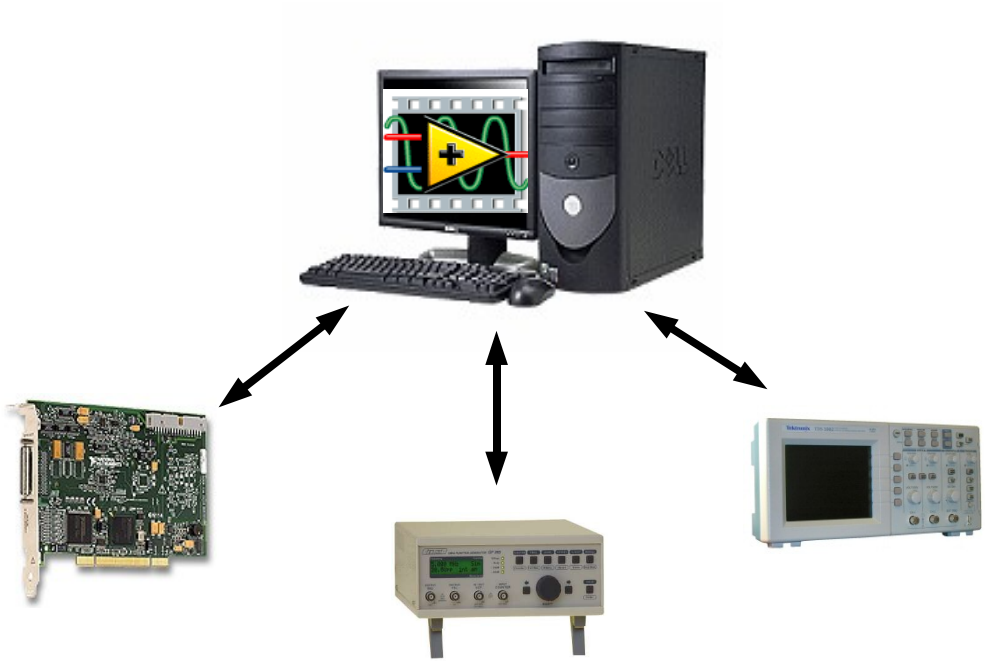
# Le langage de programmation Labview



*IUT1 Grenoble*
*Département GEII1*
*Année 2007/2008*

David FREY
Pierre-Armand DEGRYSE
Jean-Luc AMALBERTI

# Sommaire

# Introduction

Le langage de programmation Labview (Laboratory Virtual Instrument Engineering Workbench) est un environnement de programmation à caractère universel bien adapté pour la mesure, les tests, l'instrumentation et l'automatisation. C'est un programme dont le but est de contrôler et de commander des processus physiques allant du simple capteur ou de l'actionneur jusqu'à une chaîne de fabrication complète.

Le temps nécessaire à l'assemblage d'un système de mesure ou de contrôle/commande est en général négligeable par rapport à celui nécessaire à sa programmation en langage classique (C, Pascal,…). Les interfaces utilisateur développées avec ces langages, sont le plus souvent en langage texte dont il faut apprendre la syntaxe. Les utilisateur peuvent avec Labview avoir à la fois un outil intégré d'acquisition, d'analyse et de présentation des données. Le principal avantage est un gain de temps car ce langage graphique de programmation est beaucoup plus naturel à mettre en œuvre.

Il repose cependant sur les principes généraux de tout système programmé.

Comme nous allons le voir dans la suite, Labview dispose d'un nombre important de fonctions graphiques préexistantes qui permettent facilement d'acquérir les données, de les traiter et d'afficher les résultats.

# 1 Bases de la programmation en LabVIEW

## *1.1 Introduction*

LabVIEW est un langage de programmation graphique qui utilise des icônes, au lieu des lignes de textes utilisées en C par exemple, pour écrire des applications. A l'inverse d'un langage de programmation en ligne de texte où c'est la suite des instructions qui détermine l'exécution du programme. LabVIEW utilise la programmation par flux de données. C'est ce flux de données qui déterminera l'ordre d'exécution du programme.

Dans LabVIEW, vous allez créer une interface utilisateur (face avant) en utilisant un ensemble d'outils et d'objets. L'interface utilisateur correspond à ce qui apparaîtra sur l'écran du PC an mode fonctionnement et qui permettra à l'utilisateur, soit de piloter le programme (donner des entrées), soit au programme d'afficher des informations (sorties). Le principal intérêt de Labview est de permettre facilement de développer ces interfaces à l'aide de bibliothèques pré-existantes.

Le programme proprement dit est ajouté sur la face arrière. C'est un ensemble de codes utilisant un langage de programmation graphique qui permet de contrôler les objets de la face avant. Dans un certain sens, le diagramme se trouvant sur la face arrière ressemble à un flowchart.

## *1.2 Notion de VI*

Les programmes LabVIEW sont appelés instruments virtuels (virtual instruments en anglais) ou encore VI. Ceci est du au fait que leur apparence et mode de fonctionnement ressemble à celui d'instruments physiques comme les oscilloscopes ou les multimètres par exemple. Chaque VI utilise des fonctions qui manipulent des entrées de la face utilisateur ou d'autres sources et affiche les résultats des traitements ou les enregistre dans des fichiers de résultat ou sur d'autres ordinateurs.

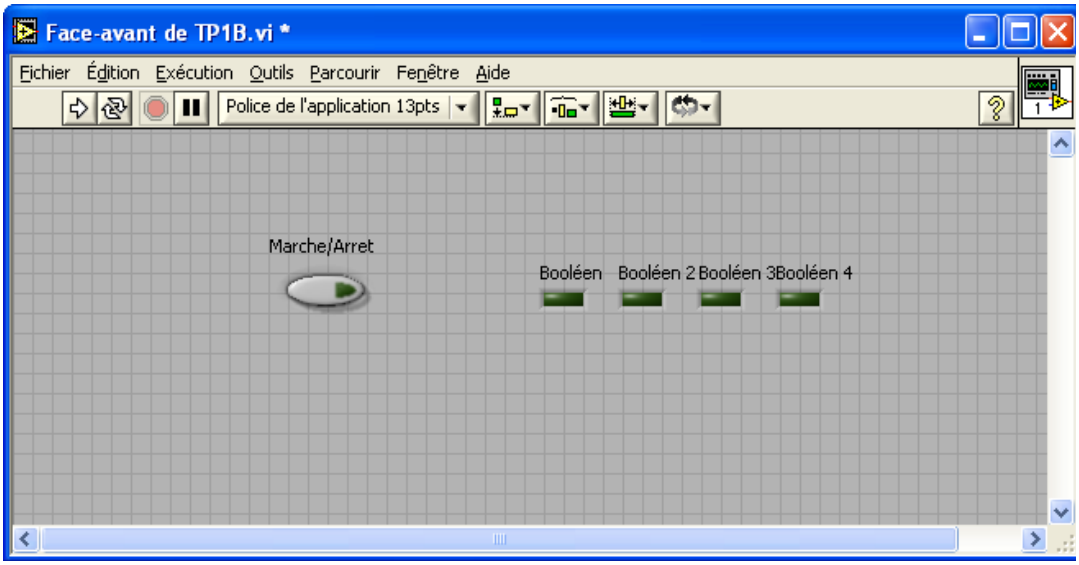
Un VI contient les 3 éléments suivants :

• **Face avant** (**Front panel)**—Sert d'interface utilisateur

• **Face arrière (Block diagram)**—Contient le code source sous forme graphique qui définit les fonctionnalités du VI.

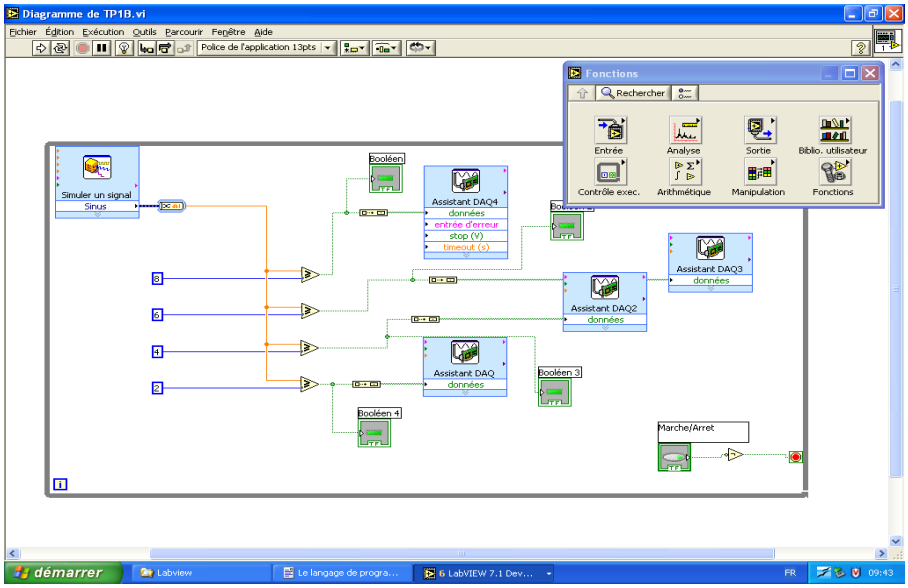
• **Icône et pattes de connections**—Identifie le VI de telle sorte que vous pouvez utiliser le VI dans un autre VI. Un Vi à l'intérieur d'un autre VI est appelé sous-VI. Un sous-Vi correspond à une sous-routine (ou encore fonction) dans un langage de programmation textuel comme le C.

## 1.3 Face avant




La face avant est l'interface utilisateur du VI. Vous allez créer la face avant avec les contrôles (entrées) et les indicateurs (sorties) qui sont les entrées et les sorties du VI qui vont interagir avec l'utilisateur.
Les contrôles peuvent être des interrupteurs, des boutons poussoir, des boites de dialogue, et d'autres composants d'entrée. Les indicateurs sont des graphiques, des LEDs et d'autres systèmes d'affichage. Les contrôles simulent des composants d'entrées qui fournissent au diagramme du VI des données. Les indicateurs simulent des instruments de sortie qui affichent des données qui ont été acquises par le diagramme ou qui ont été générées.

## 1.4 Face arrière (Diagramme block)



Après avoir réalisé la face avant, vous allez ajouter du code en utilisant des représentations graphiques de fonctions pour contrôler les objets situés en face avant. Le diagramme de la face arrière va comporter le code source sous forme de graphique. Les objets de la face avant apparaîtront comme des terminaux (d'entrée ou de sortie) du diagramme.
En plus, le diagramme va pouvoir contenir des fonctions et des structures qui sont fournies dans les librairies de VI de LabVIEW. Des fils vont connecter chacun des nœuds du diagramme en incluant les contrôles et les indicateurs, les fonctions et les structures.

## *1.5 Barres d'outils et palettes*

**Barre d'outil de la face avant**

La barre d'outil donne acces aux outils d'execution et de présentation du programme.

Execution du VI en continu

Pause


Execution du VI

Arret du VI

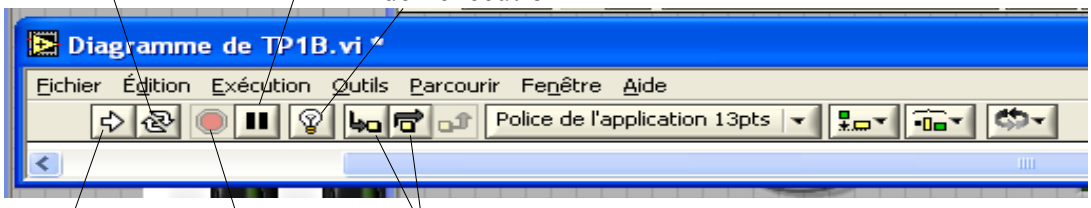
**Barre d'outil du diagramme**

La barre d'outil donne acces aux outils d'execution et de présentation du programme.

Execution du VI en continu

Pause

Animation de l execution




Execution du VI

Arret du VI

Execution en mode pas a pas

**Les palettes LabVIEW**

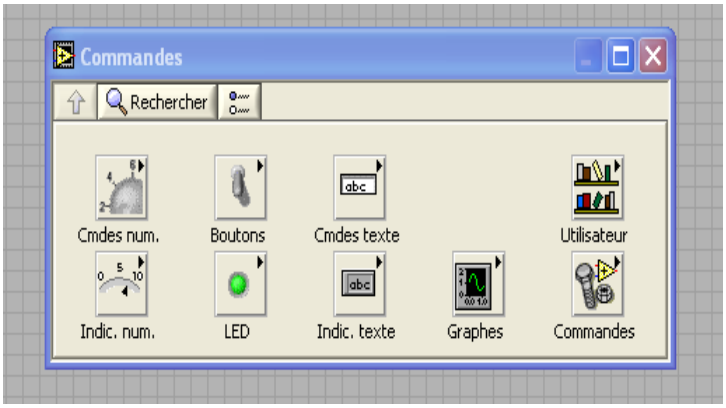
Les palettes labVIEW vous fournissent les outils qui vous permettront de créer et d'éditer la face avant ou le diagramme de la face arrière.

La palette **Outils** est disponible aussi bien sur la face avant que la face arrière. Un outil est un mode de fonctionnement spécial du curseur de la souris. Quand vous sélectionnez un outil, le curseur de la souris se modifie. Utilisez les outils pour travailler ou modifier la face avant ou le diagramme de face arrière.

Sélectionner **Fenêtre»Palette Outils** pour afficher la palette **Outils**. Vous pouvez placer la palette où vous le souhaitez sur l'écran.

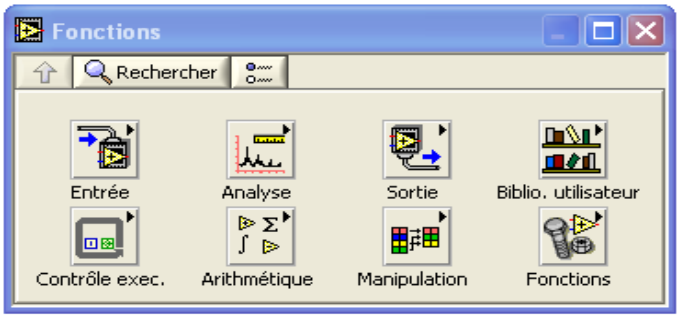
Si la sélection automatique des outils est autorisée et que vous déplacez la souris au dessus d'objets des faces avant ou arrière, LabVIEW sélectionnera automatiquement l'outil adéquat de la palette.

La palette **Commande** est uniquement accessible depuis la face avant. La palette **Commande** contient les contrôles et les indicateurs que vous pouvez utiliser pour créer la face avant. Sélectionner **Fenêtre»Palette Commande** ou faites un click droit sur la fenêtre de travail de la face avant pour faire s'afficher la palette **Commande**. Vous pouvez placer celle-ci n'importe où sur l'écran.



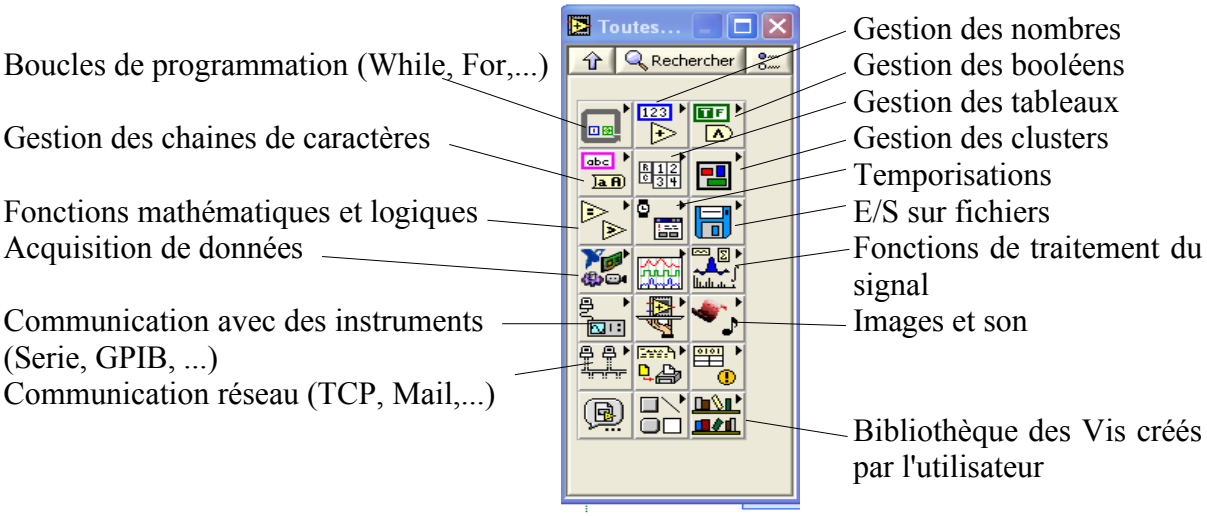

Vous allez y trouver un ensemble de composants déjà créés pour vous permettre de réaliser facilement une interface utilisateur.

La palette **Fonctions** est seulement accessible sur la face arrière. La palette **Fonctions** contient les VIs et les fonctions que vous pouvez utiliser pour créer le diagramme de la face arrière. Sélectionnez **Fenêtre»Palette Fonctions** ou faites un click droit sur la fenêtre du diagramme pour afficher la palette **Fonctions**. Vous pouvez placer la palette **Fonctions** n'importe où sur l'écran.




En bas, a droite, le bouton Fonctions vous donne accès à l'intégralité de la palette.

C'est ici que vont se trouver l'ensemble des fonctions qui ont déjà été préécrites sous Labview, mais vous pourrez également y trouver celles que vous créerez par la suite.

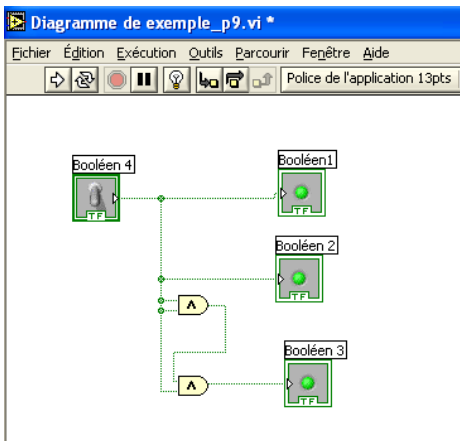



La bibliothèque des fonctions existant sous Labview est assez fournie. Notamment elle est importante pour tout ce qui concerne la gestion des E/S sur fichiers et les communications du PC avec son environnement. De nombreux outils de traitement des données existent également. N'hésitez pas à rechercher d'abord si elles n'existent pas avant de vous lancer dans votre programme.

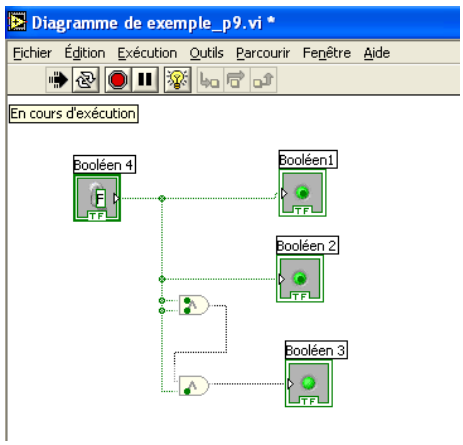
## 1.6 Programmation flux de données

LabVIEW utilise un modèle de programmation de type flux de données pour exécuter ses VIs. Ceci veut dire qu'un élément du diagramme ne s'exécutera que lorsque toutes les données à ses entrées seront disponibles. Quand un élément a terminé de s'exécuter, il fournira la donnée résultat à l'ensemble des éléments qui sont connectés à sa sortie.

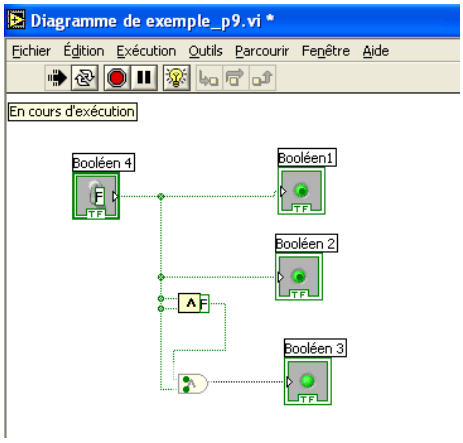
L'exemple ci-dessous nous montre un exemple de déroulement de programme avec le flux de données. Ceci a été fait en mode animation de l'execution comme le prouve la petite ampoule qui est allumée.



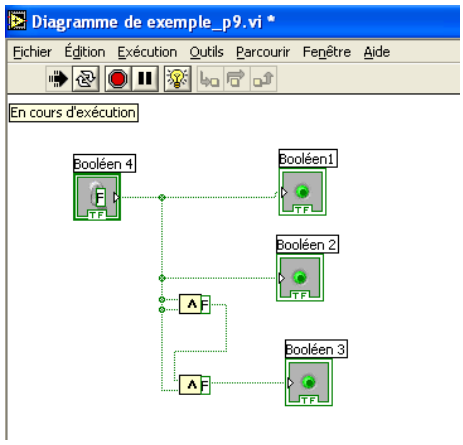

*Etape 1 : La donnée est sur Booléen 4.*




*Etape 2 : La donnée arrive sur Booléen1, sur Booléen 2 et les deux portes &.*




*Etape 3 : La première porte 1 s'execute et envoie la donnée à la deuxième.*



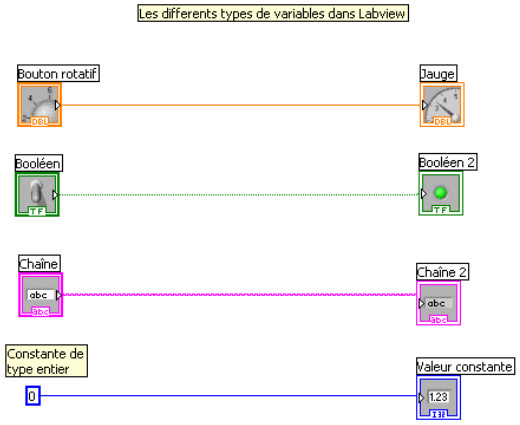

*Etape 4 : Celle-ci s'execute et envoie la donnée à Booléen 3.*

On constate donc ici, que le booléen 3 optiendra sa valeur après les booléens 1 et 2. Dans ce cas simple, il est facile de suivre le fonctionnement du système. Dans des cas beaucoup plus complexes, ce cheminement est plus compliqué.
Le principal problème de la programmation graphique associée à une execution flux de données et qu'il n'est pas toujours facile de s'assurer de l'ordre dans lequel seront exécutées les commandes. En effet, cela va dépendre de la « vitesse de propagation » de la donnée dans le programme.

A l'inverse dans les langages de programmation structurés comme le C par exemple, l'ordre d'execution des commandes est donné par l'enchainement des lignes de codes dans le programme.
Sous Labview, ceci n'est pas le cas, il a toutefois était prévu des fonctions spéciales qui permettent de pouvoir utiliser les boucles classiques des langages structurés (While, For, If, Case, ...), mais il existe en plus des fonctions spéciales qui permettent de s'assurer de l'ordre d'exécution des instructions. Nous verrons cela par la suite.

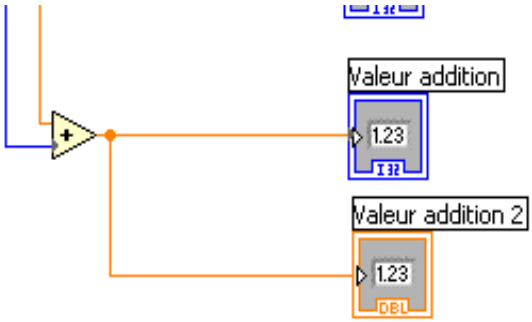
## *1.7 Les différents types de variables sous Labview*

Tout comme la grande majorité des langages de programmation, Labview possède une large palette de variables ( entier signés, non-signés sur, flottants, booléens, chaines de caractères, ...).
Afin de les identifier plus facilement dans le diagramme, les différentes familles de variables possèdent chacune une couleur.

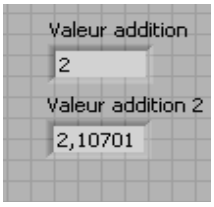



| *Type de données* | *Couleur (type de trait)* |
|---|---|
| Entier (signé et non signé sur 8, 16 et 32 bits) | Bleu (continu) |
| Flottants (16,32,...) et nombres complexes | Orange (continu) |
| Booléens | Vert (tirets) |
| Chaines de caractères | Rose (continu) |

En temps normal, il n'est pas toujours possible de travailler avec des variables de types différents. Néanmoins, Labview fait soit une adaptation automatique si par exemple, vous voulez additionner un flottant et un entier, comme présenté dans l'exemple suivant.

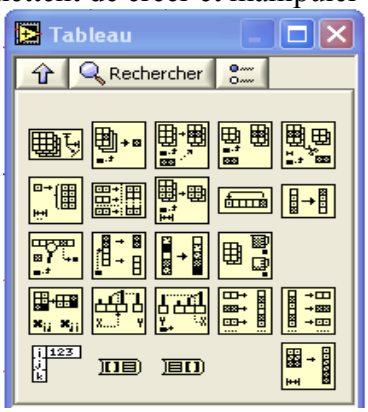



On effectue ici l'addition d'un flottant et d'un entier. Au moment de l'addition, Labview effectue automatiquement une conversion (c'est ce qu'on appelle une opération de type promotion de variable en C). Ceci est symbolisé par un petit triangle gris sur l'entrée de la donnée entière. En effet, labview donne a la variable le type qui contient le plus d'informations (entier -> flottant). Le résultat en sortie étant alors un flottant. Le résultat est ensuite affiché sur un indicateur de type entier 32 bits (I 32) et flottant (DBL pour double soit 32 bits). L'indicateur flottant affichera le nombre en entier. L'indicateur I32, n'affichera que la partie entière comme montré ci-dessous.




## *1.8 Les tableaux sous Labview*

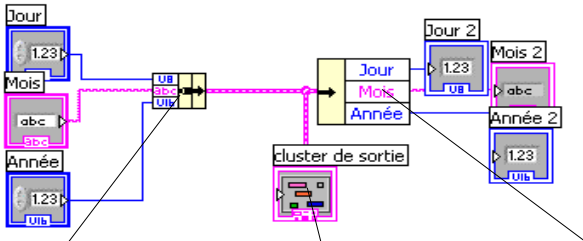
Il est évidemment possible de créer des tableaux de variables sous Labview.
Un ensemble de fonctions permettent de créer et manipuler des tableaux.




On peut faire des tableaux de n'importe quel type de variable. Contrairement a ce que vous avez fait en 'C', il n'est pas obligatoire de créer un tableau au début du programme avec le nombre de cases maximum qu'il pourra avoir. On peut en effet, créer et faire varier la taille d'un tableau en cours de programme. Ceci vous donne plus de latitude dans l'écriture de vos programmes.

## 1.9 Les clusters

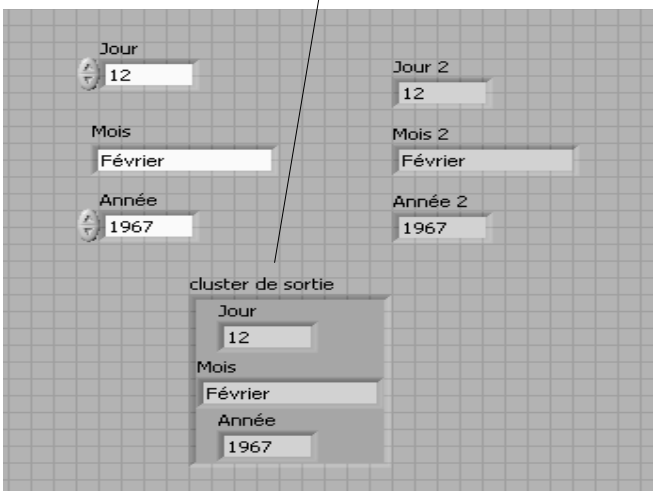
Les clusters correspondent aux structures en 'C'. Il s'agit de variables qui sont composées de variables de différents types. (Booléens, Entiers, Flottants, Chaines de caractères,...)
Les fonctions liées aux clusters sont regoupées dans le menu du diagramme de face arrière **Cluster.**
On dispose d'outils pour créer des clusters à partir de différentes valeurs, d'autres pour récupérer indépendamment les valeurs contenues dans un cluster et enfin des commandes qui permettent d'afficxher les clusters sur la face avant.
Comme présenté dans l'exemple suivant, on peut avoir un cluster « date de naissance » qui sera composée d'un entier sur 8 bits pour le jour, d'une chaine de caractères pour le mois et d'un entier sur 16 bits pour l'année.




Outil de création d'un cluster

Outil de séparation des valeurs d'un cluster

Outil permettant de visualiser sur la face avant les éléments d'un cluster




Bien évidemment, on peut manipuler les cluster comme les autres variables de labview. Notamment, on peut créer des tableaux de clusters.

# 2 Création d'un sous-VI

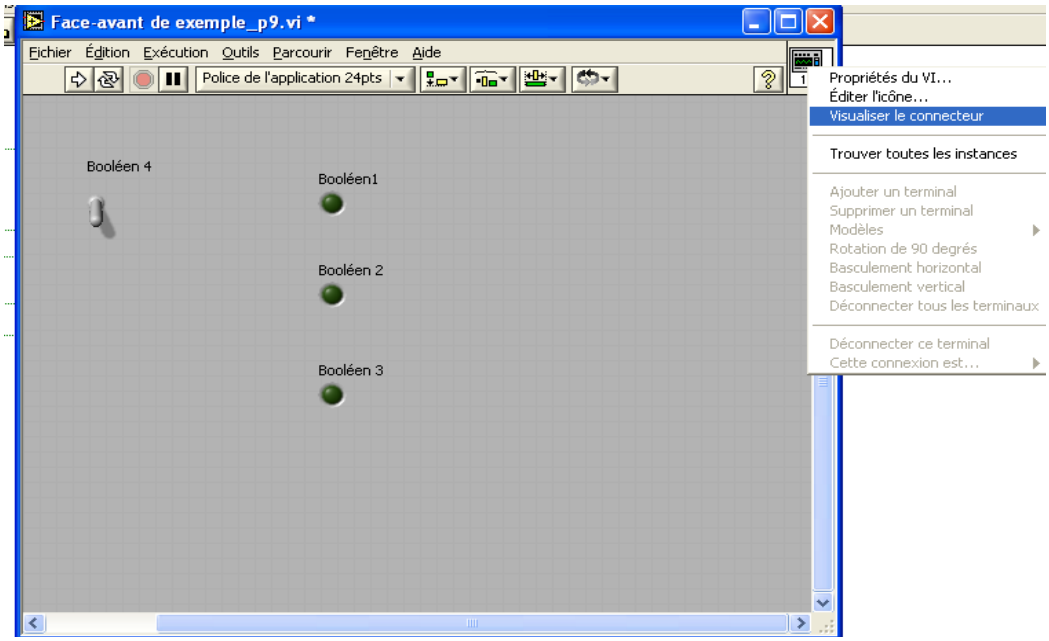
## *2.1 Introduction*

Après avoir réalisé un VI et créé son icône et ses connections, vous pouvez l'utiliser dans un autre VI. Un VI qui est utilisé dans le diagramme d'un autre VI est appelé sous-VI. Un sous-VI est l'équivalent d'une fonction dans les langages de programmation textuels. Il disposera de paramètres d'entrée qu'il faudra lui indiquer et aura des paramètres de sortie. Le nœud n'est pas le sous-VI en lui-même, de même que l'appel d'une fonction n'est pas la fonction elle-même.

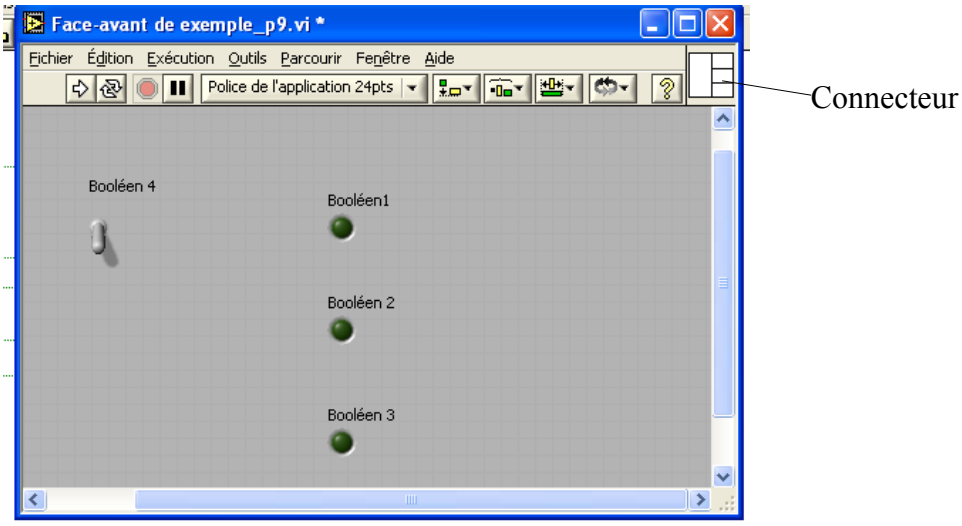
## *2.2 Configurer les bornes de connexion*


Pour pouvoir utiliser un VI comme un sous-VI, vous allez devoir lui configurer le connecteur avec les paramètres d'entrée/sortie. Pour faire ceci, vous allez configurer les terminaux qui correspondent à des contrôles (paramètres d'entrée) ou à des indicateurs (paramètres de sortie) de ce VI. Ceci est similaire à la liste des paramètres pour les appels à des fonctions.

Vous pouvez définir les connections en assignant à chacune des bornes du composant que vous allez créer un contrôle ou un indicateur de la face avant. Les paramètres d'entrée sont les contrôles, les paramètres de sortie sont les indicateurs. Pour définir une connexion, faites un click droit dans le coin supérieur droit de la fenêtre de face avant et sélectionnez **Visualisez le connecteur** du menu déroulant.



Le schéma des connecteurs remplace l'icône. Chaque rectangle du schéma des connecteurs représente un terminal. Utilisez les rectangles pour fixer les entrées et les sorties. Le nombre de terminaux que LabVIEW affiche sur le schéma des connecteurs dépend du nombre de contrôles et d'indicateurs situés sur la face avant. Par exemple ici, nous en avons quatre.




Le schéma des connecteurs peut avoir au plus 28 terminaux. Si vous utilisez sur votre face avant plus de 28 contrôles et terminaux, vous devrez les regrouper en créant des clusters et assigner ces clusters à un terminal du schéma des connections.

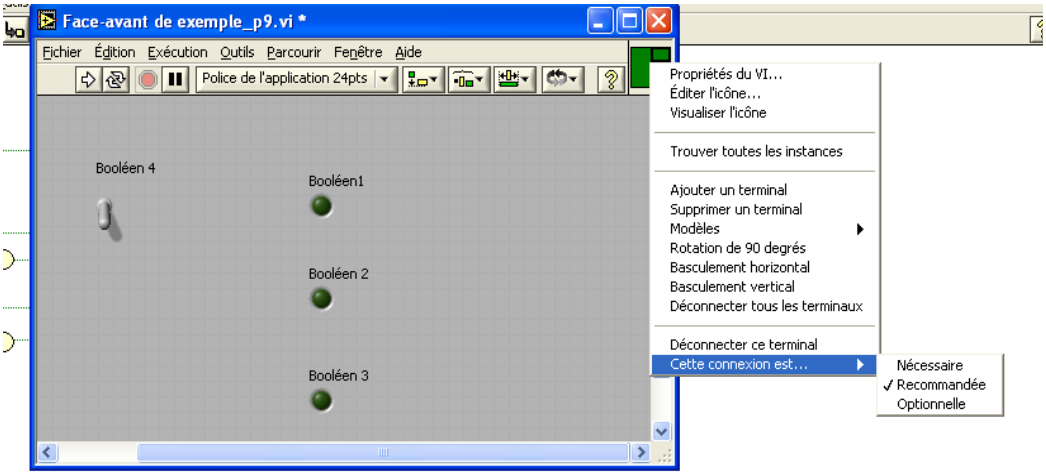
Vous pouvez sélectionner un Connecteur différent de celui que vous propose Labview en faisant un click droit de la souris sur le schéma des connexions et en sélectionnant **Modèles**. Sélectionnez un schéma de connexion avec des terminaux supplémentaires. Vous pouvez laisser les terminaux supplémentaires non connectés pour le moment où vous en aurez besoin. Cette flexibilité vous permet d'effectuer des modifications avec un impact minimal sur la hiérarchie des VIs.

Si vous utilisez souvent un groupe de sous-VIs ensemble. Donnez leur un schéma de connexion cohérent avec les entrées communes situées à des endroits proches pour vous aider à vous souvenir où connecter chaque entrée. Si vous créer un sous-Vi qui va générer une donnée utilisée par un autre sous-VI, essayez d'aligner les entrées afin de simplifier le câblage. Placez les clusters **Entrée d'erreur** dans la partie basse à gauche de la face avant et le cluster **Sortie d'erreur** dans le coin en bas à droite.

## *2.3 Positionner des entrées et des sorties (nécessaires, recommandées et optionnelles)*

Vous pouvez désigner quelle entrées et sorties sont nécessaires, recommandées ou bien optionnelles pour éviter aux utilisateurs des sous-VI d'oublier des connexions.

Faites un click droit de la souris et sélectionnez **Cette Connexion est** du menu. Une marque indique la configuration du terminal. Sélectionnez **Nécessaire**, **Recommendée**, ou **Optionelle**.

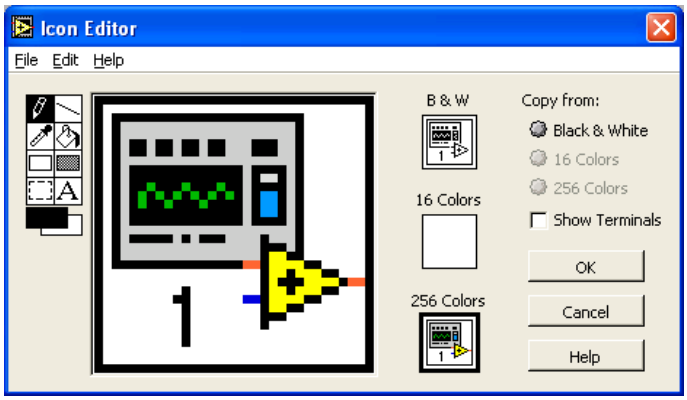



Pour les terminaux d'entrée, **Nécessaire** signifie que le diagramme bloc de la face arrière dans lequel est implanté le sous-VI **sera rompu si l'entrée nécessaire n'est pas câblée**. **Nécessaire** n'est pas disponible pour les terminaux de sortie.
Pour les terminaux d'entrée et de sortie, **Recommandée** veut dire que le diagramme de schéma bloc dans lequel le sous-VI est implanté **pourra fonctionner si les entrées ne sont pas connectées**, mais la boîte de dialogue **Warnings** va générer un message d'alerte que l'entrée n'a pas été connectée. **Optionnelle** veut dire que le diagramme de bloc dans lequel est implanté le sous-VI pourra être mis en oeuvre sans générer d'alerte si les terminaux d'entrée ou de sortie ne sont pas connectés.

Les entrées et les sortie des VIs dans vi.lib sont déjà marquées comme étant : **Nécessaire**, **Recommandée**, ou **Optionnelle**. LabVIEW met par défaut les entrées et les sorties de VIs créées comme **Recommandée**. Mettez le terminal en position **Nécessaire** seulement si le VI doit avoir cette entrée ou cette sortie connectée pour fonctionner correctement.

Dans la fenêtre d'aide contextuelle **Aide Contextuelle**, les connections nécessaires sont en **GRAS**, les connections recommandées sont en texte normal et les connections optionnelles sont atténuées si vous avez sélectionner la vue **détaillée** ou encore n'apparaissent pas si vous avez sélectionné la vue **Simple**.

## 2.4 Créer une Icône



Chaque VI montre une icône dans le coin supérieur droit de la face avant et dans le diagramme de face arrière. Une icône est une représentation graphique d'un VI. Il peut contenir du texte, des images ou une combinaison des deux. Si vous utilisez un VI comme un sous-VI, l'icône permet d'identifier le sous-VI à l'intérieur du diagramme de face arrière.

L'icône par défaut contient un nombre qui indique combien de nouveaux VIs vous avez ouvert depuis que vous avez lancé LabVIEW. Créez des icônes personnalisés pour remplacer l'icône par défaut en faisant un click-droit sur l'icône dans le coin supérieur droit de la face avant ou du diagramme de face arrière et sélectionnez **Editer l'Icône** dans le menu ou alors double cliquez sur l'icône dans le coin supérieur droit de la face avant.

Vous pouvez également importer une image d'ailleurs, la mettre dans le coin supérieur droit de la face avant ou du diagramme de face arrière. LabVIEW convertit l'image en une icône de 32 points par 32.

Vous pouvez dessiner un icône monochrome, avec 16 ou 256 couleurs en fonction du moniteur que vous utilisez. LabVIEW utilise l'icône monochrome pour l'impression sauf si vous avez une imprimante couleur.

## *2.5 Créer des Sous-VIs de parties d'un VI*

Vous pouvez convertir un VI en un sous-VI en utilisant l'outil de pointage pour sélection la partie de programme que vous voulez utiliser et sélectionnez **Edit»Create SubVI**. Une icône qui représente le nouveau sous-VI apparaît et remplace la zone sélectionnée du diagramme. LabVIEW crée les contrôles et les indicateurs pour le nouveau sous-VI et connecte le sous-VI aux liaisons qui existent déjà dans le VI principal.

Créer un sous-Vi à partir d'une sélection est très pratique, mais nécessite tout de même de prendre garde à la hiérarchie logique qui peut exister entre les Vis. Faites bien attention aux objets que vous incluez dans la sélection et évitez de changer les fonctionnalités du VI créé.

## *2.6 Equations*

**Convertir des ºC en ºF**

La formule pour convertir des degrés Celsius en degrés Fahrenheit est la suivante :

$$°F = (1.8 * °C) + 32$$

Par exemple, pour convertir une température de 100° Celsius en degrés Fahrenheit, multipliez la température en °C par 1,8 pour obtenir 180 puis ajoutez 32 pour obtenir 212 degrés Fahrenheit.

**Pente d'une droite**

La formule de la pente d'une droite est la suivante:

$$Pente = (Y2 – Y1) / (X2 – X1)$$

ou (X1, Y1) et (X2, Y2) sont des points de la ligne.

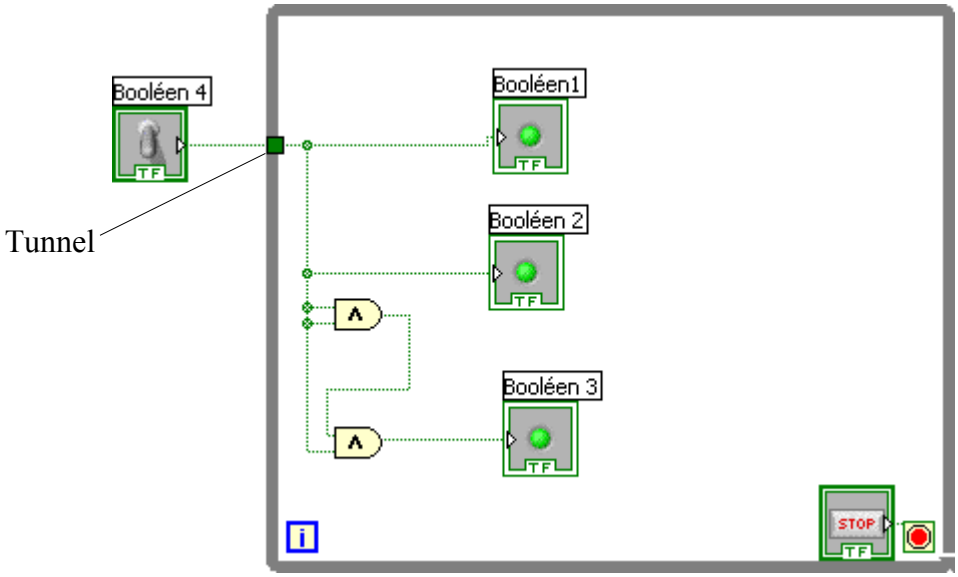
# 3 LabVIEW Structures de programmation

## *3.1 Introduction*

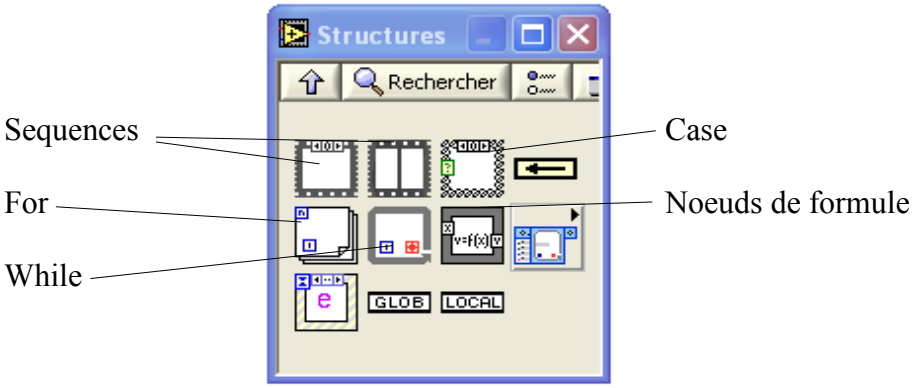
Les structures sont des représentations graphiques identiques aux structures boucles et case de programmation 'C'. Vous pouvez utiliser les structures pour répéter des parties de code ou pour exécuter du code de façon conditionnelle ou dans un ordre spécifique.

Comme les autres noeuds, les structures disposent de terminaux qui permettent de les connecter à d'autres parties du programme, s'exécutent automatiquement dès que les données sont présentent et envoient en sortie les résultats quand elles ont terminé de s'exécuter.

Chaque structure dispose d'une bordure dont la taille peut être ajustée afin d'englober la partie du diagramme qui doit être exécutée selon cette loi. La partie de programme contenue dans la structure est appelée un sous-diagramme. Les connexions qui permettent de fournir et de récupérer des données en dehors de la structure sont appelées des tunnels. Ce sont des points de connexion sur le bord de la structure.




Vous allez utiliser les structures situées dans la palette FUNCTION – STRUCTURES pour contrôler comment un diagramme va s'exécuter.




**Boucles For**—Execute un diagramme un nombre donné de fois.

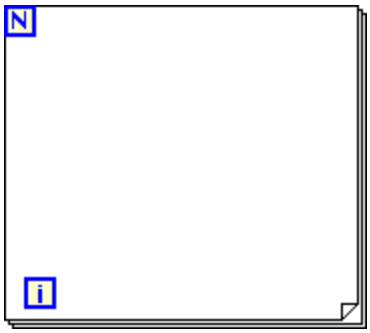
**Boucles While** —Execute un diagramme jusqu'à ce qu'une condition est réalisée.

**Structure Case** —Contient plusieurs sous-diagrammes. Un seul d'entre eux s'exécutera en fonction de la valeur en entrée de la structure.

**Structures de Sequence** —Contient un ou plusieurs sous-diagrammes qui s'exécuteront de façon séquentielle. Cette structure s'avère très utile sous Labview pour synchroniser différentes parties du programme.

**Noeuds de formules**—Réalise des opérations mathématiques sur les entrées numériques.

## *3.2 Boucles For*

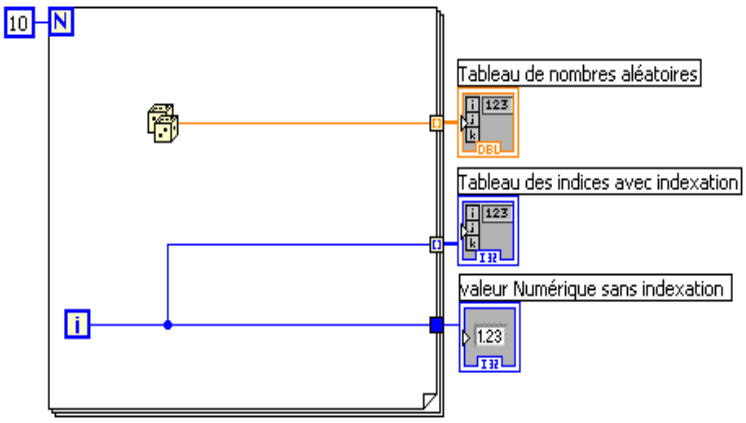



Une boucle For exécute un sous-diagramme un nombre donné de fois.

La valeur dans la connexion d'entrée de comptage (N) indique le nombre de fois que le sous-diagramme devra être répété. Cette valeur doit être fixée de façon explicite en connectant une valeur à l'extérieur de la boucle sur l'entrée N. Le compte peut également être implicite en utilisant l'auto-indexation.

La borne d'itération (terminal de sortie) contient le nombre d'itérations effectuées. Le comptage se fait toujours à partir de 0. A la première itération, la valeur de « i » est 0.

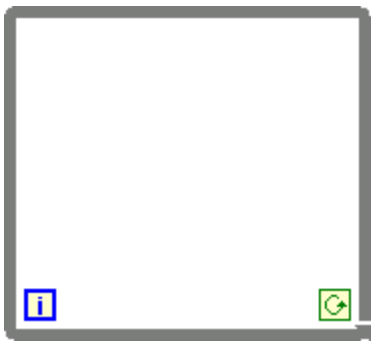
Le compteur et l'itération sont des entiers long signés. Si vous connectez un flottant, LabVIEW récupérera la partie entière. Si vous mettez 0 ou une valeur négative, la boucle ne s'exécute pas.




Dans l'exemple ci-dessus, on voit que les bornes de sorties sont de types.
Lorsque celles-ci sont pleines, se sont des bornes standard. La valeur renvoyée en sortie de la boucle For est la dernière valeur stockée dans le terminal. Dans ce cas là, il n'y a pas ce que l'on appelle « indexation ».
Pour les deux autres bornes, l'indexation a été activée. Ceci se fait en faisant un clique droit avec la souris sur le tunnel de sortie et en sélectionnant :**Activer l'indexation.**
A chaque tour de la boucle For, la valeur en sortie sera stockée dans un tableau et ajouté aux données des tours précédents.

Vous pouvez ajouter des registres à décalage pour passer des valeurs d'une itération à une autre. Ceci sera présenté plus avant.

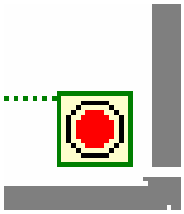
## *3.3 Boucles While*

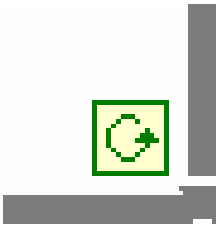



La boucle While va exécuter un sous-diagramme jusqu'à ce qu'une condition prédéfinie a été atteinte.

La boucle While va exécuter le sous-diagramme jusqu'à ce que le terminal condition (qui est une entrée) reçoive la valeur appropriée (valeur booléenne). Le fonctionnement par défaut est que la boucle continue quand la condition est Vraie. Donc pour l'arrêter il faut que celle-ci

devienne Fausse. La condition pour laquelle la boucle continue peut-être modifié en cliquant droit sur le terminal ou sur le bord de la fonction while et en sélectionnant  arrêt si Vraie.



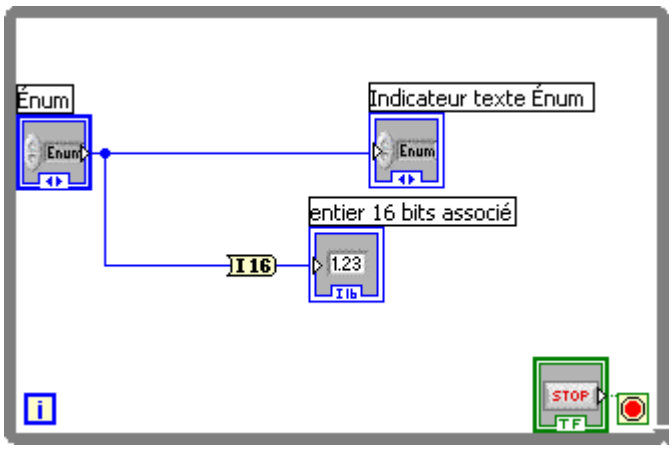


Arrêt sur condition vraie (While (toto=0)...)  Continuer sur condition vraie (While (toto=1))

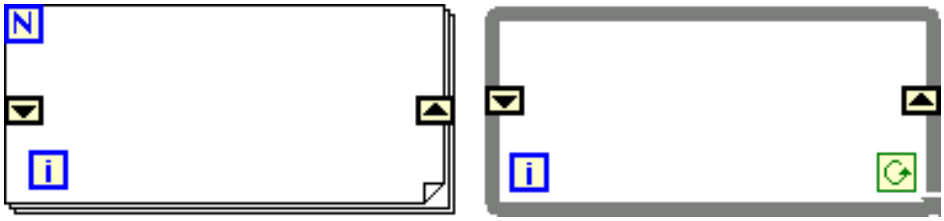
Le test de continuation du While est fait à la fin de l'itération donc celle-ci s'execute au moins une fois. Ceci s'apparente à la fonction Do ...While (..) en C. Si la condition d'arrêt n'est pas cablée, le Vi ne s'exécute pas.

Le terminal d'itération (i) contient le nombre d'itérations qui ont été effectuées. Le décompte commence toujours à partir de 0.

L'ajout de registres à décalage dans la boucle While permet de passer des variables d'une itération à la suivante.




## *3.4 Registres à décalage dans les boucles.*




On peut utiliser des registres à décalage pour transférer des valeurs d'une itération à la suivante dans le cas des boucles For et While. C'est l'équivalent des variables locales à une boucle dans le cas de la programmation C.

Le registre à décalage apparaît comme une paire de terminaux dont les flèches pointent dans des directions opposées. Le terminal de droite (flèche montante) recevra la donnée à conserver à la fin de l'itération.
Cette valeur sera récupérée dans le terminal de gauche à l'itération suivante.
La création d'un registre à décalage se fait par un click droit sur le bord de la boucle et Ajouter un registre à décalage (shift register).

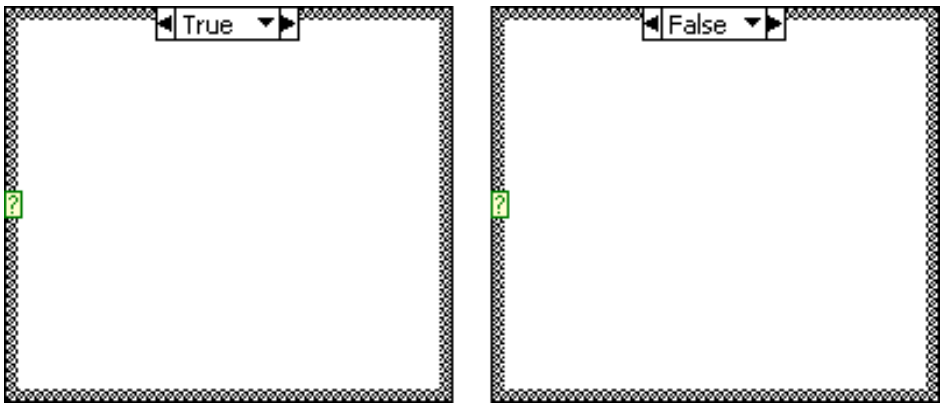
Le registre à décalage transfert tous les types de données et peut la convertir au format du premier objet qui lui est connecté. La valeur qui est envoyée au registre à décalage doit être du même type.
On peut créer plusieurs registres à décalage différents si on veut transférer diverses variables.

Après la dernière boucle la dernière valeur calculée reste dans le terminal de droite. Si on le connecte vers l'extérieur, cette valeur peut-être utilisée par les autres diagrammes.

Si le terminal de gauche n'est pas initialisée la boucle utilisera la valeur stockée lors de la dernière exécution de la boucle. Si celle-ci n'a jamais été utilisée, elle utilisera la valeur par défaut du type de variable.

Vous pouvez utiliser un registre à décalage pour exécuter un VI de façon répétitive de telle sorte que la dernière valeur de l'exécution précédente du VI devienne la première valeur à prendre pour la nouvelle exécution.
Le fait de ne pas initialiser un registre à décalage peut permettre de conservé une variable d'état entre deux lancement successifs du VI.

## *3.5 Structures Case*




Une structure de type Case possède au moins deux sous-diagrammes. Seulement l'un d'entre eux est visible à la fois et la structure n'en exécute qu'un seul à la fois. Une donnée en entrée ( ?) détermine lequel des sous-diagrammes va s'exécuter.

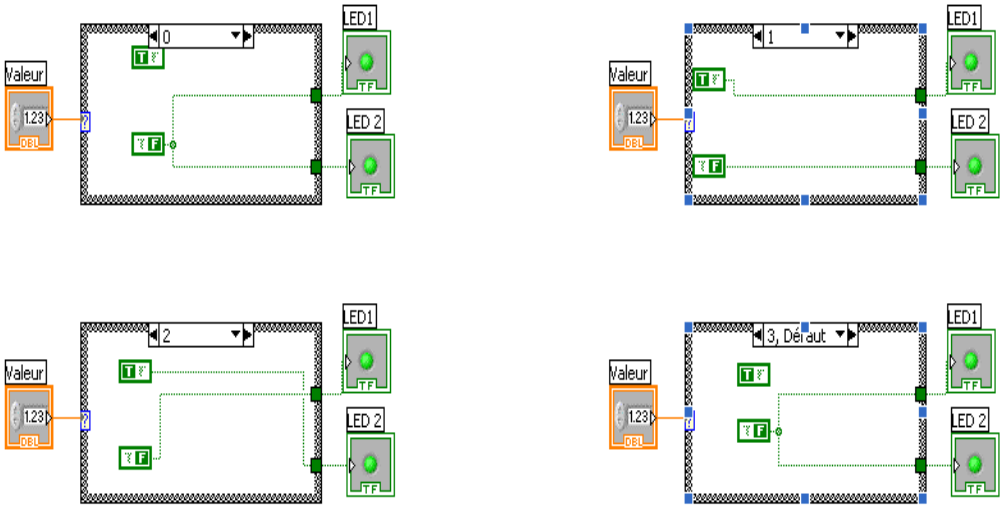
La structure Case est similaire à la programmation "if...then...else" ou "case" en language C.

Le label du cas considéré est affiché sur la partie haute de la structure. On a également des flèches de chaque côté qui permettent de passer d'un cas aux suivants. On peut aussi sélectionner à l'aide du menu déroulant en cliquant sur la flèche vers le bas.

Une valeur doit être câblée sur le terminal de sélection (?) afin de déterminer quel diagramme du case doit être effectué. On peut y câbler une variable de type entier, booléenne, une chaîne de caractère ou un type énuméré. Le terminal de sélection peut être positionné à n'importe quel endroit sur la frontière gauche de la boite Case. Si en entrée on a un booleen, la structure aura un cas « Vrai » (True) et un cas « Faux » (False).
Si on y connecte un autre type de variables, la structure pourra avoir un grand nombre de cas différents.

Il est possible de spécifier un cas "Par défaut" pour les cas où la valeur de sélection ne correspond à aucune valeur de cas. Sinon, il faut explicitement faire la liste de toutes les valeurs possibles. Par exemple, si l'entrée de sélection est un entier et que les cas définis sont 1,2 et 3, il est nécessaire de spécifier un cas « Par défaut » au cas ou l'entier serait supérieur à 3.

On peut créer de nombreux tunnels d'entrée et de sortie dans une structure case. Les entrées seront disponibles pour tous les cas, mais il n'est pas obligé de les utiliser.
La création d'un tunnel de sortie le fera apparaître sur l'ensemble des sous-diagrammes de la structure Case. Si l'un des tunnels n'est pas câblé dans un sous-diagramme, celui-ci apparaîtra en blanc. Les types de données connectées peuvent être différentes d'un sous-diagramme à l'autre, mais doivent rester compatibles.
Sinon, il est possble d'utiliser une valeur par défaut pour les cas où le tunnel de sortie ne serait pas connecté. Ceci se fait en cliquant droit sur le tunnel et en choisissant : **Use Default If Unwired.**
Dans l'exemple ci-dessous, on va allumer des Leds en fonction d'une valeur. Si valeur=0, aucune LED, si valeur=1, LED1, si valeur=2, LED2, si c'est une autre valeur, aucune LED.




On aura donc 4 conditions possibles, valeur égale à 0, 1 2 ou le cas par défaut si valeur est supérieur à 2.

## *3.6 Structures Sequence*

**Généralités**

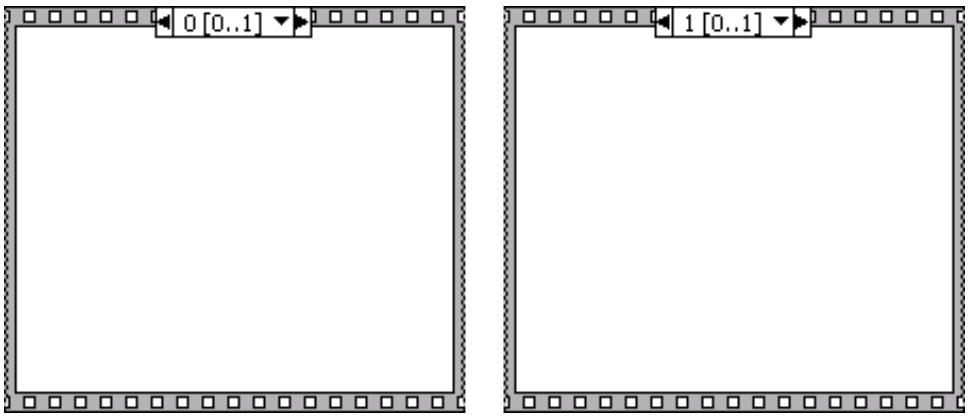
Il peut s'avérer utile dans un programme d'avoir un ensemble d'actions qui vont s'exécuter de façon séquentielle. La structure séquence contient un ou plusieurs sous-diagrammes qui vont s'exécuter l'un après l'autre. La structure séquence existe sous deux formes différentes.

La structure séquence peut être utilisée lorsqu'il n'existe pas de dépendance naturelle des données. Dans la structure flux de données c'est l'arrivée des données qui engendre l'exécution d'une commande. Donc, la structure séquence sera utilisable lorsque ce mode de fonction n'amènera pas un fonctionnement correct.

Par contre, à l'intérieur comme à l'extérieur de la structure séquence, le fonctionnement par flux de données reste valable.

La structure séquence ne fera aucune exécution et ne retournera aucune valeur tant que le dernier diagramme n'aura pas été exécuté.

**La structure empilée**




Chaque plan de la séquence correspond à une action. Les plans se dérouleront les uns après les autres (plan 0, puis 1, 2,...). Pour ajouter des plans avant ou après un plan donné, il faut fair eun clique droit avec la souris sur la séquence et choisir **Ajouter une séquence avant /après.**

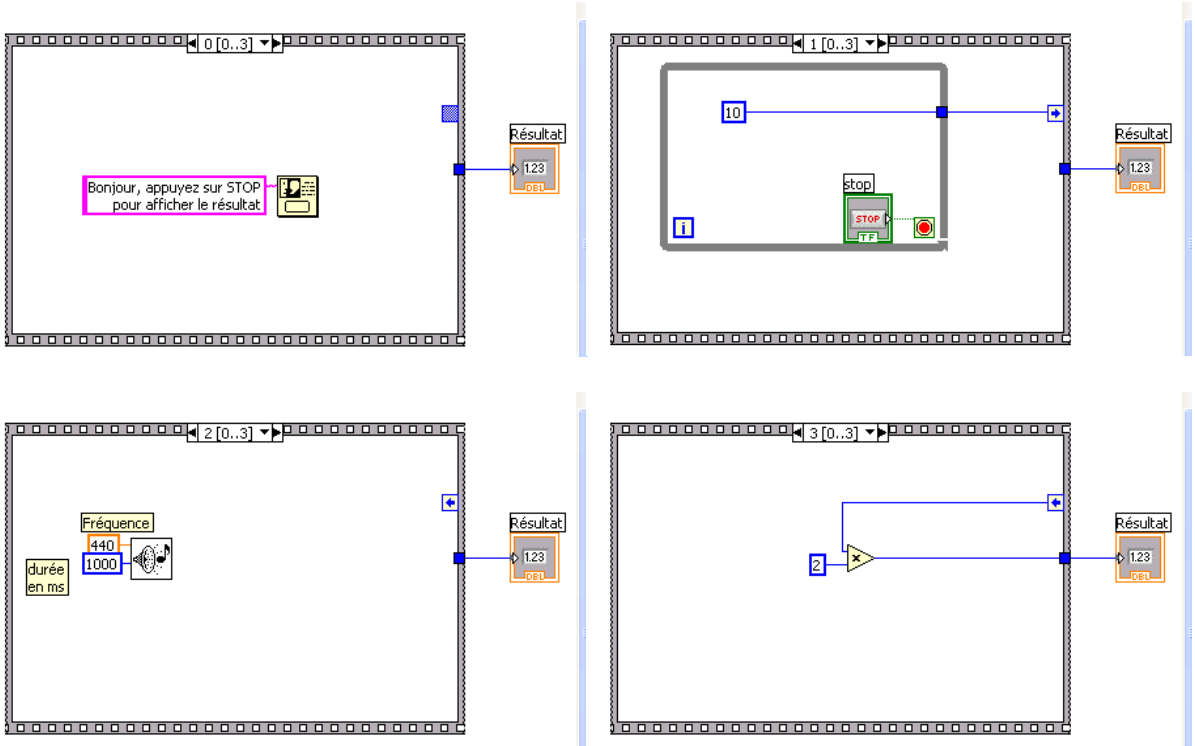
L'avantage de cette structure est qu'elle est compacte, l'inconvénient est qu'on ne voit pas toutes les étapes en une fois.

Le menu déroulant sur le haut de la structure est similaire à la fonction Case et permet de sélectionner les différents sous-diagrammes. Par contre, contrairement à la structure Case, il n'est pas possible de changer la valeur qui y est indiquée, LabVIEW ajustant tout seul les valeurs.

Pour passer les données d'une partie de la séquence à la suivante, on utilise une « sequence local terminal » de la partie contenant la donnée. La partie suivante de la structure sequence voit apparaître une flèche entrante qui indique qu'il s'agit d'une donnée pour cette partie. Cette donnée ne peut par être utilise dans des plans qui précèdent celui où elle est définie.

Par exemple, on vous présente un porgramme dont l'analyse structurée est la suivante :

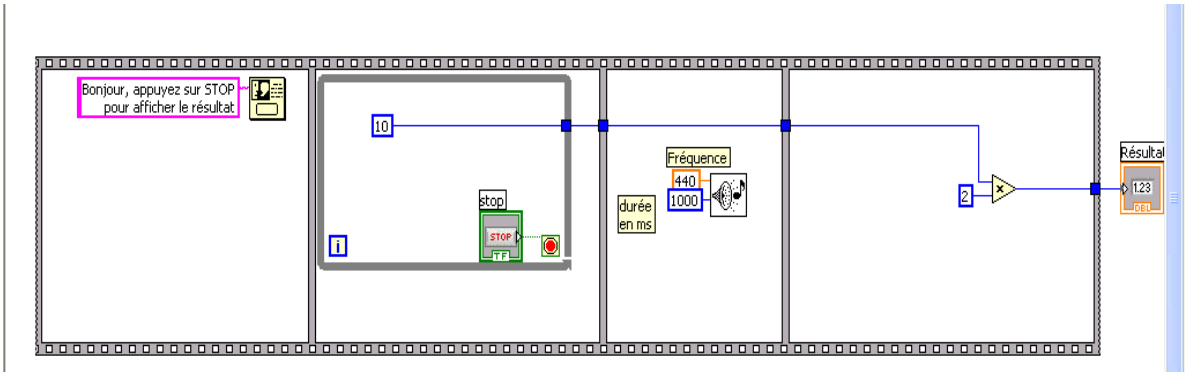
*Ouvrir une fenêtre et afficher le message "Bonjour, appuyez sur STOP pour afficher le résultat"*
*Attendre confirmation OK du message*
*Tant que (STOP est faux)*
*Ne rien faire*
*Fin tant que*
*Bip*
*Effectuer le calcul*
*Afficher le résultat*



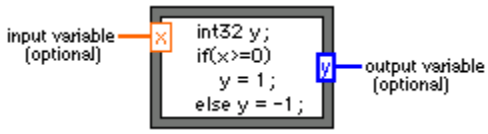

On distingue bien différentes étapes.
A l'étape 0, on génère le texte. La boîte de dialogue attend qu'on appuie sur son bouton pour finir de s'exécuter.
A l'étape 1, tant qu'on a pas appuyé sur le bouton STOP, la boucle while reste active. La valeur 10 n'est envoyé sur la variable locale de séquence qu'à ce moment là.
Il faut noter que la variable locale de séquence n'est pas active àl'étape 0 et que dans les étapes suivantes, on pourra lire son contenu mais on ne pourra plus y écrire.
A l'étape 2, la fonction permet de générer un son. Enfin, à l'étape 3, on effectue le calcul et on envoie le résultat en sortie.

**La structure déroulée**

Ici, les étapes sont posées les unes à côté des autres.
Le principe de fonctionnement est le même. Les étapes se déoulant de la gauche vers la droite.
On vous présente ci-dessous le même programme que précédemment mais avec une structure déroulée.




## 3.7 Noeuds de formule




Le noeud de formule est un bloc basé sur du texte permettant facilement d'implanter des équations mathématiques. Ceci se fait sans avoir besoin d'accéder à du code ou des applications externes et sans avoir à connecter des fonctions arithmétiques bas niveau pour créer des équations. On peut également mettre en œuvre des if, des switch ... case, des boucles while, for ou do familières aux utilisateurs de C. Attention, elles sont similaires mais pas identiques.

Les noeuds de formule sont utiles pour les équations qui ont de nombreuses variables et sont compliquées ou encore pour mettre en oeuvre des fonctions C existantes. Celles-ci peuvent être copier-coller à l'intérieur des noeud de formules.

Les noeuds de formule utilisent la vérification de type pour être sûr que les indexes de tableau sont des données numériques et que les opérandes pour les opérations sur les bits soient des entiers.
Les noeuds de formule vérifient également que les indexes des tableaux restent dans l'échelle. Pour les tableaux, une valeur hors des limites vaut par défaut zéro et un assignement hors des limites retourne un nop.

Les noeuds de formule font également les conversions de type automatiquement.

Les noeud de formule se trouvent dans la palette **Fonctions»Structures** et **Fonctions»**

**Mathematique»Formule** et créent une boîte dont la taille peut être changée comme pour les boucles For, While et les structures Case et Sequence. Par contre, au lieu de contenir des diagrammes, ils contiennent des instructions proche du C, limitées par des points virgule. Comme en C, on peut y adjoindre des commentaires (/*commentaires*/).

Quand vous travaillez avec des variables, rappelez-vous les points suivants :

• Il n'y a pas de limites quand au nombre de variables ou d'équations.

• Deux entrées ou deux sorties ne peuvent pas avoir le même nom. Par contre, une entrée et une sortie peuvent avoir le même nom.

• On déclare une variable d'entrée en cliquant avec le bouton droit sur le bord du noeud de formule avec la souris et en sélectionnant **Ajouter une entrée** du menu. On ne peut pas déclarer de variables d'entrée à l'intérieur de noeud de formule.

• On déclare une variable de sortie en cliquant avec le bouton droit sur le bord du noeud de formule avec la souris et en sélectionnant **Ajouter une sortie** du menu. Le nom de la variable de sortie doit être soit celui d'une entrée soit le nom d'une variable déclarée à l'intérieur du noeud de formule.

• On peut changer une variable d'entrée en variable de sortie et vice versa en cliquant droit et choisissant : **Changer en entrée** ou **Changer en sortie**.

• Une variable interne peut être utilisée sans être reliée en entrée ou en sortie.

• Tous les terminaux d'entrée doivent être connectés.

• Les variables peuvent être des flottants. Ceci dépend de la précision du PC. On peut aussi utiliser des entiers ou des tableaux de valeurs comme variables.

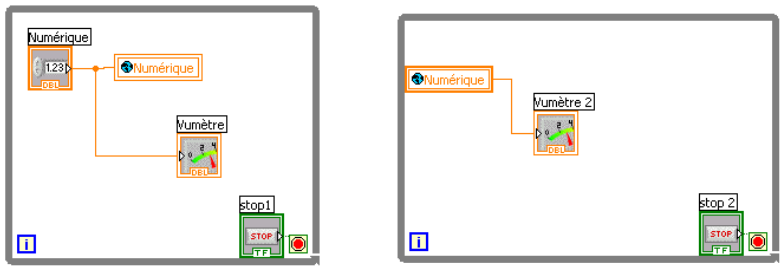
• Les variables sont sans unités.

## *3.8 Les variables locales et globales*

Le langage de programmation Labview repose sur le flux de données. Cependant, on peut parfois avoir envie de transférer des données entre deux parties de programme qui sont indépendantes les unes des autres. Ou alors, lorsque la structure du flux de données, du fait de la mise en oeuvre des boucles ou des sous-VI, ne permet pas de les relier facilement.
Si les variables locales ne peuvent être utilisées que dans le VI où elles sont définies, les variables globales peuvent être utilisées dans le VI, mais également dans les sous-VIs.

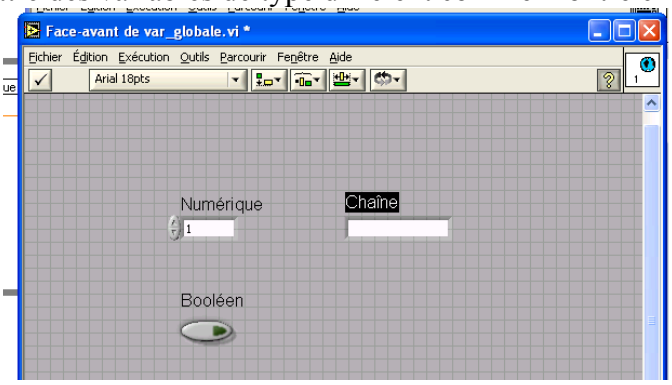
Il est donc nécessaire de les déclarer en précisant le type de la variable (Entier, Booléen,...) et le format (scalaire, tableau,...). Les variables globales seront définies dans un autre VI qui contiendra la définition de l'ensemble des variables globales.

L'exemple ci-dessous montre un VI avec deux boucles while indépendantes. Du fait même du fonctionnement flux de données, il n'est pas possible d'envoyer une donnée de la première boucle while à la seconde. Sauf si l'on arrête la première.




On peut ici utiliser avantageusement des variables globales. Pour en placer une faire : **Fonctions >> Structures >>Glob.** Vous verrez apparaître l'icône suivant :

En cliquant sur l'icône vous ouvrirez une face avant. Vous y placer vos variables globales. Ceci peut être fait en mettant des indicateur ou des commandes numériques, booléennes,... On peut très bien mettre des variables de type différent comme montré ci-dessous.




Enfin, il suffit de sélectionner la variable globale pour obtenir la face arrière précédente.

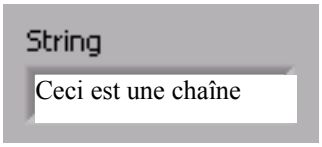
Exercices

1) Remplir un tableau avec des valeurs aléatoires puis l'afficher sur un indicateur.
   On aura un bouton Stop. Tant que celui-ci ne sera pas actionné, on génèrera une valeur numérique aléatoire que l'on stockera dans un tableau. Puis on affichera ce tableau sur l'écran.

2) Faire un chenillard sur 4 LEDs avec Marche/Arret et changement de sens sur la LED.

# 4 Chaines de caractères et mise en oeuvre des fichiers

## *4.1 Les chaines de caractères*




Une chaîne de caractères est une séquence, affichable ou non, de caractères ASCII. Les chaînes de caractère permettent d'avoir une plateforme indépendante du système pour transmettre des informations et des données. Parmi les applications les plus courantes des chaînes de caractères, on trouve :

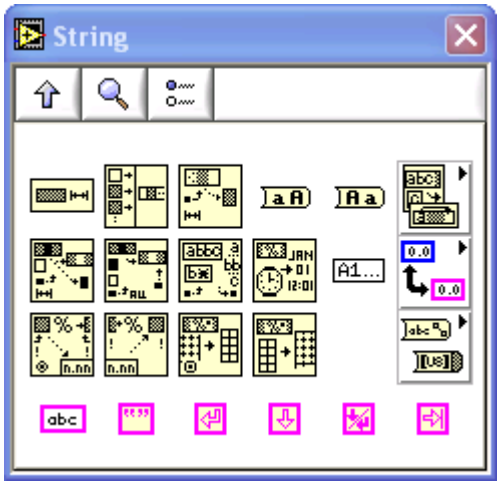
• Créer un texte simple.

• Envoyer des données numériques sous forme de chaînes de caractères aux instruments ou à d 'autres ordinateurs et ensuite les convertir en valeurs numériques.

• Sauvegarder des données numériques sur le disque. Pour stocker des données numériques dans un fichier au format ASCII, vous devez d'abord convertir les données numériques sous forme de chaîne de caractères avant de les écrire dans un fichier sur le disque.

• Donner des instructions ou des suggestions à l'utilisateur par l'intermédiaire de boîtes de dialogue.

Sur la face avant, les chaînes de caractères apparaissent comme des tableaux, des boîtes d'entrée de texte et des étiquettes.
Vous pouvez éditer et manipuler les chaînes de caractères à l'aide de fonctions de chaînes de caractères sur le diagramme de face arrière. Vous pouvez mettre en forme les chaînes de caractères pour les utiliser dans d'autres applications telles ou bien pour les utiliser dans d'autres VIs.



Utilisez les fonctions de chaînes de caractères localisées dans la palette **Fonctions»Chaines de caractères** afin d'éditer des chaînes de caractères de la manière présentée ci-dessous:

• Rechercher et remplacer des caractères ou bien de bouts de chaînes à l'intérieur d'une chaîne de caractères.

• Faire passer le texte d'une chaine de caractères en majuscule ou en minuscule.

• Trouver les éléments recherchés dans une chaîne de caractères.

• Trouver une ligne d'une chaîne de caractères.

• Faire une rotation ou renverser du texte dans une chaîne de caractères.

• Concaténer deux ou plusieurs chaînes de caractères.

• Effacer des caractères d'une chaîne de caractères.

Afin d'utiliser des données dans un autre VI, une fonction ou bien une application, vous devez souvent convertir les données dans une chaîne de caractères et ensuite formater la chaîne de caractère de telle façon que le VI, la fonction ou encore l'application soient capables de les lire. Par exemple, Microsoft Excel utilise des chaînes de caractères qui comprennent des séparateurs pour séparer les nombres et les mots dans différentes cellules.

Par exemple, pour écrire un tableau à 1 dimension de valeurs numériques dans une feuille de calcul en utilisant la fonction **Ecrire un Fichier**, vous devrez formater le tableau sous la forme d'une chaîne de caractères et séparer chaque valeur numérique avec un séparateur comme par exemple une tabulation. Pour écrire un tableau de valeurs numériques dans une feuille de calcul en utilisant le VI **Ecrire dans n fichier tableur**, vous devrez formater le tableau avec la fonction **Tableau en chaine au format tableur** et spécifier le format et le séparateur de données.

Utilisez les fonctions de chaînes de caractères localisées dans la palette **Fonctions»Chaines** pour effectuer les tâches suivantes :

• Concaténer plusieurs chaînes de caractères.

• Extraire une partie d'une chaîne de caractères.

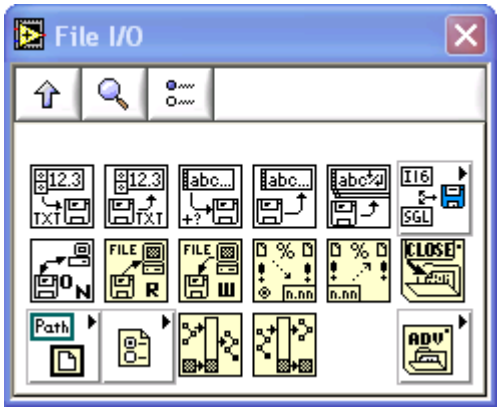
• Convertir des données sous la forme d'une chaîne de caractères.

• Formater une chaîne de caractères pour l'utiliser dans des applications de traitement de texte ou bien des tableurs.

Utilisez les VI **E/S sur fichiers**et les fonctions localisées dans la palette **Fonctions»E/S sur fichiers** pour sauvegarder des chaînes de caractères dans des fichiers texte ou tableur.

## 4.2 File I/O

**Introduction**




Les opérations sur les fichiers (File I/O) envoient et reçoivent des données d'un fichier. Utilisez les VIs File I/O et les fonctions localisées dans la palette **Functions»File I/O** pour mettre en œuvre tous les aspects de a gestion de fichiers dont quelques unes sont décrites ci-dessous :

• Ouvrir et fermer des fichiers de données

• Lire et écrire des données dans un fichier

• Lire et écrire dans des fichiers formatés au format feuille de calcul.

• Déplacer et renommer des fichiers ou des répertoires.

• Changer les caractéristiques des fichiers.

• Créer, modifier et lire un fichier de configuration.

Utilisez les VIs de haut niveau pour réaliser les opérations standard d'entrée/sortie sur les fichiers. Utilisez les VIs de bas niveau et les fonctions pour contrôler individuellement chaque opération de lecture/écriture dans un fichier.

**Bases de la lecture/écriture dans les fichiers**
Une application classique de lecture/écriture dans un fichier met en oeuvre la procédure suivante :

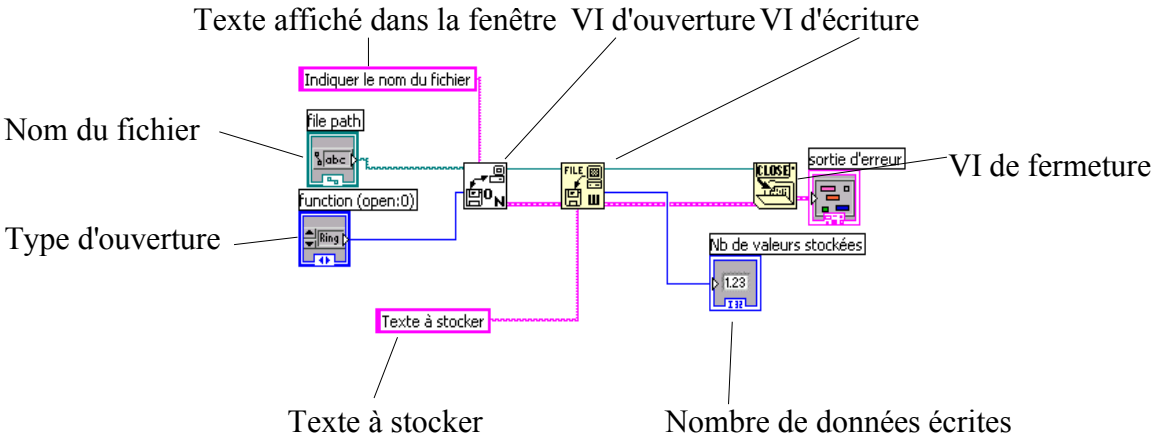
1. Créer et ouvrir un fichier. Indiquer l'emplacement d'un fichier existant ou bien l'endroit où vous voulez créer un nouveau fichier en spécifiant le chemin ou en répondant à une boîte de dialogue afin de diriger LabVIEW vers la localisation du fichier. Après l'ouverture du fichier, un numéro de référence le représente.

2. Lire ou écrire dans un fichier.

3. Fermer le fichier.

La plupart des VI de gestion de fichiers en lecture/écriture et la plupart des fonctions n'exécutent qu'une étape dans l'opération de lecture/écriture dans un fichier. Cependant, certains VI de haut niveau sont conçus pour réaliser les trois étapes. Bien que ces VIs sont souvent moins efficaces que le foncions de plus bas niveaux, vous les préférerez pour leur plus grande simplicité d'utilisation.

L'exemple ci-dessous vous montre l'ouverture d'un fichier et le stockage d'une chaine de caractères.

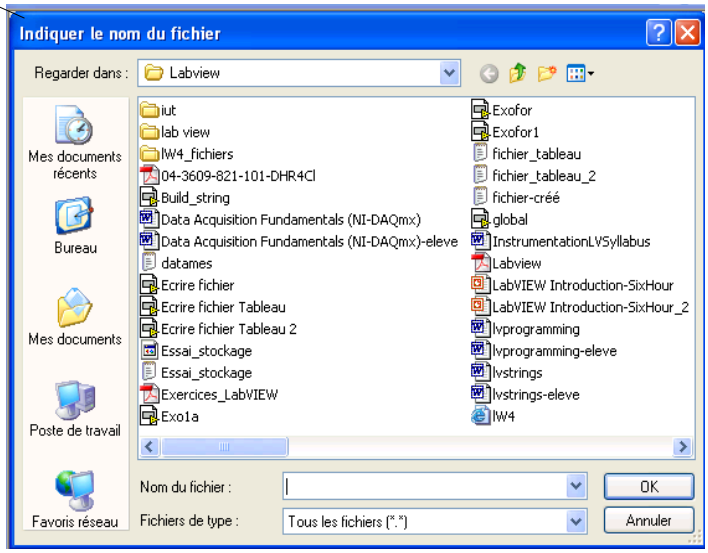



Différents types d'ouvertures sont possibles :

- Ouvrir
- Ouvrir ou créer
- Créer ou remplacer
- Créer
- Ouvrir en lecture seule

Si le nom du fichier n'est pas spécifié, une fenêtre Windows s'ouvrira.

Texte indiqué ci-dessus



**Choisir un format de gestion de fichier en lecture/écriture**

Les VIs « E/S sur fichier » que vous utilisez dépendent du format des fichiers. Trois formats de lecture/écriture sont possibles : texte, binaire ou datalog.

Le format que vous allez utiliser dépend de la donnée que vous allez acquérir ou créer et de l'application qui va accéder aux données.

Utilisez le guide de choix suivant pour déterminer le format à utiliser:

• Si vous voulez que vos données soient accessibles dans d'autres applications comme Microsoft Excel par exemple, utilisez des fichiers Texte car ils sont les plus communs et les plus portables.
• Si vous avez besoin d'accéder de manière aléatoire en lecture ou en écriture ou bien si la vitesse et l'espace sur le disque sont des critères importants utilisez des fichiers binaires car ils sont plus efficaces que les fichiers texte pour ce qui concerne l'espace disque et la vitesse d'accès.
• Si vous voulez manipuler des enregistrements de données complexes ou différents types de données dans LabVIEW, utilisez des fichiers de type « datalog » car ils représentent le meilleur stockage de données si vous ne voulez y accéder que par l'intermédiaire e LabVIEW.
**Utilisation des VIs haut niveau de lecture/écriture de fichiers.**
Utilisez le VI de haut niveau situé dans la ligne du haut de la palette **Fonctions»E/S sur fichiers** pour effectuer les opérations de lecture/écriture simples sur des données du type suivant :

• Caractères envoyés ou provenant d'un fichier texte.

• Lignes venant d'un fichier texte.

• Tableaux à une ou deux dimensions de données numériques simple précision envoyés ou reçu à un fichier de type feuille de données.

• Tableaux à une ou deux dimensions de nombre simple précision ou d'entiers signés sur 16 bits envoyés ou provenant d'un fichier binaire.

Vous pouvez gagner du temps et des efforts de programmation en utilisant les VIs haut niveau pour écrire et lire des fichiers. Les VIs de haut niveau permettent des opérations de lecture et d'écriture en plus de l'ouverture et de la fermeture des fichiers. Evitez de placer ces VIs de haut niveau dans des boucles car le VIs ouvrira et fermera le fichier à chaque fois qu'il sera lancé.

Les VIs de haut niveau attendent un chemin pour accéder aux fichiers. Si vous ne connectez pas de chemin, une boîte de dialogue apparaît pour vous permettre de spécifier le fichier à utiliser. Si une erreur apparaît, les VIs de haut niveau font apparaître une boîte de dialogue qui décrit l'erreur. Vous pourrez alors choisir d'arrêter l'exécution ou bien de la continuer.

Utilisez les VIs de gestion de fichiers binaires situés dans la palette **Fonctions»E/S sur fichiers»Fichiers binaires** pour lire ou écrire des fichiers dans le format binaire. Les données peuvent être des entiers ou des flottants en simple précision.

## *4.3 Property Nodes*


Utilisez les Property Node pour obtenir ou donner des propriétés à une application ou un VI. Sélectionnez la propriétée du noeud en utilisant l'outil de travail et en cliquant sur le terminal property ou en faisant un click droit sur la partie blanche du nœud et en sélectionnant dans le menu **Properties**.

Vous pouvez lire ou écrire différentes propriétés en utilisant un simple noeud.(You can read or write multiple properties using a single node.) Utilisez l'outil de positionnement pour redimensionner le Property Node et lui ajouter de nouveaux terminaux. Une petite flèche vers la droite indique la propriété que vous lisez.
Une petite flèche vers la gauche, indique une propriété que vous avez écrite. Faites un click droit sur la propriété et sélectionnez **Change to Read** ou **Change to Write** du menu pour changer l'opération de la propriété.

Le noeud s'exécute du haut vers le bas. Le Property Node ne s'exécute pas si une erreur apparaît avant son exécution, donc faites attention à la possibilité de voir apparaître des erreurs. Si une erreur apparaît dans une propriété, LabVIEW ignore les propriétés restantes et renvoie une erreur. Le cluster **error out** contient la propriété qui a causé l'erreur.

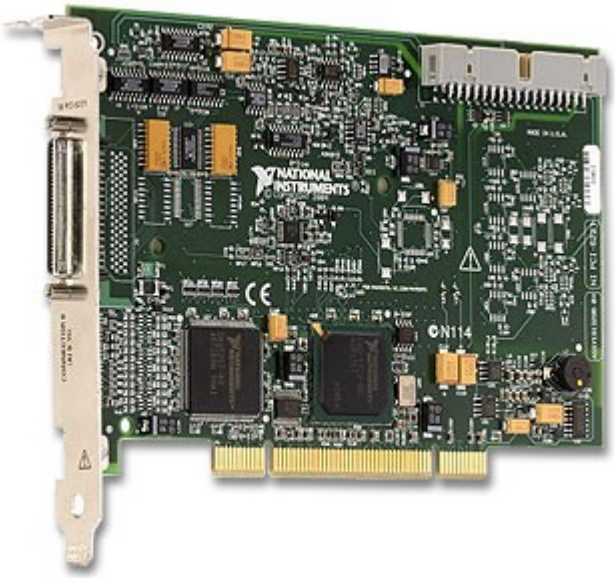
**Property Nodes connectés de façon implicite**

Quand vous créez un Property Node d'un objet de la face avant en faisant un click droit sur l'objet et en sélectionnant dans le menu **Create»Property Node**, LabVIEW crée un Property Node sur le diagramme de face arrière qui est implicitement connecté avec l'objet sur la face avant. Parce que c'est Property Nodes sont implicitement connectés à l'objet à partir duquel il a été crée, ils ne disposent pas d'entrée **refnum**, et vous n'avez pas à connecter le Property Node au terminal de l'objet de la face avant ou le contrôle refnum.

# 5 Acquisition de données avec Labview

## *5.1 Introduction*

Labview est un logiciel spécialisé dans l'acquisition et l'affichage des données. National Instrument fabrique donc un ensemble de cartes d'acquisitions (avec des E/S analogiques et numériques). Il existe un grand nombre d'autres fabricants de cartes d'acquisitions comme Keithley par exemple.
Dans notre cours, nous allons mettre en oeuvre une carte National Instrument PCI série M 6221. Celle-ci s'implante dans un connecteur d'extension PCI libre du PC.



Les caractéristiques de cette carte sont les suivantes :

- 16 entrées analogiques de 16 bits avec 250 k échantillons/s
- 2 sorties analogiques de 16 bits capables de générer 833 k échantillons/s.
- 24 E/S tout ou rien
- 2 compteurs 32 bits

Actuellement, on voit de plus en plus apparaître des cartes d'acquisitions externes au PC et qui utilisent une connexion USB 2.0 pour envoyer les données à celui-ci. Ceci est possible grâce à l'important débit de données de ce type de bus.

Après avoir installé la carte dans le PC, il faudra dans un premier temps s'assurer que la carte est convenablement installée et reconnue. Pour cela Labview propose un outil : l'explorateur « Measurement & Automation » qui est présent sur le bureau.

## 5.2 L'explorateur « Measurement & Automation »

L'explorateur "Measurement & Automation" (ou MAX) est un logiciel qui donne accès à l'ensemble des composants National Instrument implantés dans l'ordinateur. Les fonctions de MAX sont réparties selon 5 catégories :

- Voisinage de données
- Périphériques et Interfaces
- Echelles
- Logiciels
- Drivers IVI pour les systèmes déportés

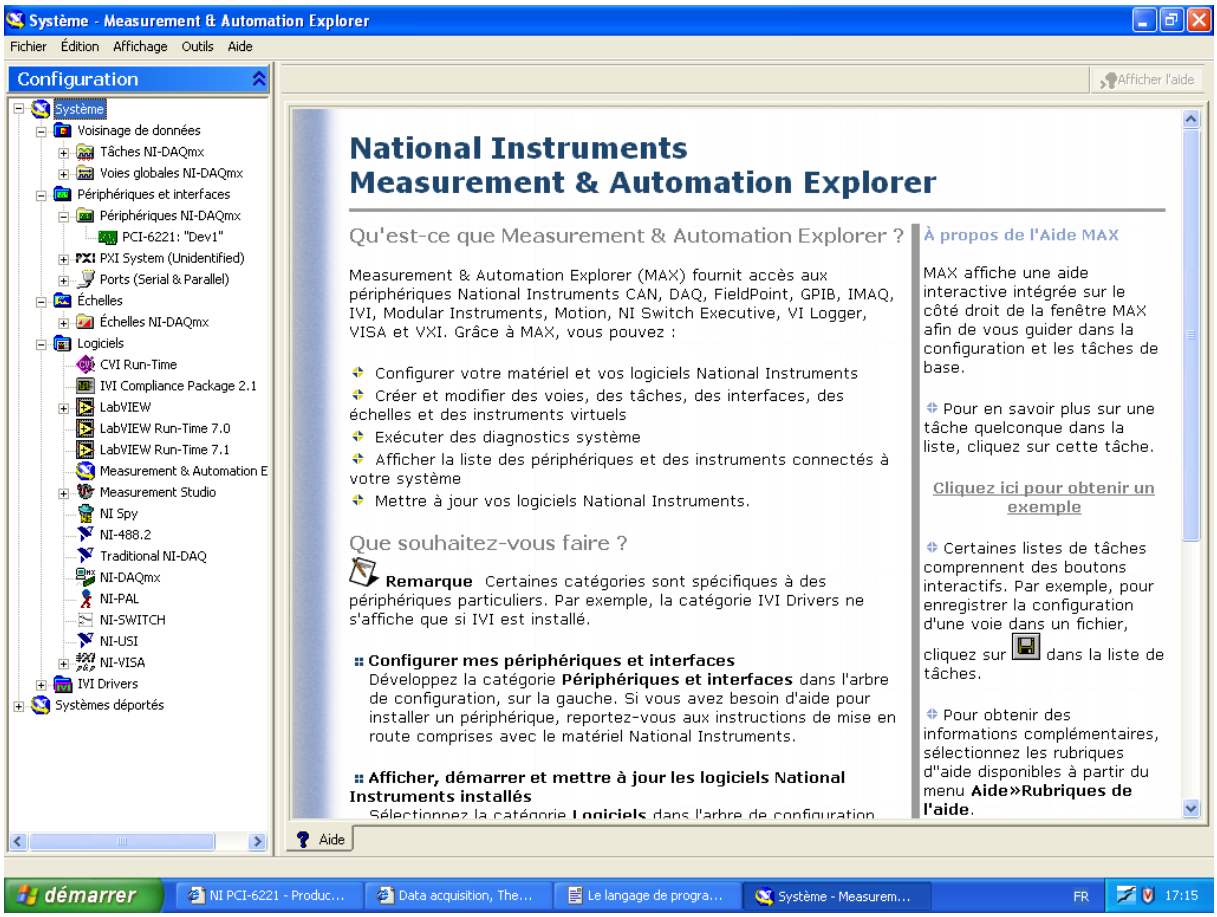


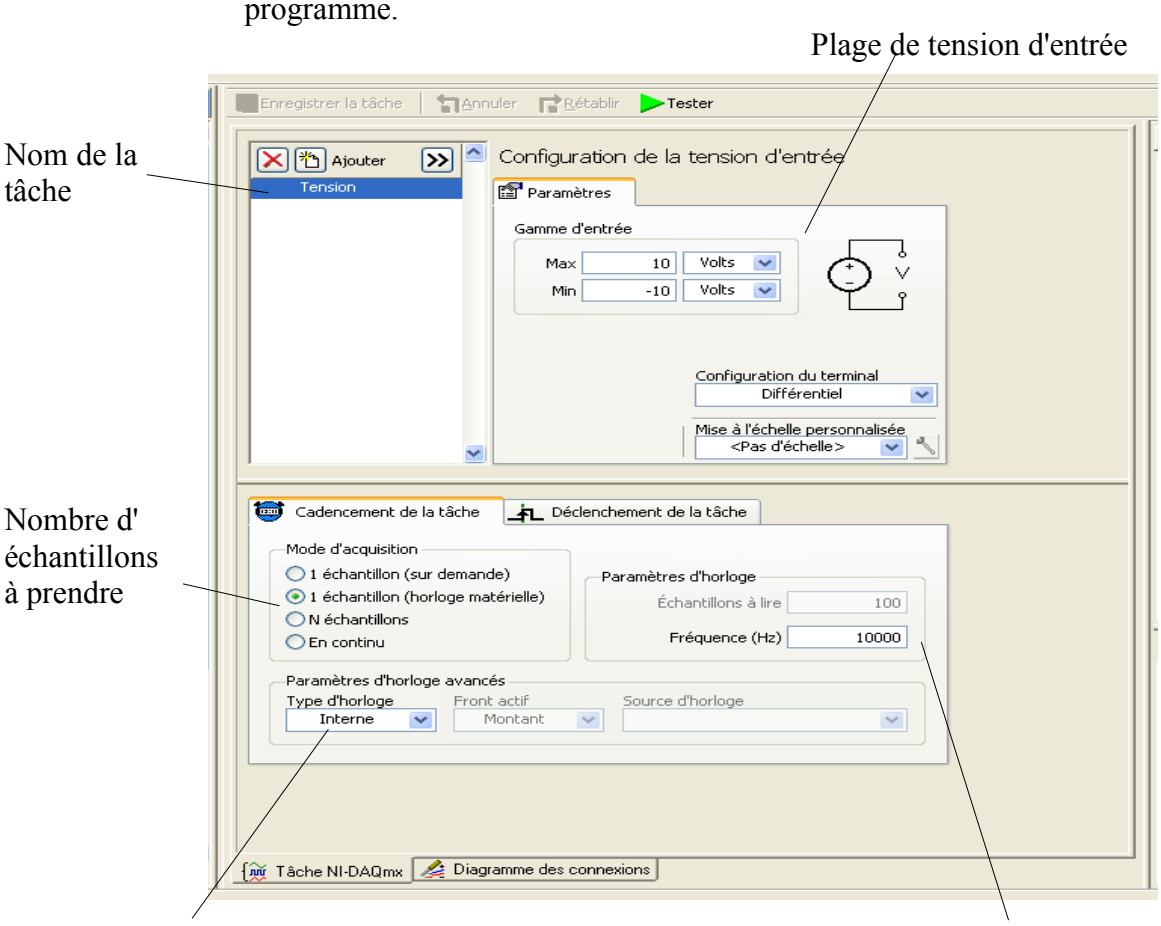

Figure 1. Explorateur « Measurement and Automation »

## 1) Voisinage de données

Voisinage de données contient l'ensemble des tâches DAQmx qui sont configurées. Ce sont les objets qui permettent à LabVIEW de communiquer avec les circuits DAQ implantés dans le PC. Ils peuvent être aussi bien configurés dans LabVIEW que dans MAX. De plus, il met à disposition des composants permettant de reconfigurer et tester les composants installés, ainsi que des outils permettant de créer de nouveaux composants.

### a) Tâches DAQmx

LabVIEW 7 et NI-DAQmx sont centrés sur la création de tâches. Une tâche peut aussi bien être créée par LabVIEW ou par MAX et contient les éléments nécessaires pour l'acquisition des données comme le canal (c'est à dire la voie que l'on va convertir), l'échelle (min – max de la tension d'entrée) ou le temps d'échantillonnage par exemple. En utilisant un composant, la même donnée peut être récupérée dans plusieurs programmes différents sans être obligé de reconfigurer l'acquisition à chaque fois. En plus, le contrôle des composants permet aux utilisateurs d'utiliser plusieurs composants différents dans le même VI LabVIEW ce qui permet d'acquérir rapidement différents types de signaux dans le même programme.



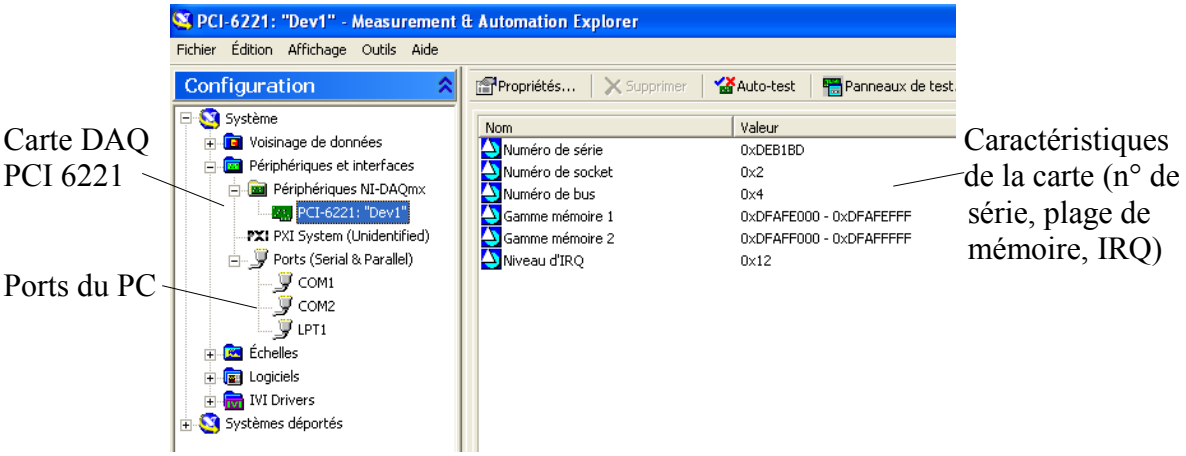
### b) Créer un nouveau composant DAQmx

Lorsqu'il crée un nouveau composant DAQmx, l'utilisateur doit indiquer un certain nombre d'options : *entrée analogique, sortie analogique, entrée compteur, sortie compteur et E/S numériques.* Après avoir choisit la catégorie de signal qu'il veut acquérir, l'utilisateur doit sélectionner le type de mesure spécifique qu'il veut effectuer dans une liste. Il choisit ensuite l'emplacement physique où le signal sera appliqué. Enfin, il lui donne un nom pour s'en souvenir. Les noms des tâches sont les mêmes entre MAX et LabVIEW et leur noms sont plus faciles à retenir que des nombres.

Une fois que le composant a un nom, sa création est terminée. La fenêtre de configuration apparaît pour que l'utilisateur puisse spécifier les paramètres du composant. Les options tels que l'échelle, l'amplitude d'entrée, le temps sont affichées et peuvent être modifiées. La fenêtre de configuration est présentée page précédente.

## 2) Périphériques et interfaces

Périphériques et Interfaces affiche les composants National Instruments qui ont été détectés dans le PC. Il inclus également des utilitaires pour tester les équipements. Les trois utilitaires pour les cartes DAQ sont : Propriétés, Auto Test, and Panneau de Test.




### a) Propriétés

C'est un utilitaire qui permet de configurer les cartes DAQ. A son lancement, une boite de dialogue apparaît avec les lignes suivantes permettant de configurer la carte :

- Accessoires – Dans ce menu, l'utilisateur peut configurer les accessoires attachés à la carte DAQ. Par exemple, pour nous l'accessoire est le bornier BNC 2120.
- RTSI Configuration

**b) Auto Test**

Ce module permet de tester si MAX est capable de communiquer avec la carte. C'est le niveau le plus simple de test.

**c) Panneau de Test**

C'est un utilitaire pour tester les entrées analogiques, sorties analogiques, E/S numériques. et les fonctionnalités du compteur du composant DAQ. Il est utile pour tester en cas de problèmes car il permet de tester directement la carte par NI-DAQ. Si la carte ne fonctionne pas avec le Panneau de Test, elle ne fonctionnera pas avec LabVIEW. **Scal**

**3) Echelles**

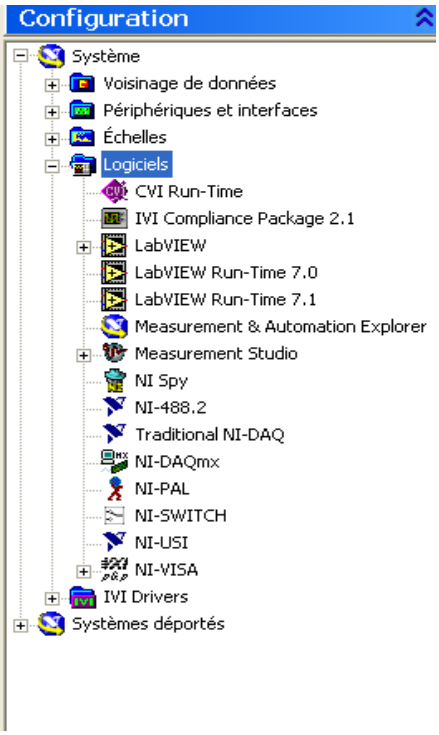
Echelles est un utilitaire qui permet de déterminer des informations d'échelle pour des canaux virtuels. Chaque échelle personnalisée possède son nom et sa description pour faciliter l'identification. Cela permet par exemple d'avoir une échelle qui n'est pas linéaire.

Une échelle personnalisée peut être de 4 types : linéaire, map ranges, polynomial ou table.

- **Linéaire** – Echelle utilisant la formule $y = mx + b$.
- **Correspondance de gamme** – On fixe la valeur maximale et minimale de l'échelle avant et après mise en forme..
- **Polynomial** – Echelle utilisant la formule $y = a0 + (a1 * x) + (a2 * x2) + \ldots + (an* xn)$.
- **Table** – C'est une échelle dans laquelle on définit les valeurs d'entrées et la valeur de sortie adéquate dans un tableau.

**4) Logiciels**

Logiciels montre tous les logiciels National Instruments installés. Les icônes permettent de lancer chaque logiciel. Vous avez aussi Software Update Wizard qui permet de tester si la version du logiciel est la dernière, dans le cas contraire Software Update Agent ouvre une page Web permettant de télécharger la dernière version.




## 5) Drivers IVI

Les Drivers IVI permettent d'utiliser des programmes écrit pour un type de carte avec un autre type en utilisant le même jeu de drivers.

## 5.3 Mise en oeuvre du DAQ avec LabVIEW 7

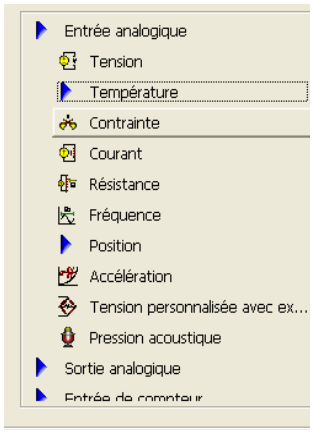
Il existe différentes façon de mettre en oeuvre les entrées-sorties sous Labview.Nous allons voir différentes solutions. Elles présentent chacunes leurs avantages mais aussi leurs inconvénients.

### 1) L'assistant DAQ

Le VI Express Assistant DAQ de LabVIEW 7 permet de configurer pas à pas le composant d'acquisition de données. Il permet facilement et rapidement de configurer les entrées/sorties. Il permet également de tester la configuration avant de la mettre en oeuvre dans le programme et de tester si elle est correcte.
Vous le trouverez dans le menu du diagramme dans **Entrées**.
Une fois posé sur le diagramme une fenêtre de configuration va s'ouvrir.
Vous aurez le choix entre plusieurs types d'acquisitions:

- Entrée analogique
- Sortie analogiques
- Entrée compteur
- Sortie compteur
- E/S numériques

En fonction des options que vous choisirez, vous aurez différentes possibilités de choix. Par exemple pour l'entrée analogique :




Il faut ensuite sélectionner la voie sur laquelle on a connecté le signal à acquérir. Puis cliquer sur **Terminer**.

Un panneau de configuration de tâche, identique à celui que l'on a vu dans MAX va s'ouvrir.
Plusieurs options sont à configurer.

- La gamme d'entrée de la tension à mesurer
- Si on souhaite utiliser une échelle définie dans MAX.
- Le nombre d'échantillons à acquérir. La configuration de ce paramètre aura une influence importante sur le VI.

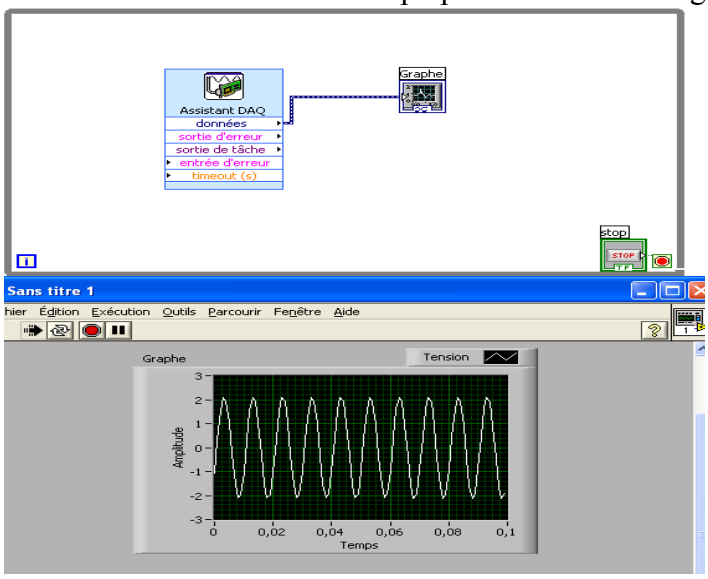
Différents choix sont possibles :

- 1 échantillon (sur demande) : A chaque fois que Labview lancera la fonction DAQ, il récuperera une valeur. L'inconvénient est que l'on ne sait pas quel est l'intervalle de temps entre deux échantillons et surtout si cet intervalle de temps restera fixe.
- 1 échantillon (horloge matérielle) : Ici, les échantillons seront pris à des instants donnés. On peut d'ailleurs fixer la fréquence d'échantillonnage. L'avantage est que l'on maitrise parfaitement le temps entre deux échantillons. Par contre, il faudra s'assurer que Labview aura le temps de traiter les informations. Sans quoi, une erreur sera générée
- N échantillons : La tâche acquérira N échantillons à une fréquence donnée. Une fois que la tâche est accomplie, elle renvoit en sortie les échantillons sous forme d'un tableau.
- En continu : La tâche va acquérir des échantillons à la fréquence qu'on lui a fixé jusqu'à ce qu'elle s'arrête.

Une fois configuré, on voit apparaître un VI de type VI Express avec un certain nombre d'entrées et de sortie.

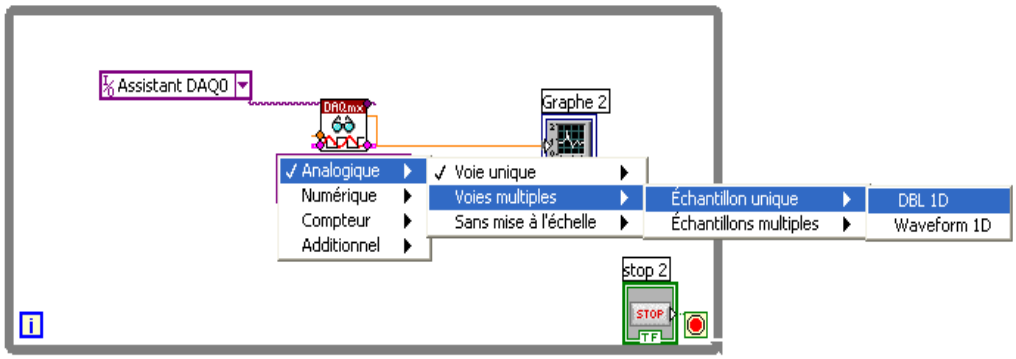

On voit notamment en sortie, une sortie d'erreur si la tâche s'est mal exécutée, les données. En entrée, on pourra avoir : un timeout au delà duquel on estime que la tâche a échouée et une entrée STOP puisque ici on est configuré en acquisition continue. Dans les autres modes d'acquisition cette entrée n'existe pas.

Il suffit ensuite de connecter un Graph pour visualiser le signal reçu.



### 2) Les VI DAQmx

Même si l'Assistant DAQ est facile à utiliser, il nécessite plus de ressources système que les fonctions standard DAQmx. Pour des situations où la vitesse d'exécution est critique, celles-ci sont préférables. Dans les versions précédentes de LabVIEW, il existait différents niveaux de VI avec des niveaux de difficultés variables en fonction des compétences des utilisateurs. Maintenant, ils ont été intégrés dans des VIs « polymorphes » pour intégrer les fonctionnalités et faciliter la mise en oeuvre. L'idée qui se cache derrière eux est d'utiliser un VI unique de fonctions haut niveau comme lire, écrire, débuter l'acquisition, l'arrêter et de le configurer pour le type de donnée spécifique. Grâce à cela, l'utilisateur utilise toujours la même fonction quelque soit ce qu'il veut faire et ne risque pas de se tromper entre les différents Vis possibles. Un exemple de VI Polymorphe est donné Figure suivante.




VI Polymorphe

Ci-dessus, le VI **Lire tâche** est placé sur la face arrière et est configuré. Le menu permet facilement de changer la configuration. De plus, ces VIs peuvent être facilement changés si les types de données changent. L'autre point intéressant est que l'on a à la fin le format de la donnée en sortie. Si on acquiert un échantillon sur plusieurs voies, on aura bien un tableau qui sera de type double.

## *5.4 Utilisation des données*

Une fois les données récupérées, il est possible de les sauvegarder dans un fichier en utilisant les fonctions présentées précédemment.
On peut également les envoyer via des lignes de communication (série, Ethernet, Mail, ...)
Enfin, on dispose d'un ensemble d'outils de traitement du signal regroupés dans le menu **Analyse**.

# 6 Communication avec Labview

## *6.1 Principes généraux*

Labview permet de gérer une grande quantité de types de communication différents. Vous disposez notamment de VIs permettant de gérer les liaisons séries ou parallèles, mais vous pouvez également communiquer via Ethernet TCP/IP, envoyer des mails ou encore gérer une communication IEEE 488 (c'est à dire GPIB).
Le bus GPIB est un bus spécialement développé pour la communication avec des instruments de mesure (oscilloscope, GBF, multimètre,...). Ceci n'est possible que si l'instrument est pourvu de ce type de connecteur et que le PC est équipé d'une carte spécifique GPIB.

Le principe de fonctionnement de ces communicatios est toujours le même. Vous devez :

- Ouvrir une communication
- Ecrire et ou Lire les données
- Fermer la communication

En fonction du type de communication que vous choisirez (série, Ethernet, ...), il faudra configurer les bons paramètres. Il s'agit ici de la configuration physique.

Liaison série :

- Vitesse de transmission
- Nb. de bits de données
- Nb. de bits Stop
- Parité
- Port de communication

Liaison Ethernet :

- Adresse IP du serveur
- Port de communication du serveur

Envoie de mail :

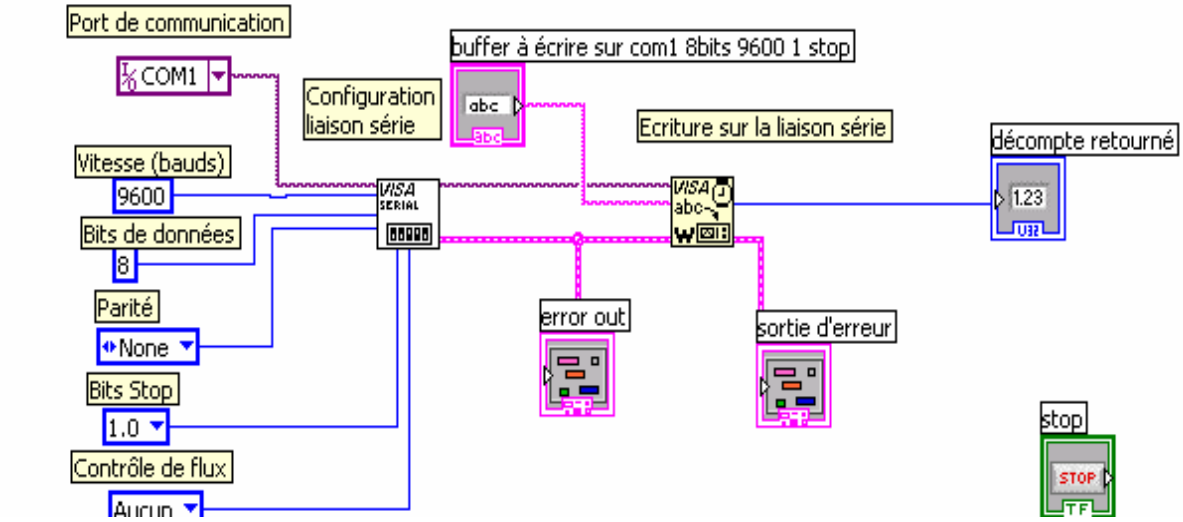
- Adresse du serveur mail SMTP
- Adresse mail du destinataire

Il faut ensuite envoyer les données selon un format adéquate. Ceci dépendra du type de protocole de communication choisit et donc générer les trames d'envoie nécessaire. Cela peut être utile si vous utilisez un protocole de type MODBUS par exemple (que vous allez utiliser en Automatismes Industriels) et qui peut fonctionner aussi bien sur une liaison série que sur une liaison Ethernet.

Labview permet également de mettre en oeuvre des communications de type USB. Ceci se faisant en utilisant les 'dll' appropriés qui doivent être fournis par le fabricant de composant USB ou alors écrits par vous.

## 6.2 La liaison série

La mise en oeuvre de la liaison série est identique à celle du PIC. Vous aurez à configurer les paramètres de communication afin que les deux équipements reliés entre eux puissent dialoguer.




Vous aurez notamment à configurer (le port de communication, la vitesse, le nombre de bits de données, la parité, le nombre de bits Stop et s'il y a un contrôle de flux).
L'ensemble des VIs qui gèrent la liaison série sont dans le sous-menu :

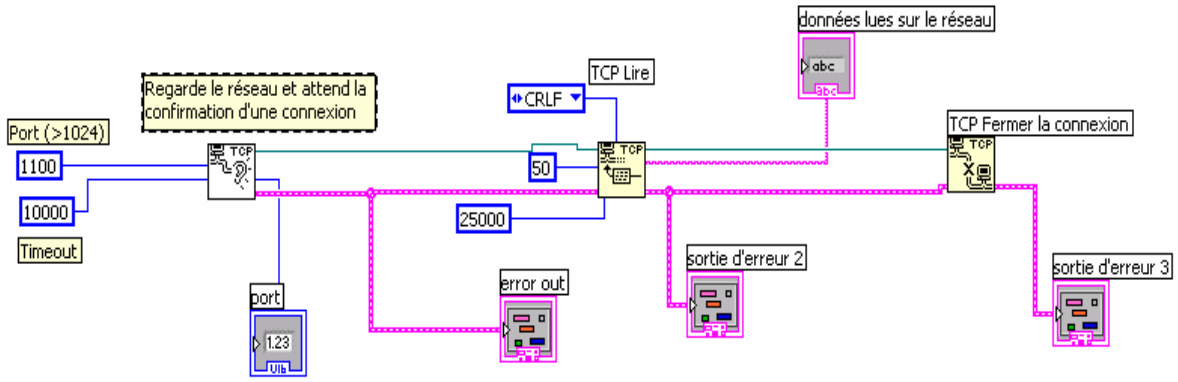
**E/S Instruments >> Série**

## 6.3 La liaison Ethernet TCP/IP

Nous sommes ici dans une configuration de type client/serveur. Ceci veut dire que si l'on souhaite communiquer entre deux équipements, l'un d'eux devra être le serveur et l'autre un client qui se connectera au serveur.
Donc les VIs du client et du serveur ne seront pas les mêmes.

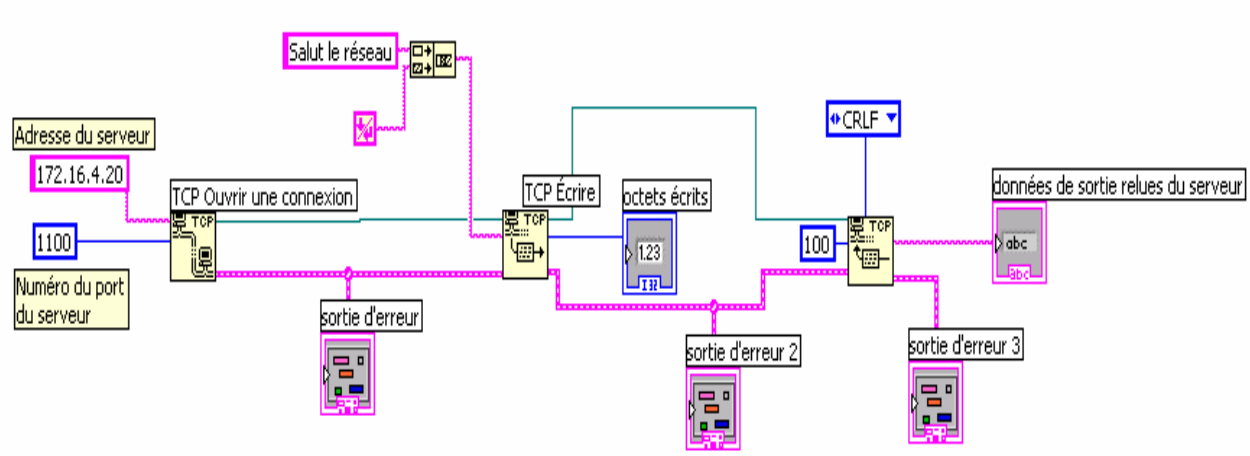
Le serveur devra :

- Ouvrir une communication et attendre qu'un client vienne se connecter. Ne connaissant pas l'adresse du client, la seule chose qu'on puisse donner c'est le numéro du port de communication au niveau du serveur.



Le client devra ouvrir une communication en indiquant :

- Sur quel serveur il veut se connecter (en indiquant son adresse IP)
- Le numéro de port que l'application utilise sur le serveur distant



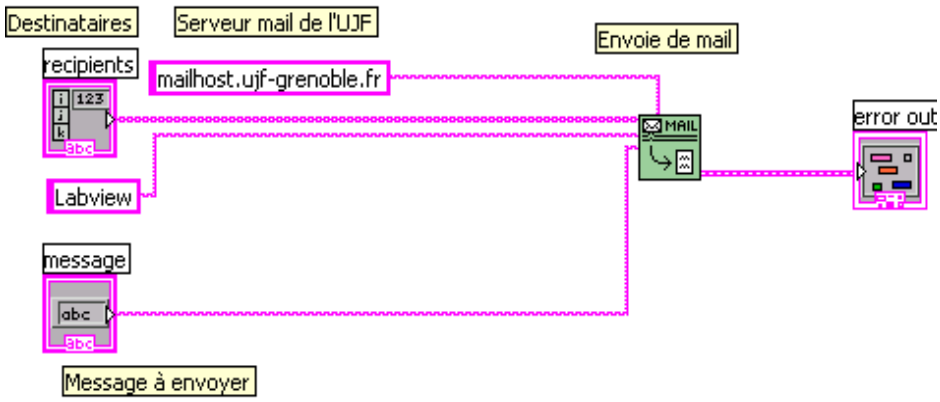

Les VIs qui permettent de gérer la liaison Ethernet se trouvent dans le menu :

**Communication >> TCP**

Il est donc nécessaire d'avoir une chronologie entre le lancement des 2 VIs. Le VI serveur devra toujours être lancé en premier. Si vous faites l'inverse, le VI client ne trouvera pas de VI serveur en état de marche.
De plus, dans le cas du VI serveur, il faudra penser à fixer un délai de communication maximum au cas où personne ne se connecterait. Il s'agit du Timeout.

## *6.4 Envoie de mails par Labview*

L'envoie de mail est une solution de plus en plus utilisée pour indiquer qu'une tâche est terminée ou encore pour envoyer des informations. Evidemment, cette solution n'est pas envisageable si l'on souhaite une réponse rapide. Le temps de transfert des mails étant assez long et souvent aléatoire.



Pour pouvoir communiquer, il faut indiquer l'adresse du serveur mail qui va réceptionner le message et le retransmettre plus loin (c'est souvent un serveur de type SMTP pour « Simple Mail Transfer Protocol »)
Pour que le mail puisse être acheminé, il est nécessaire de lui indiquer l'adresse mail du destinataire.
Si on n'indique pas d'adresse mail d'expéditeur, le serveur mail ajoutera la sienne du type :

**MAILER-DAEMON@ tana2.ujf-grenoble.fr**

## *6.5 Conclusion*

Il est relativement facile de communiquer grâce à Labview avec d'autres systèmes. En effet, s'il fallait configurer les périphériques du PC, cela pourrait poser un certain nombre de problèmes et serait assez compliqué.
Néanmoins, il ne faut pas oublier que ces blocs sont des macro-fonctions. IL peut parfois arriver que leur fonctionnement interne puisse générer des problèmes.

Labview permet de configurer la partie matérielle de l'envoie d'information. Si vous souhaitez utiliser des protocole de transmission particuliers (par exemple MODBUS, CAN, etc...) c'est à vous de générer les trames de communication (en-tête, code fonction, code de détection d'erreur,... ) propres à ces types de liaisons et ensuite de les envoyer via le support (liaison série, ethernet,...).