture et l'ouverture de la vanne.
- la section **Simulation** (Voir *Création du programme en LD pour la simulation de l'application, p. 38*) écrite en LD simulant l'application, cette section est à supprimer dans le cas d'une connexion à un automate.
- la section **Diagnostic** (Voir *Création du programme en FBD pour le diagnostic de l'application, p. 41*) écrite en FBD pour remonter les erreurs de l'application au visualisateur de diagnostic.

Note : L'animation des sections de type LD, SFC et FBD utilisées dans l'application nécessite d'être en mode connecté (Voir *Mise en route de l'application, p. 49*), automate en RUN.

Illustration de la section Gestion_cuve

L'écran ci-dessous représente le grafcet de l'application :



Note : Pour plus d'information sur la création de la section SFC, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Modes opératoires` , puis `Programmation` et `Editeur SFC`).

Description de la section Gestion_cuve

Le tableau ci-dessous décrit les différents étapes et transitions du grafcet Gestion_cuve :

Etape / Transition	Description
Initiale	C'est l'étape initiale.
Condition_initiale	C'est la transition qui va activer le démarrage de la pompe. La transition sera valide lorsque les variables : • Arret_cycle = 0, • Marche_cycle = 1, • Secu_haut_cuve = 0, • Fdc_ferme_vanne = 1
Pompe	C'est l'étape de démarrage de la pompe et de remplissage de la cuve jusqu'au niveau haut. Cette étape activera l'entrée du DFB moteur dans la section Application pour commander le démarrage de la pompe.
Sans_defaut	Cette transition est active lorsque le niveau haut de la cuve est atteinte et que le niveau sécurité haut est à 0.
Ouvrir_vanne1	C'est l'étape de vidange de la cuve et d'ouverture de la vanne. Cette étape activera l'entrée du DFB vanne dans la section Application pour commander l'ouverture de la vanne.
Vidange	Cette transition est active lorsque le niveau bas de la cuve ou le niveau sécurité bas est à 1.
Fermer_vanne1	C'est l'étape de fermeture de la vanne. Cette étape activera l'entrée du DFB vanne dans la section Application pour commander la fermeture de la vanne.
Normal	Cette transition est valide lorsque le niveau bas de la cuve et Fdc_ferme_vanne sont à 1. Dans ce cas on fait un saut vers l'étape S_1_2.
Securite	Cette transition est valide lorsque la sécurité niveau bas de la cuve et Fdc_ferme_vanne sont à 1. Dans ce cas on revient en début de cycle et on attend une initialisation de la variable sécurité et un redémarrage du cycle.
Avec_defaut	Cette transition est active lorsque la Securite niveau haut de la cuve est atteinte ou que le bouton Arret_cycle a été activé (Arret_cycle = 1).
Ouvrir_vanne2	Cette étape est identique à la Ouvrir_vane1.
Securite_bas_cuve	Cette transition est active lorsque la sécurite basse de la cuve est à 1 (après une vidange de la cuve suite à un arrêt du cycle ou à une activation de la sécurite haute de la cuve).
Fermer_vanne2	Cette étape est identique à la Fermer_vanne1.
Fdc_ferme_vanne	Cette transition est valide lorsque le Fdc_ferme_vanne est à 1. Dans ce cas on revient en début de cycle et on attend une initialisation de la variable sécurité et un redémarrage du cycle.

Note : Vous pouvez voir toutes les étapes et actions de votre diagramme SFC en cliquant sur  placé devant le nom de votre section SFC.

Marche à suivre pour créer la section SFC

Le tableau ci-dessous présente la marche à suivre pour créer la section SFC de l'application :

Etape	Action
1	Dans le `Navigateur de projet\Programme\Tâches`, double-cliquez sur `MAST`.
2	Faîtes un clic droit sur `Section` puis choisissez `Nouvelle section`. Donnez un nom à votre section (Gestion_cuve pour la section SFC) puis sélectionnez le langage SFC.
3	Le nom de votre section apparaît, vous pouvez l'éditer en double-cliquant dessus.
4	Les outils d'édition du SFC apparaîssent dans la fenêtre, vous pouvez ainsi créer votre grafcet. Exemple pour créer une étape avec une transition : ● pour créer une étape, cliquez sur  puis placez-la dans l'éditeur, ● pour créer une transition, cliquez sur  puis placez-la dans l'éditeur (généralement sous l'étape qui la précéde).

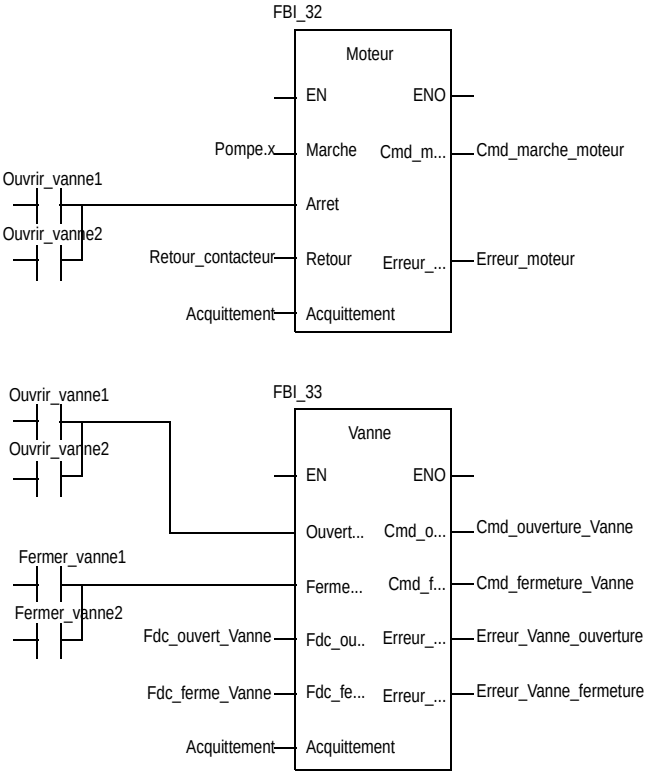
Création du programme en LD pour l'exécution de l'application

Présentation

Cette section commande la pompe et la vanne en utilisant les DFB créés (Voir *Création et utilisation des DFB, p. 24*).

Illustration de la section Application

La section ci-dessous fait partie de la tâche MAST. Elle n'a pas de condition, donc elle est exécutée en permanence :



Description de la section Application

- lorsque l'étape Pompe est active, l'entrée Marche du DFB moteur est à 1, si l'entrée Arret du DFB moteur est à 0, la sortie Cmd_marche_moteur passe à "1" et la pompe est alimentée.
- même principe pour les étapes Ouvrir_vanne1 et Ouvrir_vanne2 et pour le reste de la section.

Marche à suivre pour créer la section LD

Le tableau ci-dessous décrit la création d'une partie de la section **Application** :

Etape	Action
1	Dans le `Navigateur de projet\Programme\Tâches`, double-cliquez sur `MAST`.
2	Faîtes un clic droit sur `Section` puis choisissez `Nouvelle section`. Donnez le nom Application à cette section puis choisissez le langage LD. La fenêtre d'édition s'ouvre.
3	Pour créer le contact Ouvrir_vanne1.x, cliquez sur  puis placez la dans l'éditeur. Double-cliquez sur ce contact puis écrivez le nom de l'étape avec un ".x" à la fin (signifiant une étape d'une section SFC) et validez par `OK`.
4	Pour utiliser le DFB Moteur il faut l'instancier. Faites un clic droit dans l'éditeur puis cliquez sur `Sélection de données` et sur  . Cliquez sur l'onglet `Fonction et types de bloc fonction` et sélectionnez votre DFB puis validez par `OK` et placez votre DFB . Pour relier le contact Ouvrir_vanne1.x à l'entrée Arret du DFB, alignez horizontalement le contact et l'entrée, enfin cliquez sur  et placez-le entre le contact et l'entrée.

Note : Pour plus d'information sur la création de la section LD, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Modes opératoires` , puis `Programmation` et `Editeur LD`).

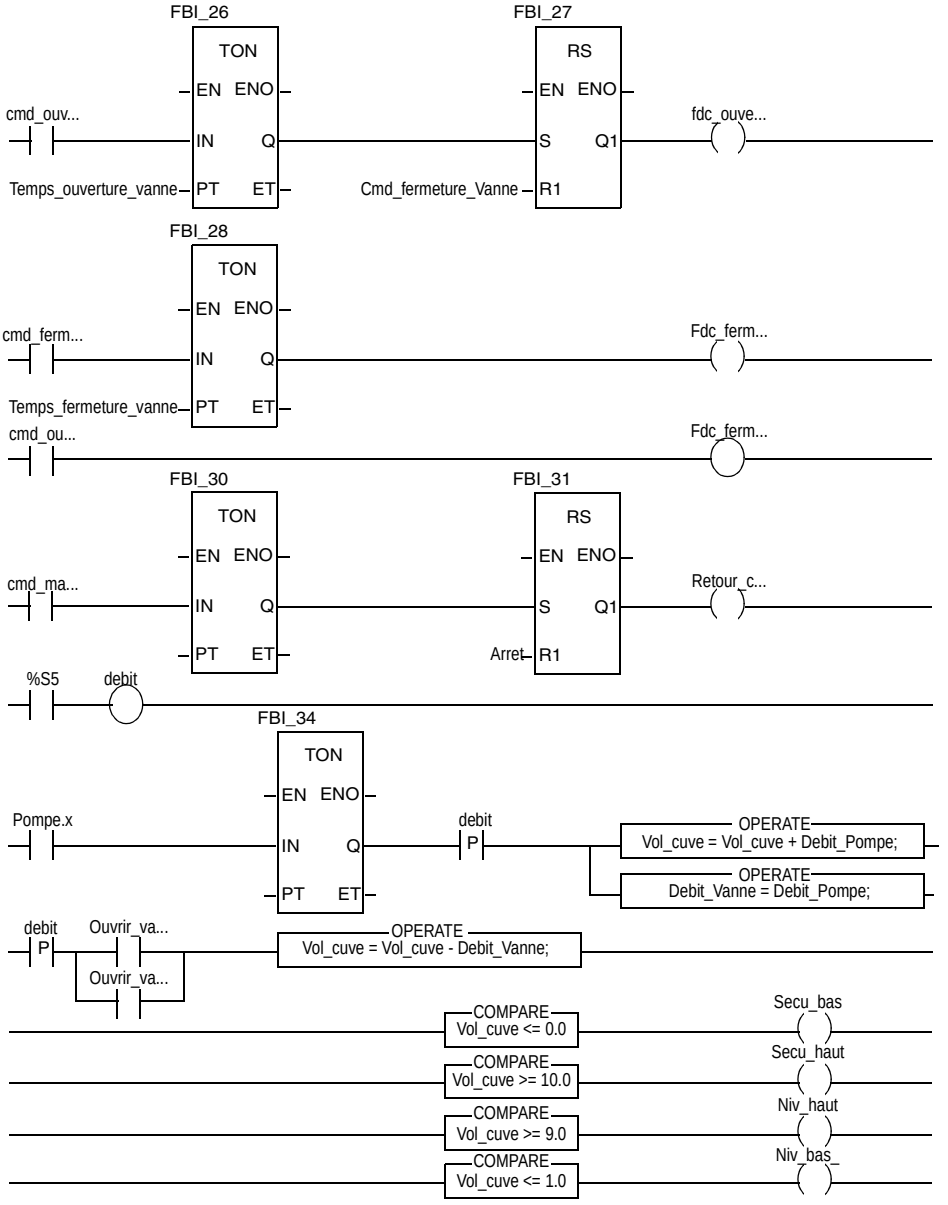
Création du programme en LD pour la simulation de l’application

Présentation

Cette section sert uniquement pour la simulation de l’application. Elle n’est donc pas à utiliser dans le cas d’une connexion à un automate.

Illustration de la section Simulation

La section ci-dessous fait partie de la tâche MAST. Elle n'a pas de condition, elle est donc exécutée en permanence :



Note : Pour plus d'information sur la création de la section LD, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Modes opératoires` , puis `Programmation` et `Editeur LD`).

Description de la section Simulation

- la premiere ligne sert à simuler la valeur de la variable Fdc_ouvert_vanne. Si on commande l'ouverture (Cmd_ouverture_vanne = 1) on déclenche un temporisateur TON. Lorsque le temps PT est atteint la sortie du TON passe à "1" et fait monter à "1" la sortie Fdc_ouvert_vanne sauf si on commande en même temps la fermeture de la vanne.
- même principe pour les sortie Fdc_ferme_vanne et Retour_contacteur.
- la dernière partie de la section sert à la simulation du niveau de la cuve et au déclenchement des différents niveaux. Pour cela, on utilise des blocs OPERATE et COMPARE disponible dans la bibliothèque.

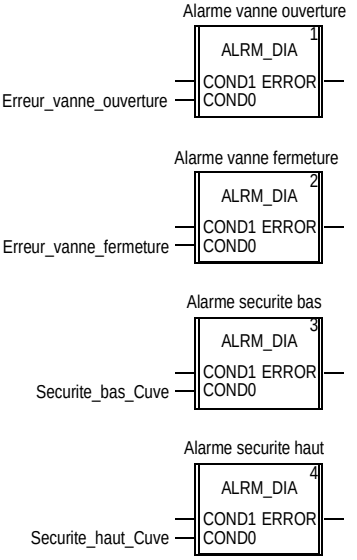
Création du programme en FBD pour le diagnostic de l'application

Présentation

Cette section est utilisée pour déclarer les variables qui seront remontées dans le viewer de diagnostic en cas d'erreur.

Illustration de la section Diagnostic

L'écran ci-dessous représente la section FBD utilisant les Blocs fonction (Voir *Illustration des blocs fonction utilisés par l'application, p. 31*) Alarme_securite_bas, Alarme_securite_haut et erreur_vanne :



Description de la section Diagnostic

Le principe de cette section est basé sur l'utilisation des blocs fonction ALMR_DIA. Dans tous les blocs, on surveille le changement d'état de la variable d'entrée. Les entrées étant toujours connectées à COND0, le déclenchement de l'affichage dans la fenêtre du Viewer de diagnostic se fera lors d'un passage à 1 de la variable d'entrée.

Marche à suivre pour créer la section FBD

Le tableau ci-dessous décrit le principe de la section **Diagnostic** :

Etape	Action
1	Dans le `Navigateur de projet\Programme\Tâches`, double-cliquez sur `MAST`.
2	Faîtes un clic droit sur `Section` puis choisissez `Nouvelle section`. Donnez le nom Diagnostic à cette section puis choisissez le langage FBD. La fenêtre d'édition s'ouvre.
3	Pour utiliser le bloc fonction de type ALRM_DIA créé il faut l'instancier. Faites un clic droit dans l'éditeur puis cliquez sur `Sélection de données` et sur  . Cliquez sur l'onglet `Blocs fonction` et sélectionnez votre bloc puis validez par `OK` et placez-le dans l'éditeur FBD. Pour affecter une variable à une entrée ou une sortie, double-cliquez dessus, cliquez sur  et dans l'onglet `Variable` choisissez votre variable.

Note : Pour plus d'information sur la création de la section LD, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Modes opératoires` , puis `Programmation` et `Editeur FBD`).

Création de la table d'animation

Présentation

La table d'animation est utilisée pour surveiller des valeurs de variables, modifier et/ou forcer des valeurs. Seules les variables déclarées dans `Variables et instances FB` peuvent être ajoutées dans la table d'animation.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Modes opératoires` , puis `Mise au point et réglage` puis `Visualisation et réglage des variables` et `Tables d'animation`).

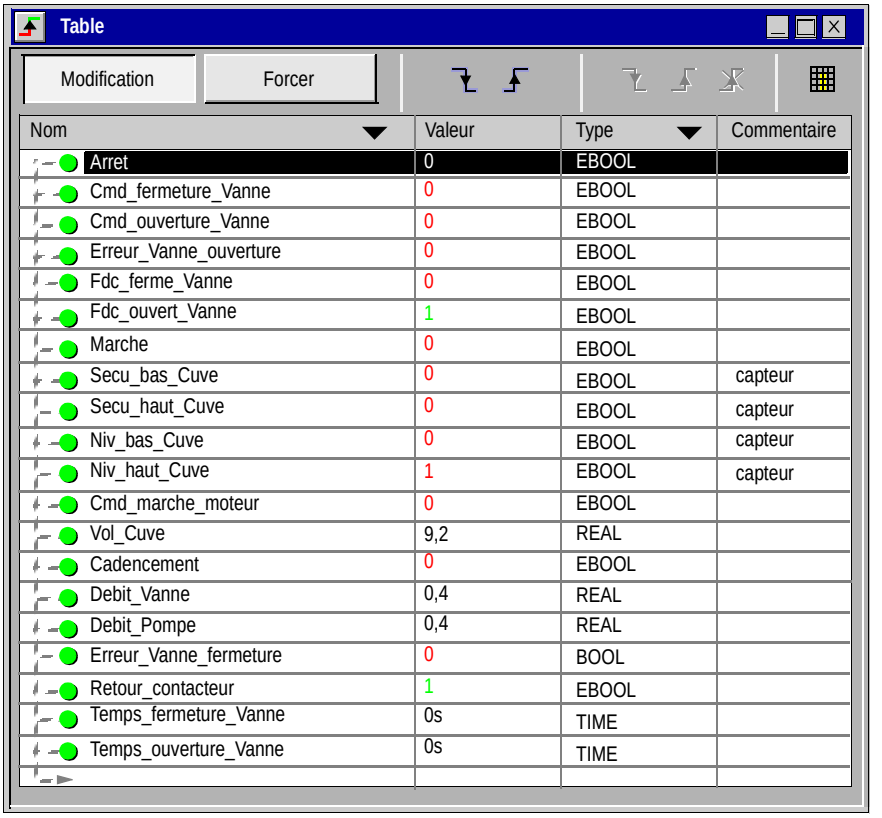
Marche à suivre pour créer la table d'animation

Le tableau ci-dessous présente la marche à suivre pour créer la table d'animation :

Etape	Action
1	Dans le `Navigateur de projet`, faites un clic droit sur `Tables d'animation`, la fenêtre d'édition s'ouvre.
2	Cliquez dans la première cellule de la colonne nom puis sur le bouton et rajoutez les variables de votre choix.

Table d'animation créée pour l'application

L'écran ci-dessous représente la tabe d'animation utilisée par l'application :



Nom	Valeur	Type	Commentaire
Arret	0	EBOOL	
Cmd_fermeture_Vanne	0	EBOOL	
Cmd_ouverture_Vanne	0	EBOOL	
Erreur_Vanne_ouverture	0	EBOOL	
Fdc_ferme_Vanne	0	EBOOL	
Fdc_ouvert_Vanne	1	EBOOL	
Marche	0	EBOOL	
Secu_bas_Cuve	0	EBOOL	capteur
Secu_haut_Cuve	0	EBOOL	capteur
Niv_bas_Cuve	0	EBOOL	capteur
Niv_haut_Cuve	1	EBOOL	capteur
Cmd_marche_moteur	0	EBOOL	
Vol_Cuve	9,2	REAL	
Cadencement	0	EBOOL	
Debit_Vanne	0,4	REAL	
Debit_Pompe	0,4	REAL	
Erreur_Vanne_fermeture	0	BOOL	
Retour_contacteur	1	EBOOL	
Temps_fermeture_Vanne	0s	TIME	
Temps_ouverture_Vanne	0s	TIME	

Note : La table d'animation est dynamique seulement en mode connecté (affichage des valeurs des variables).

Création de l'écran d'exploitation

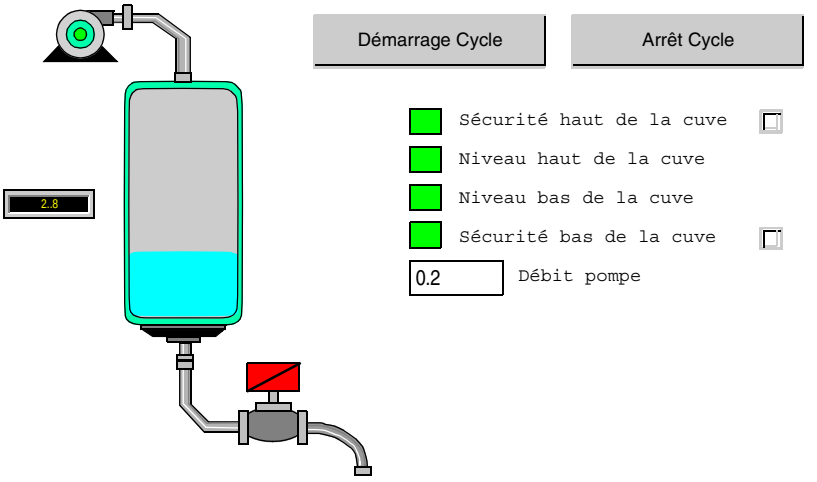
Présentation

L'écran d'exploitation est utilisé pour animer des objets graphiques symbolisant l'application. Ces objets peuvent appartenir à la bibliothèque d'Unity Pro ou ils peuvent être créés à l'aide de l'éditeur graphique.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur , puis sur `Unity` , puis Logiciel `Unity Pro`, puis `Modes opératoires` et `Ecrans d'exploitation`).

Illustration de l'écran d'exploitation

L'illustration ci-dessous représente l'écran d'exploitation de l'application :



Note : Pour animer les objets en mode connecté, il faut cliquer sur le bouton . En cliquant sur ce bouton, vous pouvez valider les écritures.

Marche à suivre pour créer l'écran d'exploitation

Le tableau ci-dessous présente la marche à suivre pour créer le bouton `Démarrage_cycle` :

Etape	Action
1	Dans le `Navigateur de projet`, faites un clic droit sur `Ecrans d'exploitation` et cliquez sur `Nouvel Ecran`. L'éditeur d'écran d'exploitation apparaît.
2	• cliquez sur le bouton  et placez-le dans l'éditeur de l'écran d'exploitation. Double-cliquez sur le bouton et dans l'onglet `Pilotage`, sélectionnez la variable `Marche` en cliquant sur le bouton  et validez par `OK`, puis entrez le nom du bouton dans la zone `Texte`. Le bouton est à présent affecté à la variable `Marche`.

Le tableau ci-dessous présente la marche à suivre pour insérer et animer la cuve :

Etape	Action
1	Dans le `Navigateur de projet`, faites un clic droit sur `Ecrans d'exploitation` et cliquez sur `Nouvel Ecran`. L'éditeur d'écran d'exploitation apparaît.
2	• dans le menu `Outils`, sélectionnez `Bibliothèque des écrans d'exploitation`. La fenêtre s'ouvre, double-cliquez sur `Fluides` puis sur `Cuve`. Sélectionnez la cuve dynamique de l'écran d'exploitation, et faîtes un `Copier` (Ctrl + C) puis `Coller` (Ctrl + V) dans le dessin dans l'éditeur de l'écran d'exploitation (pour revenir sur votre écran, cliquez sur le menu `Fenêtre` puis `Ecran`). • la cuve est maintenant dans votre écran d'exploitation. Il faut maintenant une variable pour animer le niveau. Dans le menu `Outils`, cliquez sur `Fenêtre des variables`. La fenêtre apparaît sur la gauche et dans la colonne `Nom` on trouve le mot %MW0. Pour avoir la partie animée de l'objet graphique (ici la cuve), il faut double-cliquer sur %MW0. Une partie de la cuve est sélectionnée, faîtes un clic droit sur cette partie puis cliquez sur `Caractéristiques`. Sélectionnez l'onglet `Animation` et entrez la variable concernée en cliquant sur le bouton  (à la place de %MW0). Dans notre application se sera Vol_cuve. • il faut définir les minimum et maximum de la cuve. Dans l'onglet `Type d'animation`, cliquez sur `Bargraphe` puis sur le bouton  et rentrez les champs en fonction de la cuve. • validez par `Appliquer` et `OK`.

Le tableau ci-dessous présente la marche à suivre pour insérer et animer la vanne :

Etape	Action
1	Dans le `Navigateur de projet`, faites un clic droit sur `Ecrans d'exploitation` et cliquez sur `Nouvel Ecran`. L'éditeur d'écran d'exploitation apparaît.
2	• dans le menu `Outils`, sélectionnez `Bibliothèque des écrans d'exploitation`. La fenêtre s'ouvre, double-cliquez sur `Actionneurs` puis sur `Vanne`. Sélectionnez une vanne dynamique (de l'écran d'exploitation) et faîtes un `Copier` (Ctrl + C) puis `Coller` (Ctrl + V) dans le dessin dans l'éditeur de l'écran (pour revenir sur votre écran, cliquez sur le menu `Fenêtre` puis `Ecran`). • Sélectionnez la vanne, faîtes un clic droit dessus et cliquez sur Dissocier, sélectionnez le rectangle rouge et déplacez le de manière à voir l'autre rectangle vert dessous. Double-cliquez sur le rectangle vert, cliquez sur l'onglet Animation et ajoutez la variable Cmd_ouverture_vanne. Toujours dans la fenêtre `Propriétés de l'objet`, dans la zone `Condition d'affichage`, sélectionnez `Bit = 1`. Ce paramétrage rends visible le rectangle vert lorsque le bit %M2 = 1 sinon ce rectangle est invisible. • Même procédure pour le rectangle rouge, mais avec une condition d'affichage `Bit = 0`. Si l'animation ne fonctionne pas, mettez en arrière-plan le rectangle se trouvant au premier plan.

Mise en route de l’application

4

Présentation

Objet de ce sous-chapitre

Ce chapitre présente la marche à suivre pour mettre en route l’application. Il décrit les différents type d’exécution de l’application.

Contenu de ce chapitre

Ce chapitre contient les sujets suivants :

Exécution de l'application en mode simulation

Présentation

Il est possible de vous connecter au simulateur d'API qui permet de tester une application sans raccordement à l'automate et autres matériels.

Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis puis `Modes opératoires`, puis `Mise au point et réglage` et `Simulateur de l'automate`).

Exécution de l'application

Le tableau ci-dessous présente la marche à suivre pour lancer l'application en mode simulation :

Etape	Action
1	Dans le menu `Automate`, cliquez sur `Mode Simulation`,
2	Dans le menu `Génération`, cliquez sur `Regénérer tout le projet`. Votre projet est généré et prêt à être transféré sur le simulateur. Lorsque vous générez le projet, vous apercevez la fenêtre de résultats. En cas de d'erreur dans le programme, Unity Pro indique l'emplacement en double-cliquant sur la phrase surligné.
3	Dans le menu `Automate`, cliquez sur `Connexion`. Vous êtes maintenant connecté au simulateur.
4	Dans le menu `Automate`, cliquez sur `Transférer le projet vers l'automate`, la fenêtre `Transfert du projet vers l'automate` s'ouvre, cliquez sur `Transférer`. L'application est transférée dans le simulateur d'automate.
5	Dans le menu `Automate`, cliquez sur `Exécuter`, la fenêtre `Exécuter` s'ouvre, cliquez sur `OK`. L'application est en cours d'exécution (mode RUN) dans le simulateur.

Exécution de l'application en mode standard

Présentation

Le mode standard impose l'utilisation d'un automate et de modules d'E/S TOR et ANA pour affecter les sorties aux différents capteurs et actionneurs.
Les variables utilisées dans le mode simulation doivent être modifiées. En effet, en mode standard , les variables sont obligatoirement localisées afin d'être associées à des E/S physiques.

Note : Pour plus d'information pour l'adressage, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis puis `Références langages`, puis `Description des données` et `Instances de données`).

Configuration matérielle de l'application

Le tableau ci-dessous présente la marche à suivre pour configurer l'application :

Etape	Action
1	Dans la fenêtre `Navigateur de projet` double-cliquez sur Configuration puis sur `0:Bus X` et sur `0:TSX RKY ●●●` (0 étant le numéro du rack).
2	Dans le fenêtre `Bus X` choisissez un emplacement, par exemple 3 et double-cliquez dessus.
3	Insérez un module d'entrées TOR, par exemple `TSX DEY 16A5`.
4	Validez par `OK`. Ce module d'entrée est utilisé pour câbler les entrées de type EBOOL de l'application.

Affectation des variables au module d'entrées

Le tableau ci-dessous présente la marche à suivre pour l'adressage direct des variables

Etape	Action
1	Dans la fenêtre `Navigateur de projet` et dans `Variables et instances FB` double-cliquez sur `Variables élémentaires`.
2	Dans la colonne `Adresse` entrez l'adresse Rack\Module\Voie\Donnée associée a la variable. **Exemple :** Dans le module TSX DEY 16A5, il y a 2 voies, la voie 0 et la voie 8. La voie 0 comprend les entrées 0 à 7 et la voie 8 les entrées 8 à 15. Si on câble sur l'entrée 0 du module la sortie du fin de course fermeture vanne, on obtient dans la colonne adresse de l'éditeur de la variable Fdc_fermeture _vanne l'adresse %I0.3.0.0 Illustration : 
3	Procédez de la même manière pour toutes les variables localisées.

Exécution de l'application

Le tableau ci-dessous présente la marche à suivre pour lancer l'application en Mode Standard :

Etape	Action
1	Dans le menu `Automate`, cliquez sur `Mode Standard`,
2	Dans le menu `Génération`, cliquez sur `Regénérer tout le projet`. Votre projet est généré et prêt à être transféré sur l'automate. Lorsque vous générez le projet, vous apercevez la fenêtre de résultats. En cas de d'erreur dans le programme, Unity Pro indique l'emplacement en cliquant sur la phrase surligné.
3	Dans le menu `Automate`, cliquez sur `Connexion`. Vous êtes maintenant connecté à l'automate.
4	Dans le menu `Automate`, cliquez sur `Transférer le projet vers l'automate`, la fenêtre `Transfert du projet vers l'automate` s'ouvre, cliquez sur `Transférer`. L'application est transférée dans l'automate.
5	Dans le menu `Automate`, cliquez sur `Exécuter`, la fenêtre `Exécuter` s'ouvre, cliquez sur `OK`. L'application est en cours d'exécution (mode RUN) dans l'automate.

Viewer de diagnostic

Présentation

La visualisation du diagnostic permet de surveiller des variables lorsque celles-ci sont associées à des blocs fonction de type diagnostic (ALMR_DIA par exemple).

Note : Pour plus d'information sur la déclaration de ces variables pour l'utilisation du diagnostic, se référer à la partie DFB (Voir *Marche à suivre pour personnaliser un DFB existant à partir d'un DFB de la bibliothèque, p. 31*).

Création du diagnostic

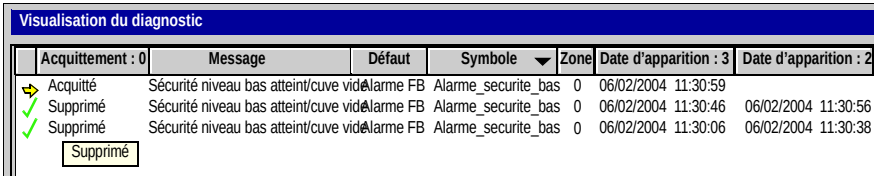
Le tableau ci-dessous présente la marche à suivre pour afficher la fenêtre de diagnostic

Etape	Action
1	Dans le menu `Outils`, cliquez sur `Viewer de diagnostic`, la fenêtre s'affiche à l'écran.
2	Dés que les variables `Secu_bas_cuve` ou `Secu_haut_cuve` ou `Erreur_ouverture_vanne` ou `Erreur_fermeture_vanne` passeront de 0 à 1, un message apparaîtra dans le diagnostic.

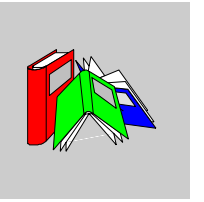
Note : Pour plus d'information, veuillez consulter l'aide en ligne d'Unity Pro (cliquez sur  , puis sur `Unity` , puis Logiciel `Unity Pro`, puis puis `Modes opératoires` et `Diagnostic`).

Illustration du viewer de diagnostic

L'illustration ci-dessous représente un exemple d'affichage lorsque la variable Secu_niveau_bas passe de 0 à 1:



Glossaire



!

%I Selon le standard IEC, %I indique un objet langage de type entrée TOR.

%M Selon le standard IEC, %M indique un objet langage de type bit mémoire.

%MW Selon le standard IEC, %MW indique un objet langage de type mot mémoire.

%Q Selon le standard IEC, %Q indique un objet langage de type sortie TOR.

B

BIT Unité binaire pour une quantité d'informations pouvant représenter deux valeurs distinctes (ou états distincts) : 0 ou 1.

BOOL BOOL est l'abréviation du type booléen. Il s'agit de l'élément de données de base en informatique. Une variable de type BOOL possède l'une ou l'autre des valeurs suivantes : 0 (FALSE) ou 1 (TRUE).
Un bit extrait de mot est de type BOOL, par exemple : %MW10.4.

BYTE Lorsque 8 bits sont regroupés, on parle alors de BYTE (octet). La saisie d'un BYTE s'effectue soit en mode binaire, soit en base 8.
Le type BYTE est codé sur un format 8 bits qui, au format hexadécimal, va de 16#00 à 16#FF.

D

DFB

DFB est l'abréviation de Derived Function Block (bloc fonction dérivé).
Les types DFB sont des blocs fonction programmables par l'utilisateur en langage ST, IL, LD ou FBD.
L'utilisation de ces types DFB dans une application permet :

- de simplifier la conception et la saisie du programme ;
- d'accroître la lisibilité du programme ;
- de faciliter sa mise au point ;
- de diminuer le volume du code généré.

E

EBOOL

`EBOOL` est l'abréviation du type Extended BOOLean (booléen étendu). Il permet de gérer les fronts montants ou descendants ainsi que le forçage.
Une variable de type `EBOOL` occupe un octet en mémoire.

Ecran d'exploitation

Editeur intégré à Unity Pro et utilisé pour faciliter le fonctionnement d'un processus automatisé. L'utilisateur contrôle et surveille l'opération d'installation et, en cas de problème, peut intervenir rapidement et simplement.

EFB

Abréviation de Elementary Function Block (bloc fonction élémentaire).
Il s'agit d'un bloc, utilisé dans un programme, qui réalise une fonction logicielle prédéfinie.
Les EFB possèdent des états et des paramètres internes. Même si les entrées sont identiques, les valeurs des sorties peuvent différer. Par exemple, un compteur possède une sortie qui indique que la valeur de présélection est atteinte. Cette sortie est paramétrée sur 1 lorsque la valeur en cours est égale à la valeur de présélection.

F

FBD

Abréviation de Function Block Diagram (langage en blocs fonctionnels).

FBD est un langage de programmation graphique qui fonctionne comme un logigramme. En complément des blocs logiques simples (ET, OU, etc.), chaque fonction ou bloc fonction du programme est représenté sous cette forme graphique. Pour chaque bloc, les entrées se situent à gauche et les sorties à droite. Les sorties des blocs peuvent être liées aux entrées d'autres blocs afin de former des expressions complexes.

I

IEC 61131-3 Standard international : commandes de logique programmable
Partie 3 : langages de programmation.

IL IL est l'abréviation de Instruction List (liste d'instructions).
Ce langage est une suite d'instructions basiques.
Il est très proche du langage d'assemblage utilisé pour programmer les processeurs.
Chaque instruction est composée d'un code instruction et d'un opérande.

Instance DFB Une instance de type DFB se produit lorsqu'une instance est appelée depuis un éditeur de langage.
L'instance possède un nom et des interfaces d'entrée et de sortie ; les variables publiques et privées sont dupliquées (une duplication par instance, le code n'étant pas dupliqué).
Un type DFB peut comporter plusieurs instances.

Instancier Instancier un objet consiste à allouer un espace en mémoire dont la taille dépend du type de l'objet à instancier. Lorsqu'un objet est instancié, cela prouve qu'il existe et il peut être manipulé par le programme.

INT INT est l'abréviation du format Single INTeger (entier simple) (codé sur 16 bits).
Les butées inférieure et supérieure sont les suivantes : -(2 puissance 31) à (2 puissance 31) - 1.
Exemple :
-32768, 32767, 2#1111110001001001, 16#9FA4.

L

LD LD est l'abréviation de Ladder Diagram (langage schéma à contacts).

LD est un langage de programmation, représentant les instructions à exécuter sous forme de schémas graphiques très proches d'un schéma électrique (contacts, bobines, etc.).

O

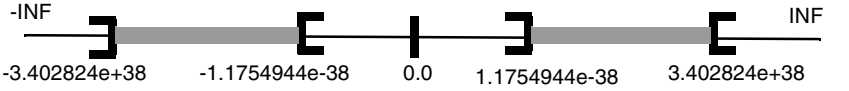
Objets SFC

Un objet SFC est une structure de données représentant les propriétés d'état d'une action ou d'une transition d'un graphe séquentiel.

R

REAL

Le type REAL (réel) est un type codé sur 32 bits.
Les plages de valeurs possibles sont illustrées en gris dans la figure suivante :



Lorsqu'un résultat est :

- compris entre -1,175494e-38 et 1,175494e-38, il est considéré comme étant un `DEN` ;
- inférieur à -3,402824e+38, le symbole `-INF` (pour - infini) est affiché ;
- supérieur à +3,402824e+38, le symbole `INF` (pour + infini) est affiché ;
- indéfini (racine carrée d'un nombre négatif), le symbole `NAN` est affiché.

S

Section

Module programmable appartenant à une tâche pouvant être écrit dans le langage choisi par le programmeur (FBD, LD, ST, IL ou SFC).
Une tâche peut être composée de plusieurs sections, l'ordre d'exécution des sections au sein de la tâche correspondant à l'ordre dans lequel elles sont créées. Cet ordre peut être modifié.

SFC

Abréviation de Sequential Function Chart (diagramme fonctionnel en séquence). Le SFC permet de représenter graphiquement et de façon structurée le fonctionnement d'un automatisme séquentiel. Cette description graphique du comportement séquentiel de l'automatisme et des différentes situations qui en découlent s'effectue à l'aide de symboles graphiques simples.

Sous-programme Module programmable appartenant à une tâche (Mast, rapide) pouvant être écrit dans le langage choisi par le programmeur (FBD, LD, ST ou IL).
Un sous-programme ne peut être appelé que par une section ou un autre sous-programme appartenant à la tâche dans laquelle il est déclaré.

ST Abréviation de Structured Text (langage littéral structuré).
Le langage littéral structuré est un langage élaboré proche des langages de programmation informatiques. Il permet de structurer des suites d'instructions.

Structure Vue dans le navigateur de projet qui représente la structure du projet.

T

Tâche Ensemble de sections et de sous-programmes, exécutés de façon cyclique ou périodique pour la tâche MAST, ou périodique pour la tâche rapide.
Une tâche possède un niveau de priorité, et des entrées et des sorties de l'automate lui sont associées. Ces E/S sont actualisées en conséquence.

Tâche maître Tâche principale du programme.
Elle est obligatoire et est utilisée pour effectuer le traitement séquentiel de l'automate.

TIME Le type `TIME` exprime une durée en millisecondes. Codé sur 32 bits, ce type permet d'obtenir des durées de 0 à (2 puissance 32) -1 millisecondes.

V

Variable Entité mémoire du type `BOOL`, `WORD`, `DWORD`, etc., dont le contenu peut être modifié par le programme en cours d'exécution.

Variable localisée Variable dont la position dans la mémoire de l'automate peut être connue. Par exemple, la variable `Pression_eau` est associée au repère `%MW102`. `Pression_eau` est dite localisée.

Variable non localisée Variable dont la position dans la mémoire de l'automate ne peut pas être connue. Une variable à laquelle aucune adresse n'a été affectée est dite non localisée.

Vue fonctionnelle

Vue permettant d'afficher la partie du programme de l'application via les modules fonctionnels créés par l'utilisateur (voir la définition relative au module fonctionnel).

W

WORD

Le type `WORD` est codé sur un format de 16 bits et est utilisé pour effectuer des traitements sur des chaînes de bits.
Ce tableau donne les butées inférieure/supérieure des bases qui peuvent être utilisées :

Base	Butée inférieure	Butée supérieure
Hexadécimale	16#0	16#FFFF
Octale	8#0	8#177777
Binaire	2#0	2#1111111111111111

Exemples de représentation

Donnée	Représentation dans l'une des bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Index