Université Paris-Sud

Licence et magistère de physique fondamentale

PROJETS EXPÉRIMENTAUX DE PHYSIQUE

Manuel d'initiation à LabView

http://hebergement.u-psud.fr/projets

Le logiciel LabVIEW est une "Plate-forme Expérimentale d'Instruments Virtuels pour Laboratoire" (***Lab****oratory* ***V****irtual* ***I****nstrument* ***E****ngineering* ***W****orkbench*). C'est un environnement de programmation disponible sur plusieurs systèmes d'exploitation commercialisé par la société "National Instruments" et possédant un langage graphique (le langage G), des bibliothèques de fonctions et de sous-programmes, ainsi que des outils de développement.

Nous décrirons ici quelques caractéristiques de LabVIEW 8.0 pour Windows, version du logiciel installée sur nos micro-ordinateurs PC sous système d'exploitation Windows XP.

La langue « naturelle » de LabVIEW étant l'anglais, de nombreux termes de ce manuel sont laissés en anglais. Ils sont (sauf omissions) indiqués en italique. Toutes les indications pouvant être obtenues de l'aide en ligne du logiciel ont également été laissées en anglais.

Certaines parties du texte sont écrites dans une police plus petite que le reste du manuel. Elles doivent être omises lors d'une première lecture, mais leur compréhension ultérieure vous simplifiera la vie …

Ce manuel est surtout destiné à servir d'aide à l'écriture des programmes avec LabVIEW. Certains points sont donc détaillés à plusieurs endroits pour la cohérence du contexte local. Si vous les remarquez, cela signifiera que vous avez tout lu !!

Enfin, il reste nécessairement des erreurs, omissions, etc... N'hésitez pas à les signaler, cela aidera à l'amélioration de ce manuel pour les générations futures.

1 Introduction à LabVIEW

1.1 Les fenêtres de LabVIEW

Le logiciel LabVIEW, grâce à son environnement à multiples fenêtres, permet de réaliser et d'exécuter rapidement des programmes simples, comparables aux fonctions d'un langage conventionnel, mais pouvant prendre à l'écran l'apparence d'un appareil de mesure, d'où leur appellation **Instrument Virtuel** (**V**irtuel **I**nstrument en anglais, ou « **VI** »).

Le langage de programmation employé par LabVIEW est le langage graphique G. Ce langage est assimilable aux langages orientés objet, tel le C++, offrant des classes de données ayant des attributs spécifiques, ainsi que des opérateurs et des fonctions polymorphes agissant sur les données.

Tout programme exécutable construit en langage G comporte une interface utilisateur et un programme graphique situés dans deux fenêtres distinctes (**panneau avant** et **diagramme**). L'environnement LabVIEW offre, dans trois autres fenêtres distinctes, des palettes indépendantes d'outils et d'objets permettant d'éditer les deux fenêtres du programme et de tester son fonctionnement.

1.1.1 Les fenêtres du programme

Un programme de LabVIEW comprend 2 fenêtres distinctes : le **panneau avant** servant d'interface avec l'utilisateur et le **diagramme** contenant le programme source en langage graphique G.

1.1.1.1 Le « panneau avant »

Cette fenêtre, où apparaissent des objets sous forme de commandes d'entrée ou **contrôleurs** (*Controls*) ou d'**indicateurs** de sortie (*Indicators*), constitue l'interface interactive du programme. Le « panneau avant » vide apparaissant par défaut lors de la création d'un programme est indiqué sur la figure ci-dessous.

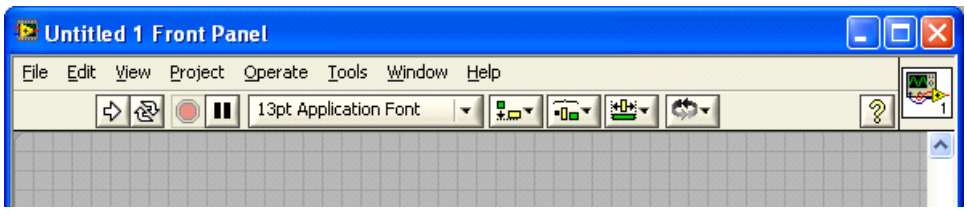

Figure 1-1 : Un « panneau avant » vide

Un programme écrit en langage G sera utilisable de façon interactive dès que son « panneau avant » sera visible.

1.1.1.2 Le diagramme

Cette fenêtre contient le code source graphique représentant le programme écrit en langage G. Le diagramme vide apparaissant par défaut lors de la création d'un programme est indiqué sur la figure ci-dessous.

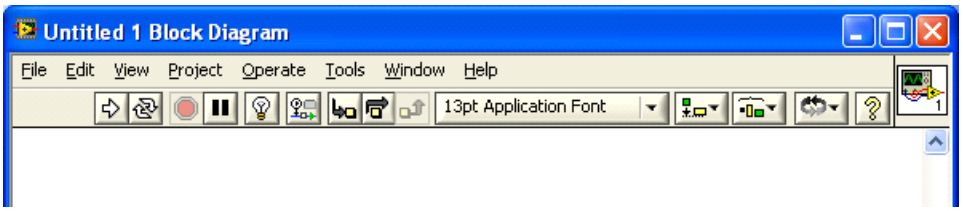

Figure 1-2 : Un diagramme vide

On passe du « panneau avant » au diagramme à l'aide du choix *Show Block Diagram* du menu *Window* de la barre de menus. Réciproquement, on retourne au « panneau avant « à partir du diagramme à l'aide du choix *Show Front Panel* du menu *Window* de la barre de menus.

1.1.1.3 Un 1er exemple

On a représenté sur la figure ci-dessous un exemple simple de programme écrit en langage G. Le but de ce programme est d'ajouter 1 à la valeur fournie par un contrôleur et d'afficher le résultat sur un indicateur.

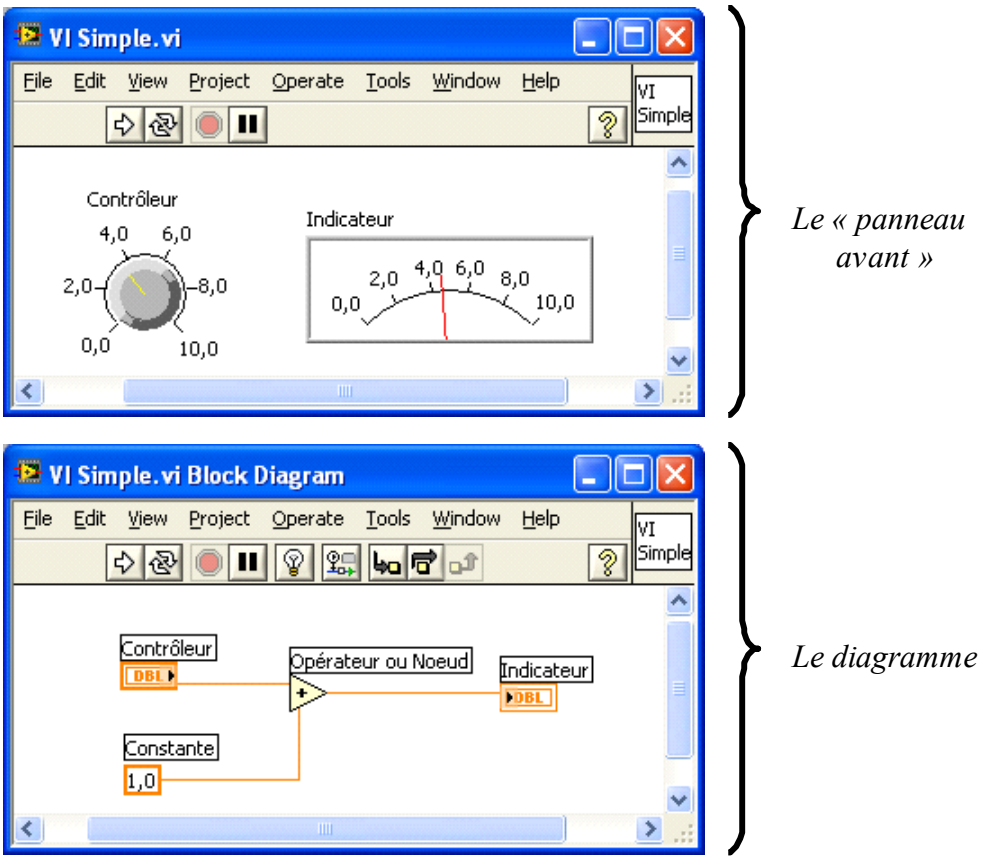

Figure 1-3 : Exemple de programme LabVIEW

On voit sur le « panneau avant » un contrôleur et un indicateur. Sur le diagramme, on observe les **terminaisons** associées au contrôleur et à l'indicateur, reliées entre elles par des **fils**.

Les **terminaisons**, suivant leur nature, correspondent à des entrées ou des sorties de données du programme. Celles qui dépendent des objets du panneau avant (contrôleurs et indicateurs) apparaissent automatiquement sur le diagramme dès que le contrôleur ou l'indicateur est placé sur le « panneau avant ». D'autres terminaisons (dans notre exemple les deux entrées et la sortie de l'opérateur ) ne sont pas reliées à des objets du panneau avant mais à des objets introduits directement sur le diagramme. En langage G, on appellera **nœud** une instruction exécutable, un opérateur, une fonction ou un sous-programme ayant des entrées et/ou des sorties (qui ne sont pas nécessairement reliées au « panneau avant »).

Les **fils** reliant les contrôleurs aux indicateurs ou aux nœuds symbolisent le passage des données entre les éléments du programme. Leur aspect et leur couleur dépendent du type de données qui les parcourent. Par exemple, un fil mince correspond à une donnée simple, un fil épais correspond à un ensemble de données, tableau ou agrégat (cf Figure 3-11). Leur type est défini par celui de la **terminaison source** (dans notre exemple un contrôleur réel). La **terminaison de destination** (dans notre exemple un indicateur réel également) doit être compatible avec ce que transmet le fil. Si ce n'est pas le cas, on crée une erreur et notre programme ne passe pas l'étape de la compilation …

Les terminaisons sont toujours de catégories différentes à chaque extrémité d'un fil : une donnée part d'un contrôleur pour arriver sur un indicateur.

Il est important de noter, que contrairement à l'exécution séquentielle des langages de programmation classiques (Pascal, Fortran, C, …), LabVIEW exécute un programme sous contrôle du flot de données et la programmation doit toujours être faite en fonction de cette dépendance. Un nœud quelconque du diagramme n'est exécuté que lorsque toutes ses données d'entrée sont disponibles, et ses données de sortie ne sont disponibles qu'à la fin de son exécution. Dans notre exemple, le nœud « + » ne peut s'exécuter que lorsque la donnée du contrôleur lui est parvenu. L'affichage sur l'indicateur ne s'effectuera qu'à la fin de l'exécution du nœud  .

1.1.1.4 Les caractères communs au panneau avant et au diagramme

Barre de titre

Située en haut de chaque fenêtre, la barre de titre (cf Figure 1-4) affiche le nom du programme. Ce nom est *Untitled i* (où *i* est un nombre > 0) par défaut à l'ouverture d'un nouveau programme, et le reste tant que le programme n'a pas été sauvegardé.

Barre de menus

Située sous la barre de titre (cf Figure 1-4), elle présente une série de menus déroulant communs aux deux fenêtres. Ces menus à structure hiérarchisée proposent des choix pouvant entraîner une action immédiate, conduire à des sous-menus (▶), ou aboutir à une fenêtre de dialogue (…). Les principales actions déroulant de ces menus sont indiquées ci-dessous.

Menus hiérarchisés communs aux fenêtres du diagramme et du panneau avant :

File	Pour manipuler les fichiers (créer, ouvrir, sauver, imprimer)
Edit	Pour éditer la fenêtre active (copier, coller)
View	Pour voir la hiérarchie des VI en mémoire et rappeler les palettes *Tools* ou *Controls*
Project	Nous ne nous en servirons pas cette année
Operate	Pour choisir le mode de fonctionnement
Tools	Nous ne nous en servirons pas cette année
Window	Pour changer la fenêtre active (« panneau avant »/diagramme)
Help	Pour afficher la fenêtre d'aide de LabVIEW et accéder à l'aide en ligne (*Online Reference* …)

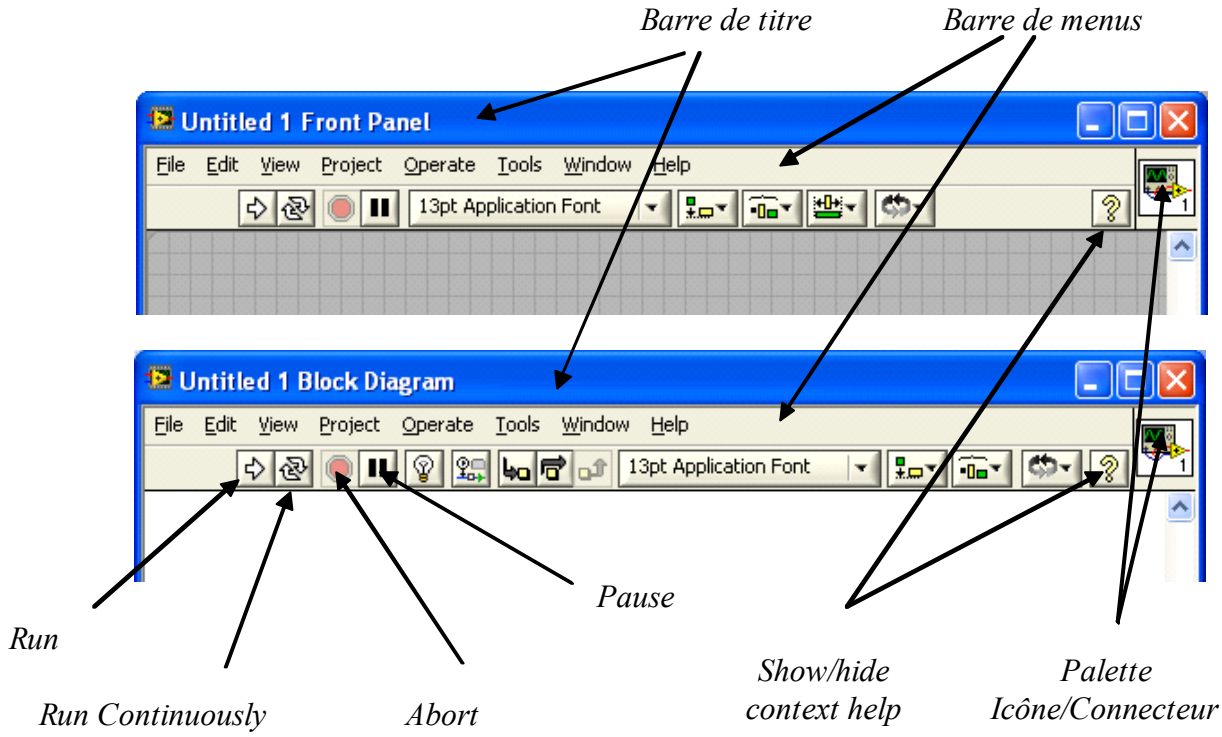

Figure 1-4 : Caractères communs aux deux fenêtres

Palette de boutons de sélection

Située juste au-dessous de la barre de menus, elle offre de nombreux boutons de sélection, dont au moins quatre boutons de commande communs aux deux fenêtres.

Boutons de commande communs aux fenêtres du diagramme et du « panneau avant » :

Exécution (Run)	Pour exécuter le programme
Exécution Répétée (Run Continuously)	Pour exécuter le programme continûment
Arrêt (Abort Execution)	Pour mettre fin au programme dans la phase de mise au point, l'algorithme achevé devant prévoir une fin de programme « normale »
Pause (Pause)	Pour faire une pause à l'appel d'un sous-programme afin d'observer ses entrées avant son exécution ou pour accéder au mode d'exécution pas à pas
Aide (Help)	Pour activer ou désactiver l'aide de LabVIEW sur chaque objet du programme

Menus surgissants

La zone située sous la palette des boutons de sélection est spécifique au diagramme ou au « panneau avant ». Lorsqu'on clique dans cette zone avec le bouton droit de la souris, un **menu surgissant** (*Pop-up Menu*) ou une palette d'objet apparaît à l'endroit pointé. Si l'on a cliqué dans une zone vide, une palette apparaît (*Controls* ou *Functions* suivant la fenêtre concernée), permettant d'introduire des objets dans la fenêtre. Si l'on a cliqué sur un objet situé dans la fenêtre, le menu surgissant sur l'objet est spécifique à l'objet et propose plusieurs choix pour modifier, initialiser, … l'objet en question.

L'emploi des menus surgissant sur les objets est très utile car il permet d'accéder aux paramètres de l'objet.

Palette icône/connecteur (Icon/Connector Panel)

Cette palette est située dans la partie supérieure droite des deux fenêtres de programme de LabVIEW (cf Figure 1-4). Elle comprend deux éléments superposés : l'icône et le connecteur. L'**icône** (*Icon*), visible par défaut, symbolise le nom du programme tel qu'il apparaîtra sur les diagrammes lorsqu'il sera utilisé en tant que sous-programme. Le **connecteur** (*Connector*), accessible par le choix *Show Connector* du menu surgissant sur l'icône du « panneau avant », permet le passage des arguments du sous-programme vers le programme appelant.

Il est conseillé d'écrire le nom du VI (ou un nom permettant sans ambiguïté de retrouver le VI) dans le carré réservé à l'icône. Ceci se fait à l'aide de l'éditeur d'icône accessible en mode d'édition, par le choix *Edit Icon* du **menu surgissant** de la palette icône/connecteur du « panneau avant ».

1.1.1.5 Les outils de développement spécifiques au diagramme

La palette de boutons du diagramme offre cinq outils de développement spécifiques à cette fenêtre (voir Figure ci-dessous).

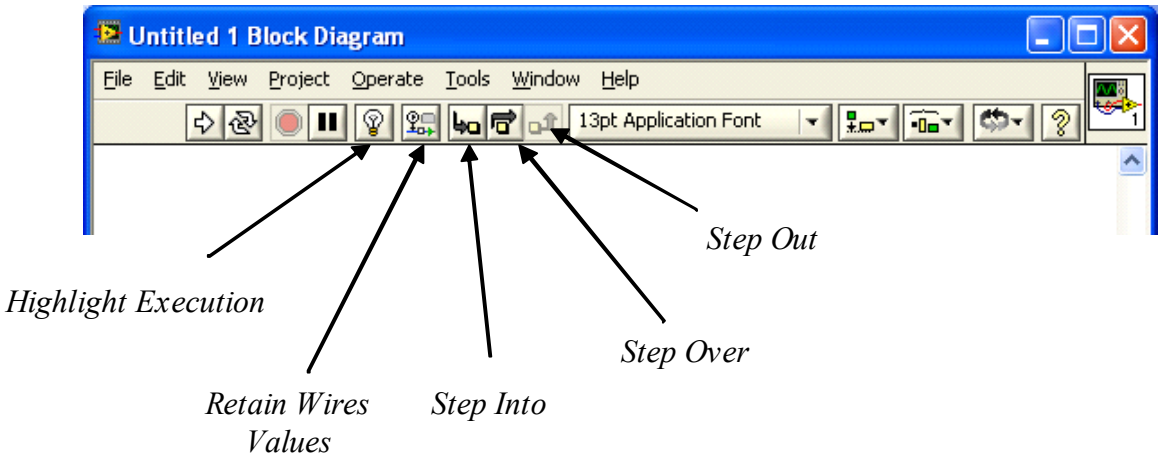

Figure 1-5 : Outils spécifiques au diagramme

Commandes spécifiques au diagramme :

Exécution ralentie animée (Highlight Execution)	Utile pour observer le déroulement du programme à l'aide de bulles simulant le passage des données sur les fils, mais en ralentissant considérablement l'exécution
Etat de la mémoire (Retain Wires Values)	Sert à visualiser les valeurs sur les fils reliant les VI en cours d'exécution
Entrée en mode exécution pas à pas (Step Into)	Sert à poursuivre, après une pause, l'exécution du programme pas à pas par pressions successives de ce bouton
Saut de nœuds en exécution pas à pas (Step Over)	Permet d'exécuter certains nœuds en mode normal et revenir au mode pas à pas à la sortie du nœud
Sortie du mode pas à pas (Step Out)	Permet de sortir du mode pas à pas

Ces options sont très pratiques si on parvient à les maîtriser. Pour en savoir plus, voir l'aide en ligne sur le sujet.

1.1.2 Les palettes

Sous l'environnement LabVIEW, le programmeur dispose d'une palette d'outils et de deux palettes d'objets, apparaissant par défaut en fonction du contexte, selon qu'on travaille sur le « panneau avant » ou sur le diagramme.

1.1.2.1 La palette d'outils

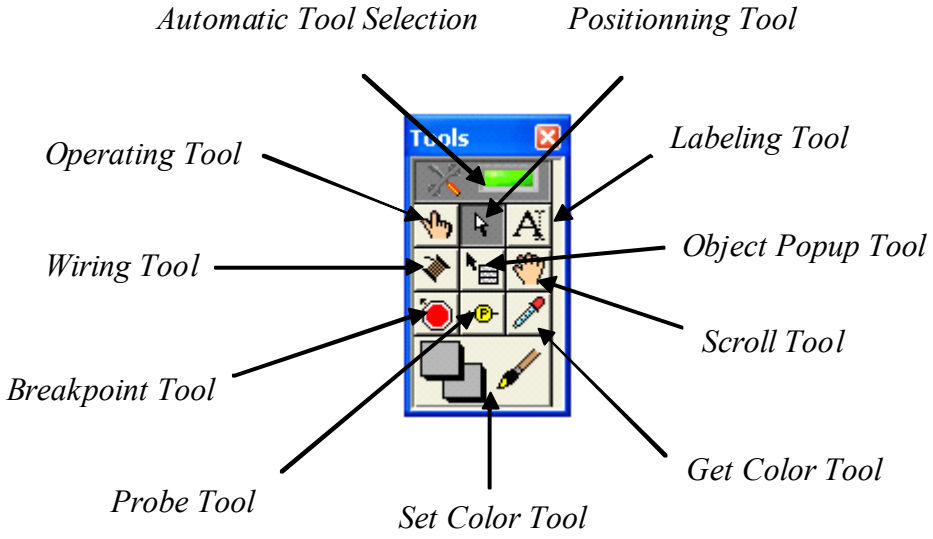

Figure 1-6 : Palette d'outils

Une **palette d'outils** (*Tools*) utilisables aussi bien pour le diagramme que pour le « panneau avant » apparaît par défaut lorsqu'on ouvre un VI et reste néanmoins toujours accessible par le choix *Tools Palette* du menu *View* si on l'a supprimée. Elle permet de définir divers modes de fonctionnement du curseur :

Outils de développement :

Breakpoint Tool	Pour insérer des points d'arrêt
Probe Tool	Pour insérer des sondes sur les fils permettant de connaître les valeurs transmises

Outils d'édition :

Operating Tool	Sert à actionner les objets de la fenêtre
Positionning Tool	Sert à sélectionner, déplacer, modifier les objets
Labeling Tool	Utilisé pour entrer du texte
Wiring Tool	Véhicule le flot de données entre les objets
Object Popup Tool	Fait surgir un menu sur un objet
Scroll Tool	Déplace la zone visualisée
Get Color Tool	Sert à copier la couleur d'un objet
Set Color Tool	Sert à colorier les objets et l'arrière-plan des fenêtres
Automatic Tool Selection	Laisse à LabVIEW la possibilité de sélectionner lui-même les outils de la palette à utiliser selon le lieu où se trouve le curseur (*Positionning Tool*, *Wiring Tool*, ..)

1.1.2.2 La palette d'objets du « panneau avant » (*Controls Palette*)

Cette palette apparaît par défaut lorsque la fenêtre du « panneau avant » est active et reste accessible, si on l'a fermée, par le choix *Controls Palette* du menu *View* ou par le bouton de droite de la souris sur le « panneau avant ».

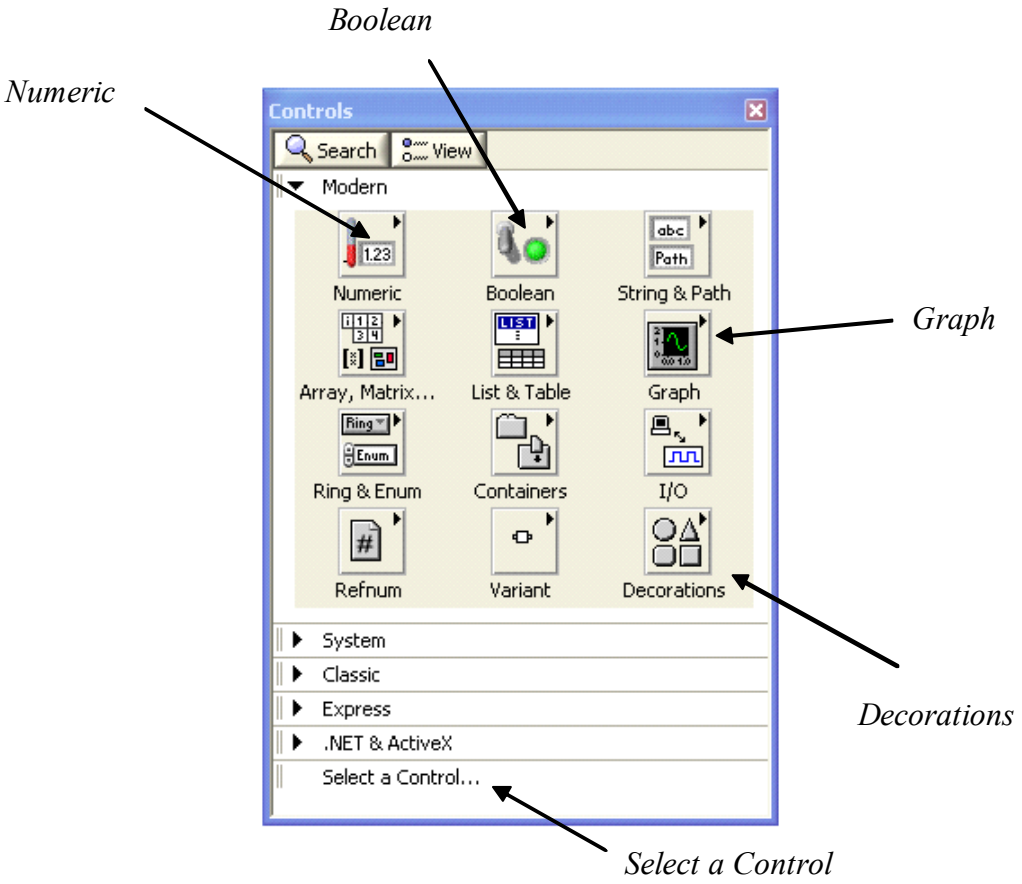

Figure 1-7 : Palette d'objets du panneau avant (*Controls Palette*) apparaissant par défaut

Les principaux choix offerts par cette palette fournissent des objets représentant les entrées-sorties du programme. Ils correspondent à des types prédéfinis du langage (*Numeric*, *Boolean*, *String & Path*, *List & Tables*, *Array & Matrix*, *Graph*, *Refnum, ...*), à des symboles de décoration divers (*Decorations*) ou à des types définis par l'utilisateur (*Select a Control ...*). Les types prédéfinis du langage seront détaillés au chapitre 3.

En fait, sur cette palette, les objets sont représentés plusieurs fois. La forme la plus synthétique de la *Controls Palette* est représentée sur la Figure 1-8. Schématiquement, on peut dire que les quatre choix « *Modern* », « *System* », « *Classic* » et « *Express* » représentent quatre styles graphiques différents, mais rien de plus. Les objets LabVIEW peuvent être pris dans l'un des quatre, seule leur apparence physique sur le « panneau avant » sera différente. Leurs fonctions seront identiques. Ceci est souligné sur la Figure 1-9 qui représente quatre objets identiques (des contrôleurs numériques) d'apparence différente, mais leur fonction intrinsèque reste la même.

Figure 1-8 : Forme la plus synthétique de la *Controls Palette*

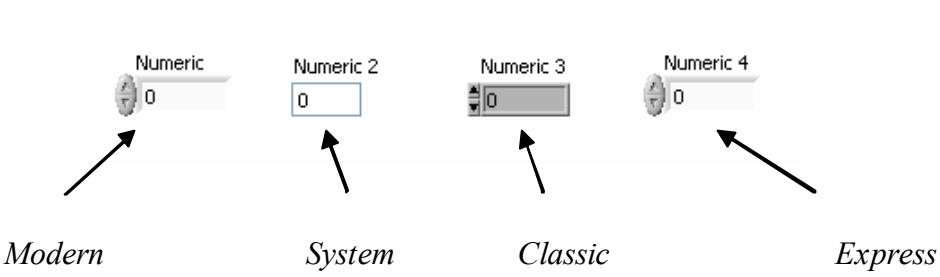

Figure 1-9 : Les quatre types de contrôleur numérique disponibles

Comme l'ouverture de la *Controls Palette* se fait par défaut sur le style « *Modern* » (Figure 1-7), on choisira ce style pour les exemples donnés ici et dans le polycopié d'exercices, mais vous pouvez choisir un autre style si bon vous semble !

1.1.2.3 La palette d'objets du diagramme (*Functions Palette*)

Cette palette apparaît par défaut lorsque la fenêtre du diagramme est active et reste accessible, si on l'a fermée, par le choix *Functions Palette* du menu *View* ou par le bouton de droite de la souris sur le diagramme.

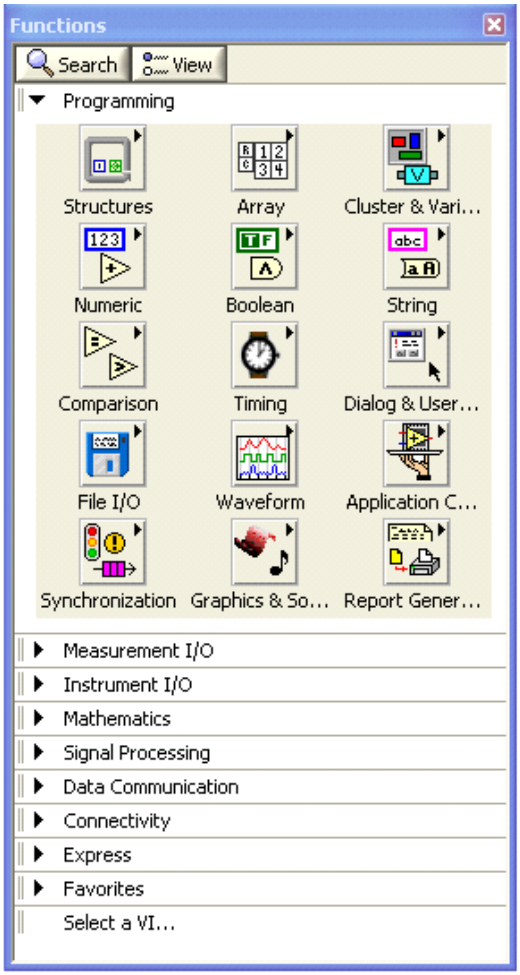

Figure 1-10 : Palette d'objets du diagramme (*Functions Palette*) apparaissant par défaut

Les principaux choix offerts par cette palette correspondent à des opérateurs ou des fonctions prédéfinies du langage. Sans panneau avant ni diagramme, les fonctions fournissent du code machine en ligne et sont pour la plupart **polymorphes**, c'est à dire qu'elles s'adaptent aux types (réels, entiers, tableaux de réels, tableaux d'entiers, …) et représentations des données (décimales ou hexadécimales par exemple). La plupart de ces fonctions seront détaillées au chapitre 3.

La forme la plus synthétique de la *Functions Palette* est représentée sur la Figure 1-11. Cette palette regroupe toutes les fonctions qui peuvent être effectuées par LabVIEW. Elles seront détaillées par la suite.

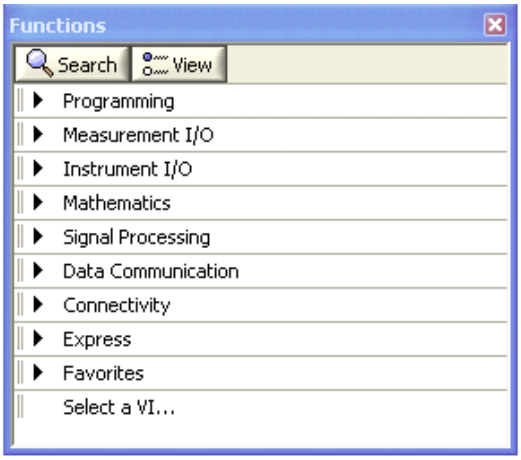

Figure 1-11 : La forme la plus synthétique de la *Functions Palette*

Comme tous les langages évolués, le langage G permet à une fonction d'en appeler une autre. Ceci se fait en insérant sur le diagramme l'icône de la fonction qui servira de sous-programme à l'aide du choix « *Select a VI...* » de la *Functions Palette* (Figure 1-12).

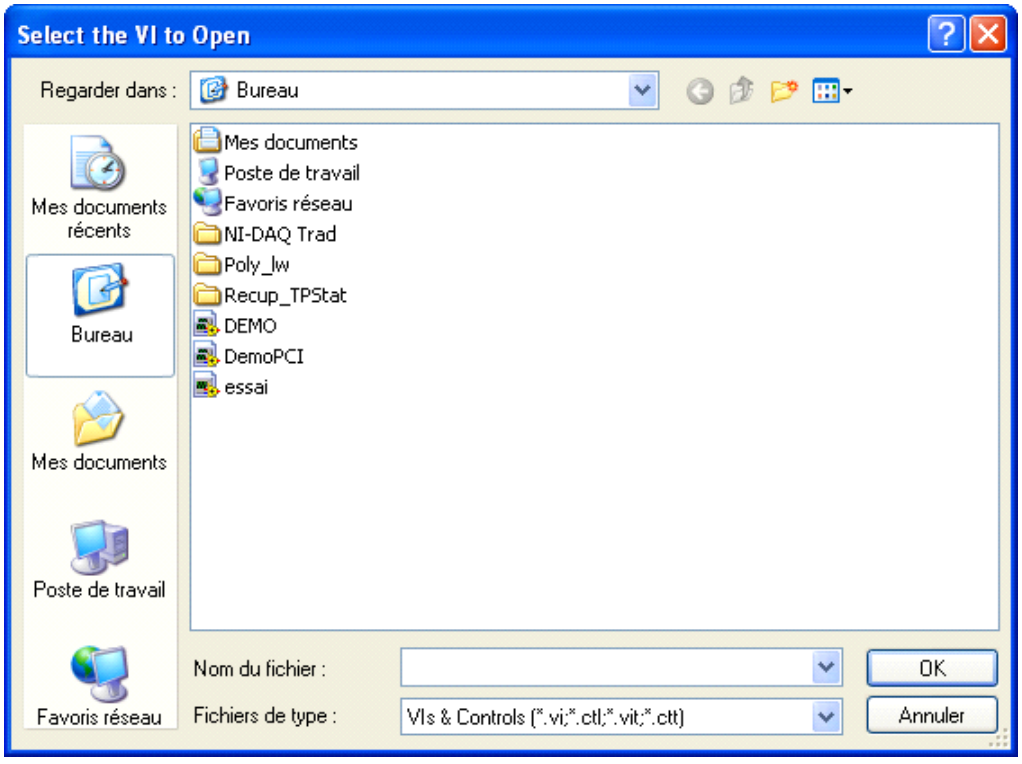

Figure 1-12 : Fenêtre apparaissant en choisissant le menu « *Select a VI...* » de la *Functions Palette*

1.2 Les modes de fonctionnement de LabVIEW

LabVIEW présente deux modes de fonctionnement : un mode d'édition (*Edit Mode*) et un mode d'exécution (*Run Mode*). Le passage d'un mode à l'autre se fait par le choix *Change to Run/Edit Mode* du menu *Operate*.

1.2.1 Le mode d'édition

Le mode d'édition est proposé par défaut dès l'ouverture d'un nouveau programme. Il faut être en mode d'édition pour installer des objets dans le panneau avant, créer le programme dans le diagramme, accéder à l'éditeur d'icône sur le panneau avant, … Toute modification dans le programme ne peut se faire qu'en mode d'édition.

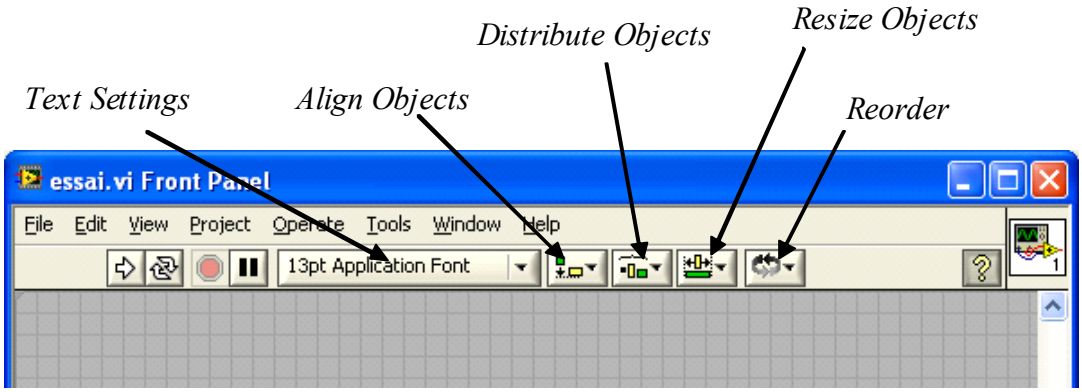

Figure 1-13 : Palette de boutons spécifiques au mode d'édition

Visuellement, le mode d'édition se distingue très facilement du mode d'exécution car la grille qui apparaît automatiquement

En mode d'édition, la palette de boutons offre, en plus des fonctionnalités déjà décrites, les moyens d'accès à trois nouvelles palettes d'outils ainsi que la possibilité de modifier les caractéristiques d'affichage des textes (police, taille, …).

Outils supplémentaires en mode d'édition (diagramme et panneau avant) :

Text Settings	Modifie les caractéristiques de l'affichage des textes (taille, police, couleur, …)
Align Objects	Aligne les objets sur l'écran
Distribute Objects	Répartit la densité des objets sur l'écran
Resize Objects	Modifie la dimension des objets
Reorder	Modifie l'ordre des objets sur l'écran

Le menu surgissant sur un objet du panneau avant propose huit choix standard en mode d'édition, auquel s'ajoutent des choix liés au type de l'objet. Les plus utiles sont détaillés ci-dessous.

Ces options sont très pratiques si on parvient à les maîtriser. Pour en savoir plus, voir l'aide en ligne sur le sujet.

Choix standard d'un menu surgissant sur un objet en mode d'édition :

Change to Indicator/Control	Permet de changer un contrôleur en indicateur et vice-versa
Find Terminal/Indicator/Control	Permet de trouver la terminaison associée à l'objet
Show	Fait afficher certains caractéristiques de l'objet dont une chaîne de caractère qui lui est associée (*Label*)
Data Operations	Permet d'accéder notamment aux valeurs par défaut
Create Attribute Node / Local Variable	Pour accéder aux paramètres de l'objet en mémoire
Replace	Permet de remplacer l'objet directement par un autre à l'aide du menu surgissant

1.2.2 Le mode d'exécution

Le mode d'exécution (*Run mode*) est proposé par défaut à l'ouverture d'un programme exécutable. C'est le mode recommandé pour l'utilisation des programmes après leur mise au point. Visuellement, le mode d'exécution se distingue très facilement du mode d'édition car la grille qui apparaît automatiquement en mode d'édition (Figure 1-13) disparaît du « panneau avant » en mode d'exécution (Figure 1-14).

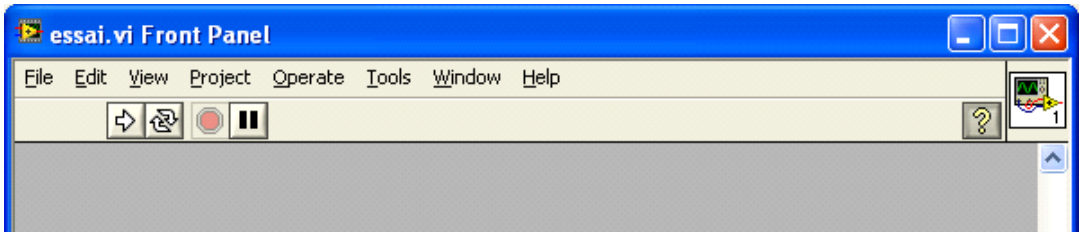

Figure 1-14 : Palette de boutons en mode d'exécution

Le menu surgissant sur un objet du panneau avant propose six choix standard en mode d'exécution, auxquels s'ajoutent des choix liés au type de l'objet.

Choix standard d'un menu surgissant sur un objet en mode d'exécution :

Reinitialize to Default	Réinitialise la valeur d'un contrôleur à sa valeur par défaut
Cut Data	Supprime de la fenêtre et copie dans une mémoire tampon de Windows un objet ou un ensemble d'objets
Copy Data	Copie dans une mémoire tampon de Windows un objet ou un ensemble d'objets
Paste Data	Affiche sur la fenêtre active le contenu de la mémoire tampon de Windows utilisée par les opérations *Copy* et *Paste*
Description …	Associe un commentaire à l'objet
Online Help	Aide en ligne. Très utile !!!

Le mode d'exécution permet, sans modification du programme, d'installer dans le diagramme des afficheurs temporaires sur les fils (*Probes*) afin de visualiser les valeurs des données circulant sur les fils pendant l'exécution du programme. Ces sondes s'installent en utilisant le choix *Probe* du menu déroulant sur le fil que l'on souhaite tester.
La différence entre les options accessibles sur un objet en mode d'exécution et en mode d'édition est visible par exemple sur la Figure 1-15 pour un indicateur.

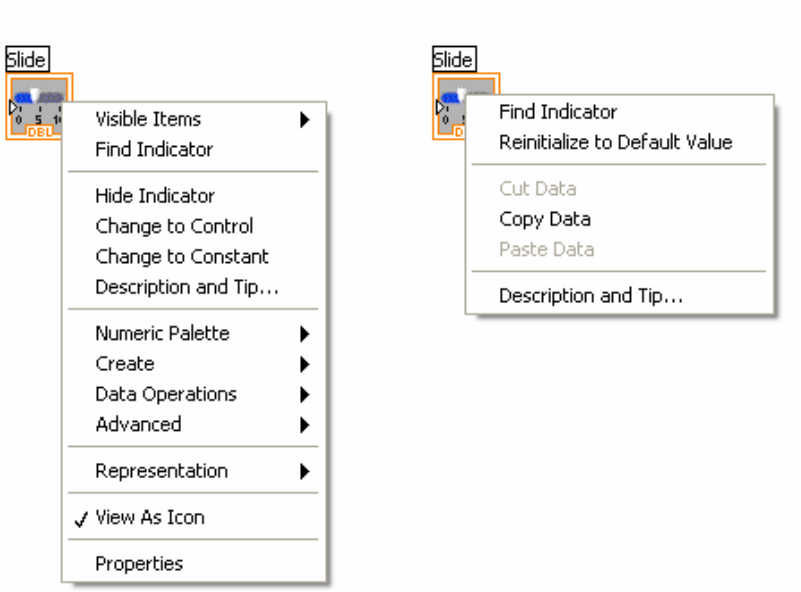

Figure 1-15 : Menus surgissant sur un indicateur en mode d'édition (à gauche) et en mode d'exécution (à droite)

1.3 Ouverture d'un programme LabVIEW

1.3.1 Ouvrir LabVIEW la 1ère fois

Pour ouvrir LabVIEW à partir du logiciel, il suffit d'utiliser le menu « démarrer » de la barre de menu du bas de l'écran, puis de sélectionner « LabVIEW 8.0 » dans « Tous les programmes ». On voit alors apparaître la fenêtre de la Figure 1-16. Il suffit ensuite de sélectionner *Blank VI* pour créer un VI vide.

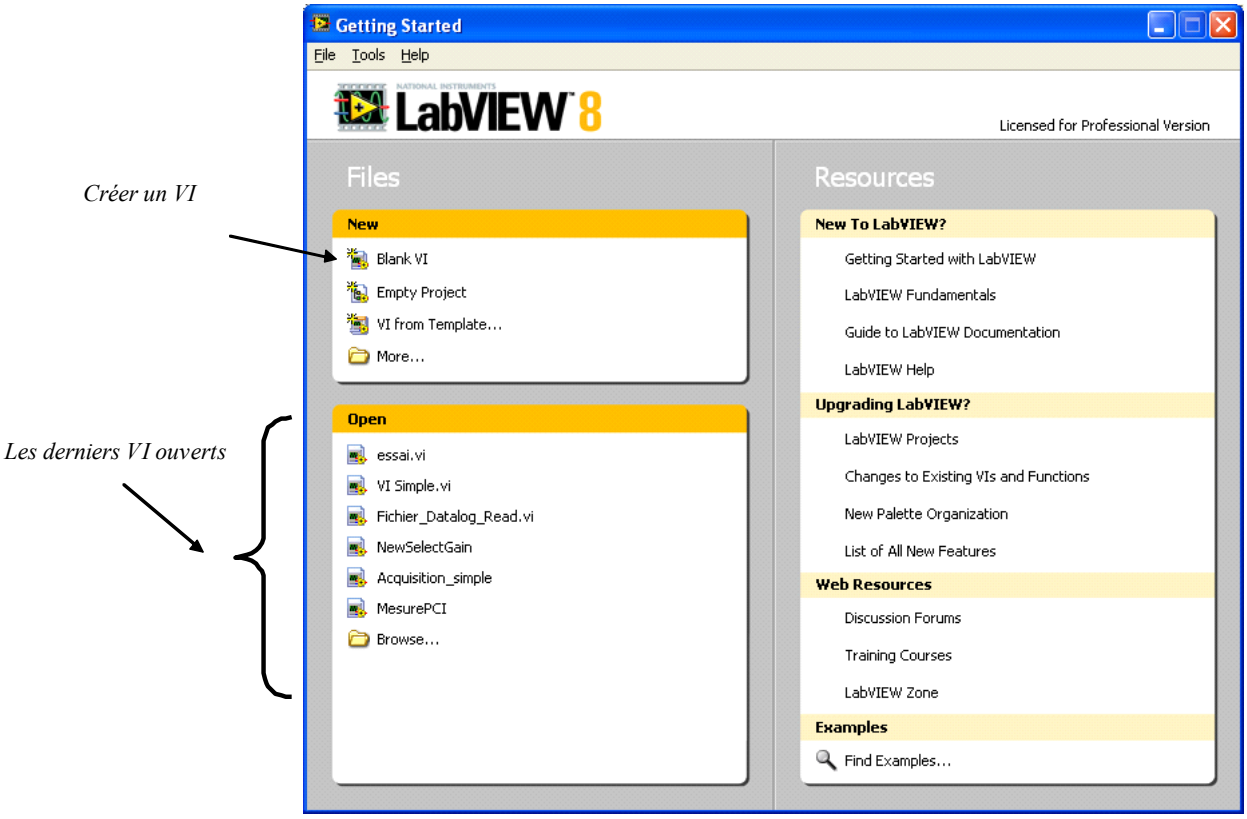

Figure 1-16 : Fenêtre apparaissant à l'ouverture de LabVIEW

1.3.2 Programme existant hors d'une bibliothèque

Pour ouvrir un VI qui a été sauvé en dehors d'une bibliothèque, il suffit de l'ouvrir à l'aide de la souris, comme n'importe quel objet *Windows*.

1.3.3 Programme existant au sein d'une bibliothèque

Pour ouvrir un VI qui a été sauvegardé dans une bibliothèque LabVIEW (par exemple Demo.llb), il suffit d'ouvrir la bibliothèque à l'aide de la souris, comme n'importe quel dossier *Windows*. Il aura alors l'apparence de la Figure 1-17. Les VI sont traités par *Windows* comme n'importe quel fichier. On peut les copier d'une bibliothèque vers une autre, les supprimer, etc .. à l'aide des facilités standards de *Windows*. La seule restriction est qu'on ne peut mettre dans les bibliothèques LabVIEW que des objets reconnus par LabVIEW (les VI).

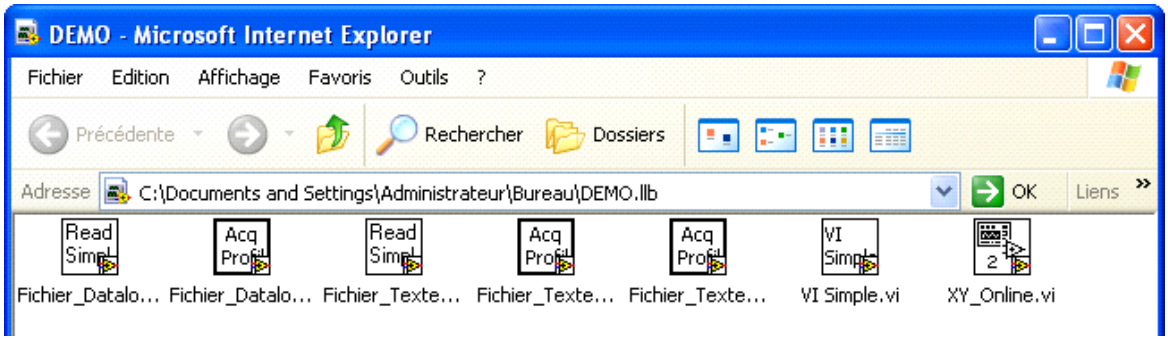

Figure 1-17 : Une bibliothèque LabVIEW

1.4 Sauvegarde des programmes

LabVIEW permet de sauvegarder un programme sous deux formes :

1. sous forme d'un fichier indépendant. Le nom du fichier doit alors être conforme au système d'exploitation, sans limitation sur le nombre maximum de caractères (dans les limites du raisonnable). On pourra par exemple prendre « Nom de VI.vi ».
2. sous forme de constituant d'un fichier bibliothèque. Ni la longueur du nom du fichier ni celle de la bibliothèque ne sont limitées

Pour des raisons de portabilité, de gestion par LabVIEW et d'économie d'espace disque, il faut privilégier les sauvegardes sous la deuxième forme. Les bibliothèques apparaissent sous *Windows* comme des dossiers d'extention « .llb » (par exemple Demo.llb »).

La sauvegarde d'un programme est réalisée via les options *Save* ou *Save as* du menu *File* de la barre des menus. Si le chemin n'est pas correct, il convient de faire les choix convenables dans l'arborescence des répertoires et fichiers, à l'aide des menus déroulants de la fenêtre (choix du volume, des répertoires, … etc).

Remarque

Lorsque la sauvegarde doit être faite dans un fichier bibliothèque qui n'existe pas encore, on doit d'abord créer la bibliothèque par le choix *Nouvelle bibliothèque de VIs* de la fenêtre qui apparaît lorsqu'on sélectionne *Save* ou *Save as*. L'utilisateur doit taper le nom de la nouvelle bibliothèque (MaBiBli.llb) dans une fenêtre spécifique.

LabVIEW présente une nouvelle fenêtre *Name the VI* avec le nom du nouveau fichier bibliothèque, afin de procéder à la sauvegarde du programme dans ce fichier.

1.5 Quelques conseils de programmation

La programmation en langage G doit utiliser la modularité en créant des sous-programmes spécifiques aux diverses tâches à exécuter. Il faut néanmoins se rappeler que le langage G n'est pas récursif.

Il est conseillé de limiter la taille des programmes en limitant la taille des diagrammes : si le diagramme est plus grand que l'écran, c'est que le programme est vraisemblablement trop complexe. Il faut alors créer un ou plusieurs sous-programmes pour simplifier le diagramme.

1.6 Utilisation de l'aide de LabVIEW

L'aide de LabVIEW est accessible à partir du menu *Help*. L'aide la plus simple est obtenue par le choix « *Show Context Help* » et donne des informations sur les objets, la façon de les relier les uns aux autres, etc… On peut également faire apparaître/disparaître cette fenêtre à l'aide du bouton de la barre de menus (Figure 1-18).

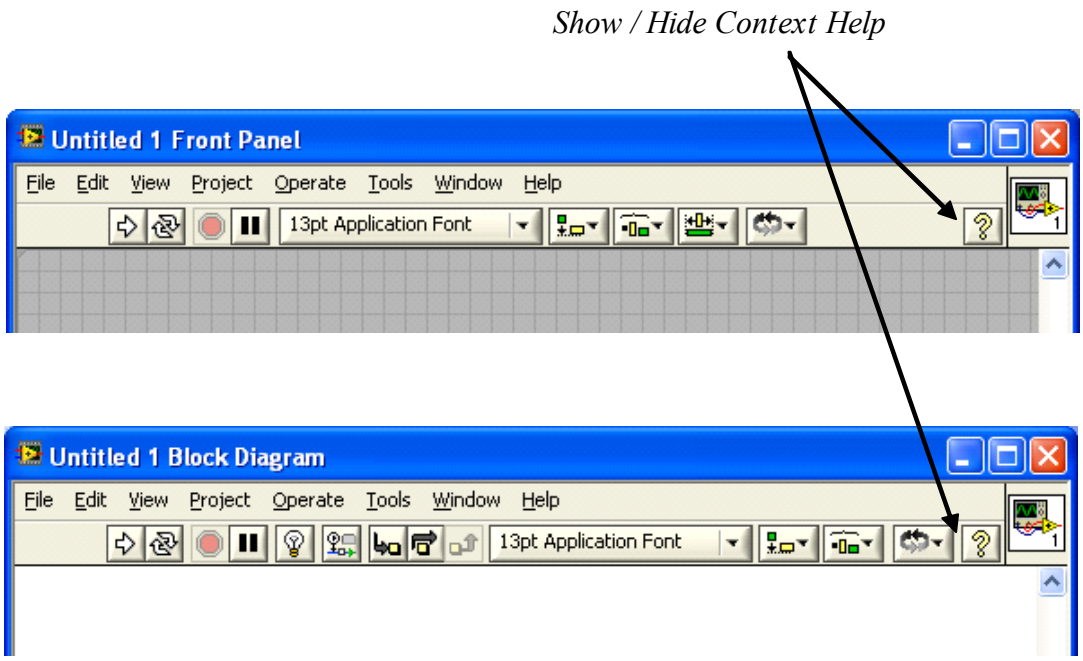

Figure 1-18 : La fenêtre *Context Help* peut être accédée directement depuis la barre de menus

Un niveau plus poussé est obtenu par le choix « *Search the LabVIEW Help* » qui permet de faire une recherche par mot clef (Figure 1-19). La maîtrise de cet outil vous facilitera grandement la vie !

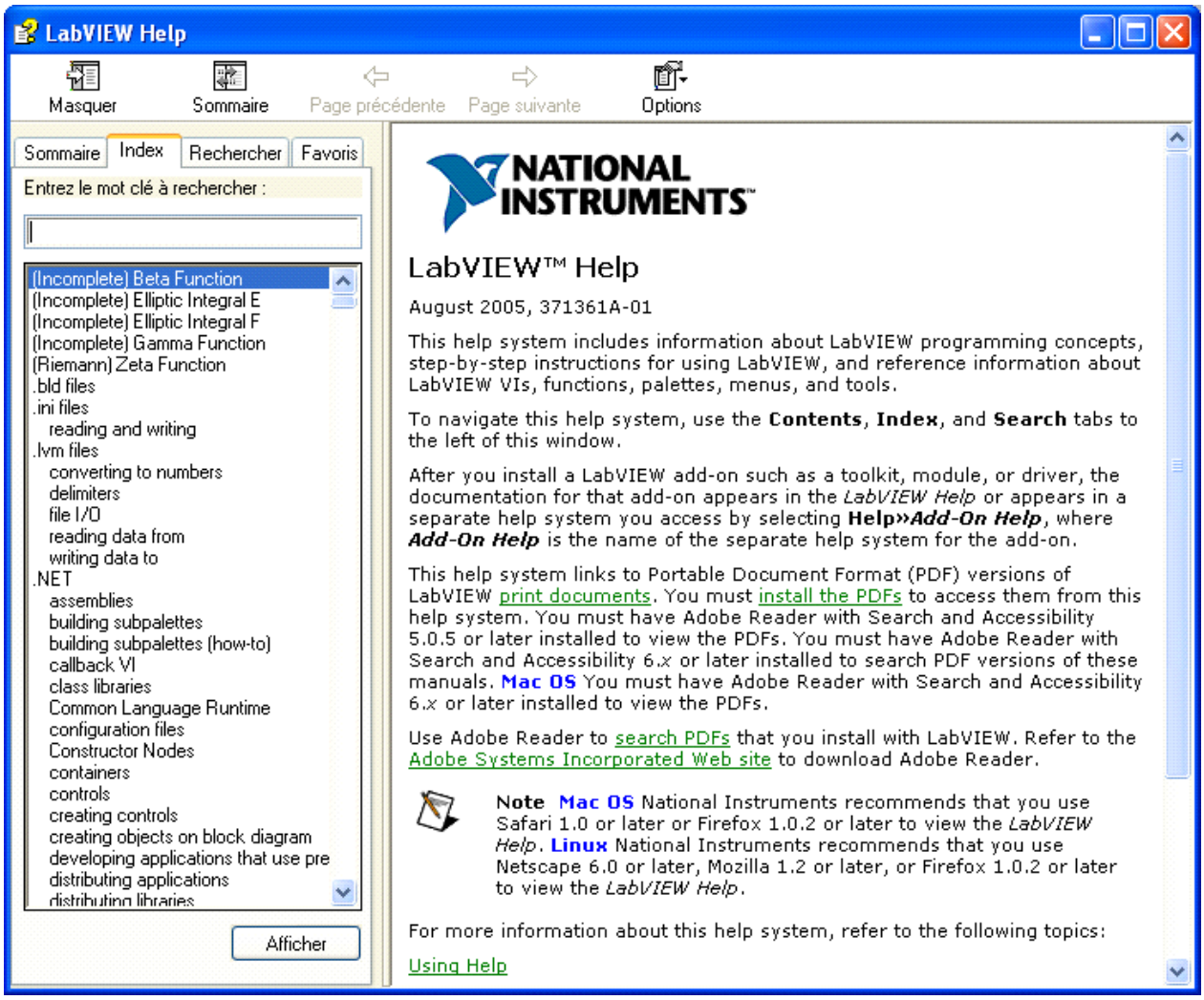

Figure 1-19 : Fenêtre principale de l'aide en ligne de LabVIEW

2 Guide pratique pour l'utilisation de LabVIEW

La programmation de LabVIEW étant organisée à partir des *Controls Palette* et *Functions Palette* décrites ci-dessus, toute l'information nécessaire se trouve rassemblée sur ces palettes. Ce paragraphe décrit donc de manière succincte où trouver dans LabVIEW les fonctionnalités les plus utiles pour nos applications.

On peut arbitrairement classer en deux catégories les concepts utilisés en programmation : certains sont liés à une représentation graphique et d'autres n'y sont pas. Les premiers seront rattachés au « panneau avant », tandis que les seconds seront rattachés au diagramme.

2.1 Concepts liés au « panneau avant »

Les concepts liés au « panneaux avant » sont liés à une représentation graphique et sont accessibles à partir de la *Controls Palette*. On aura par exemple dans cette catégorie :

- **Les nombres** : On y accède par un panneau *Numeric* de la *Controls Palette*
- **Les booléens** : On accède à ces variables ne pouvant prendre que deux valeurs par un panneau *Boolean* de la *Controls Palette*
- **Les chaînes de caractère** : On y accède par un panneau *String & Path* de la *Controls Palette*
- **Les graphiques :** On y accède par le panneau Graph de la Controls Palette

2.2 Concepts liés au diagramme

Les concepts liés au diagramme uniquement ne donnent pas lieu à une représentation graphique. Ce sont généralement des concepts de programmation. On aura par exemple :

- **Le traitement mathématique des données** : On y accède par un panneau *Mathematics* de la *Functions Palette*
- **Le traitement du signal** : On y accède par un panneau *Signal Processing* de la *Functions Palette*
- **L'acquisiton des données** : On verra ultérieurement qu'on y accède par les panneau *Instrument I/O* et *Measurement I/O* de la *Functions Palette*

Les principales palettes qu'on utilisera sont regroupées sur la Figure 2-1.

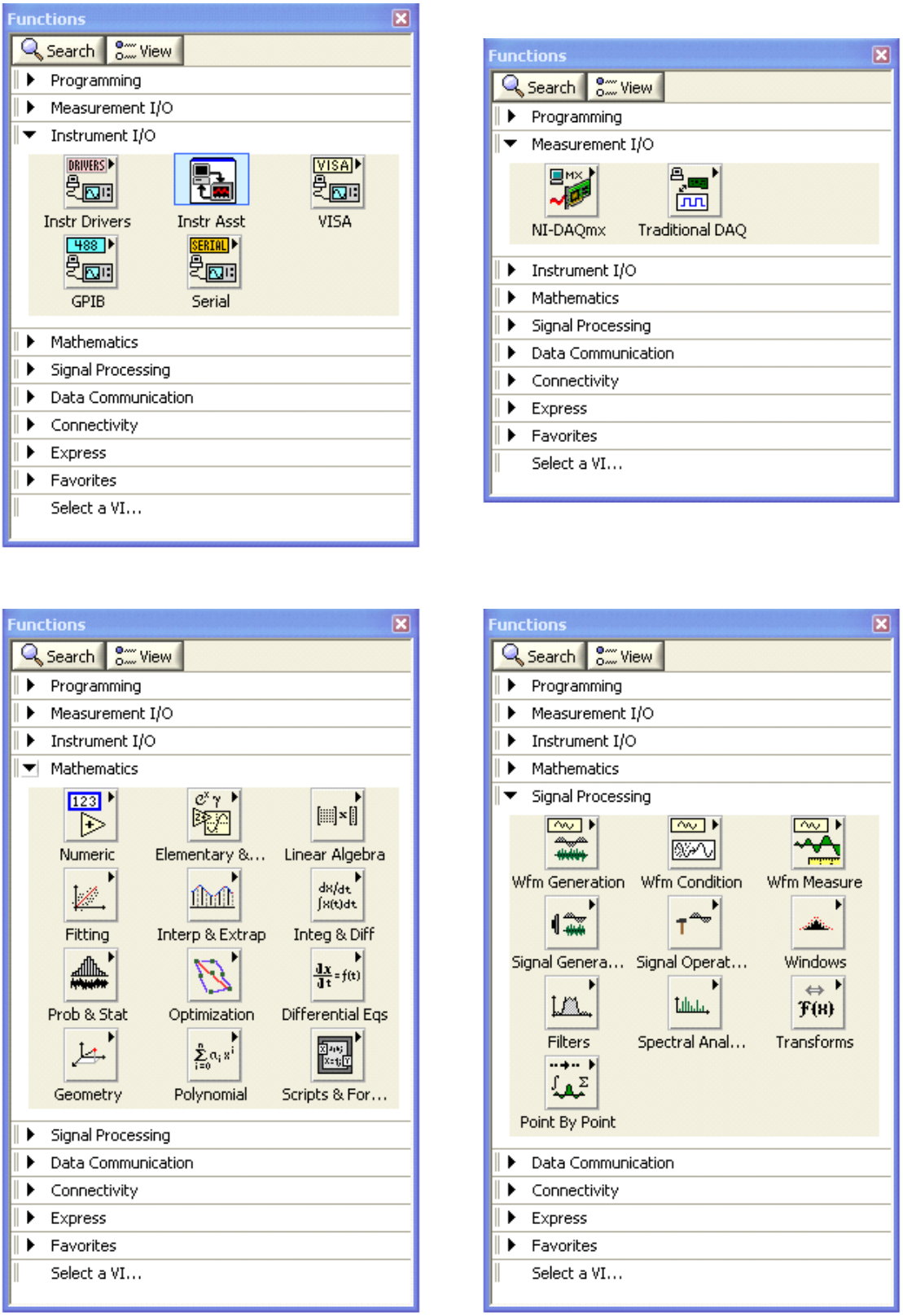

Figure 2-1 : Les principaux choix de la palette *Functions*

3 Les principaux objets de LabVIEW

3.1 Les objets présents sur le panneau avant et sur le diagramme

Ces objets sont accessibles uniquement à partir de la *Controls Palette* du « panneau avant » et ne peuvent pas être introduits dans le programme par le diagramme.

3.1.1 Les nombres

Sur le panneau avant

Le contrôleur/indicateur digital (*Digital Control*/*Digital Indicator*), accessible dans la sous-palette *Numeric* de la palette *Controls*, est l'objet fondamental pour la manipulation des nombres. Il se place sur le panneau avant et se présente à l'écran sous forme d'une fenêtre rectangulaire dans laquelle s'affichent les chiffres constituant le nombre, avec une palette d'incrémentation/décrémentation lorsqu'il s'agit d'un contrôleur (un indicateur n'aura évidemment pas cette palette d'incrémentation/décrémentation).

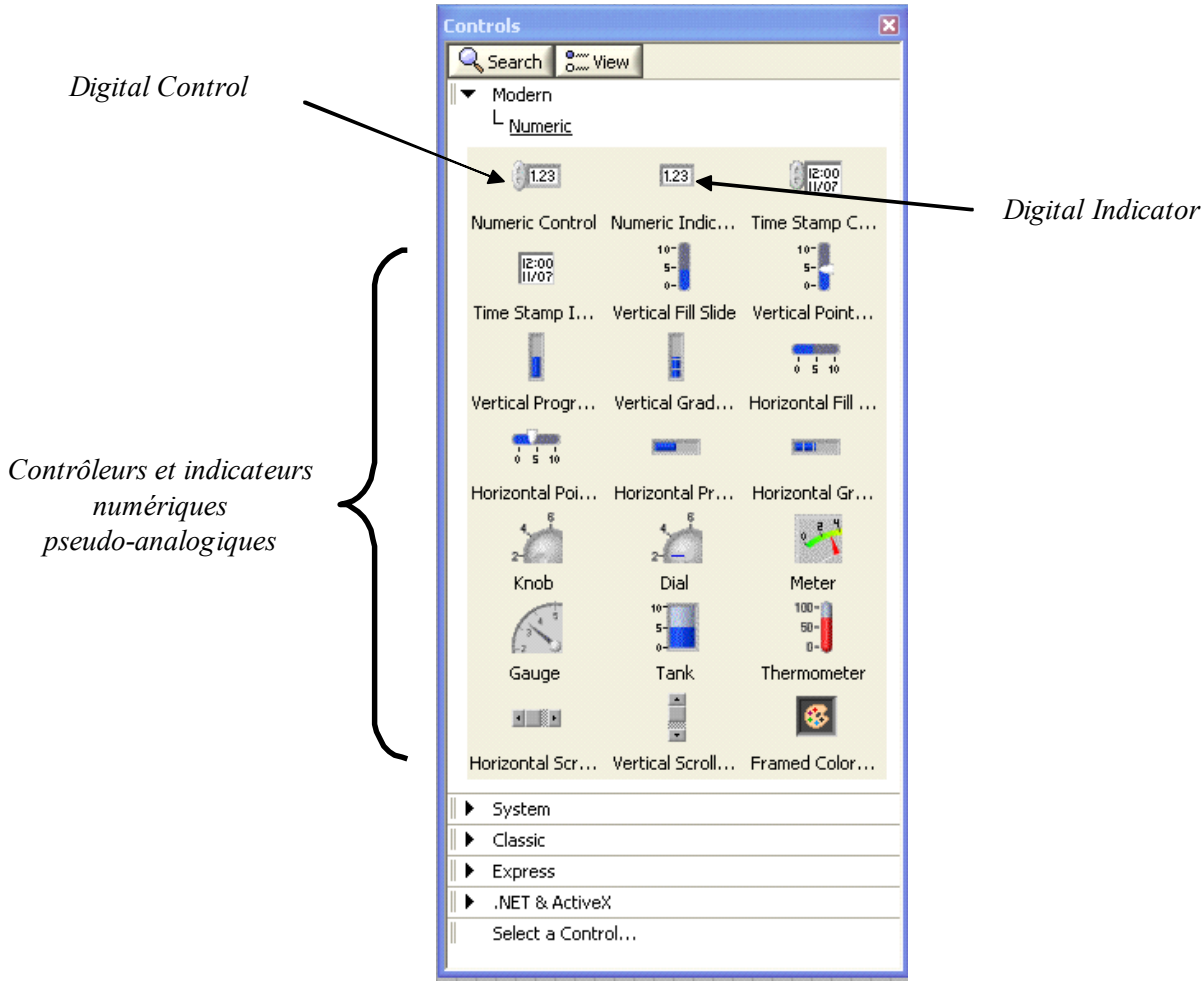

Figure 3-1 : Choix *Numeric* de la palette *Controls*

De nombreux contrôleurs et indicateurs numériques apportent, sous forme d'image à l'écran, une représentation dite « pseudo-analogique » symbolisant le rôle attribué à une donnée dans le programme. Ils disposent, en plus des attributs de l'afficheur digital, d'attributs supplémentaires associés à leur image. Ces contrôleurs, qui ressemblent à des éléments de réglage couramment employés sur les appareillages de laboratoire, et ces indicateurs, qui ont l'apparence d'instruments de mesure, ont conduit à attribuer aux programmes de LabVIEW le nom d'Instruments Virtuels. Ils sont accessibles sur la même palette que les contrôleurs et indicateurs digitaux et ont des formes de bouton, d'indicateur à aiguille, etc .. Fondamentalement, ils n'apportent qu'un confort visuel (dont il serait néanmoins dommage de se passer) et peuvent être toujours remplacés par des contrôleurs et indicateurs digitaux.

Sur le diagramme

Sur le diagramme, la terminaison associée à un contrôleur digital (ou un contrôleur pseudo-analogique) est un rectangle à contour épais. Ce rectangle est à contour mince pour un indicateur. Dans les deux cas, il contient par défaut les trois caractères « DBL », caractéristiques d'un nombre réel en double précision.

Figure 3-2 : Représentation des variables numériques sur le diagramme

Représentation en mémoire

Lorsqu'on ne veut pas de la représentation par défaut, l'option spécifique *Representation* du menu surgissant (accessible sur le panneau avant et sur le diagramme) permet de modifier la représentation du nombre en mémoire. Les principales représentations sont :

Pour les nombres entiers :

		Codage (octet)	Gamme
I8	*Integer*	1	-128 .. +127
I16	*Integer*	2	-32768 .. +32767
I32	*Integer*	4	-2147483648 .. +2147483647
U8	*Unsigned Integer*	1	0.. 255
U16	*Unsigned integer*	2	0 .. 65535
U32	*Unsigned integer*	4	0 .. 4294967295

Pour les nombres réels :

		Codage (octet)	Chiffres significatifs (, et . inclus)
SGL	*Single Precision Real*	4	7
DBL	*Double Precision Real*	8	15

Les modes I64, U64 et EXT (*Extended Precision Real*) n'ont pas d'applications pratiques ici, ainsi que les modes CSG, CDB et CXT correspondant à l'utilisation des nombres complexes.

Les nombres entiers (I8, I16, I32, U8, U16, U32) apparaissent dans des rectangles bleus sur le diagramme, les nombres réels (SGL, DBL) comme des rectangles oranges (Figure 3-3).

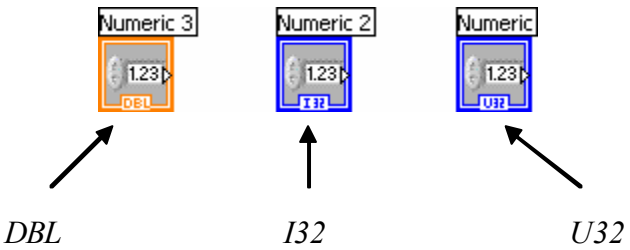

Figure 3-3 : Représentation sur le diagramme de contrôleurs DBL, I32 et U32

Constantes numériques

On peut avoir besoin à certains moments de constantes numériques et non plus de variables. Elles sont accessibles sur le diagramme par le choix *Numeric* de la palette *Functions*. Attention, les constantes ne sont pas accessibles depuis le « panneau avant » ! Comme pour les variables, on peut modifier leur représentation.

Remarques

- Sur le panneau avant, l'option *Format & Precision* du menu surgissant permet de modifier la précision de l'affichage des nombres (6 décimales par défaut).
- Sur le panneau avant, l'option *Format & Precision* du menu surgissant permet de modifier la base (décimale, hexadécimale, octale ou binaire) sous laquelle sont entrés et affichés les nombres entiers (base décimale par défaut). Attention, pour une représentation hexadécimale, il faut utiliser un nombre entier !!
- Les valeurs limites d'utilisation d'une variable ainsi que des options d'incrémentation et de contrôle peuvent être choisies par l'utilisateur via l'option *Data Range* du menu surgissant.
- De manière générales, de nombreuses options sur l'apparence des contrôleurs/indicateurs sont accessibles par l'option *Properties* du menu surgissant de l'objet.

3.1.2 Les booléens

Sur le panneau avant

LabVIEW fournit des contrôleurs logiques, ou **booléens**, permettant à l'utilisateur de contrôler le déroulement du programme en imposant leur état (via une commande) ou en le recevant (via un indicateur). En mode édition, la sous-palette *Boolean* de la palette *Controls* permet d'accéder à ces contrôleurs booléens (Figure 3-4).

Les booléens peuvent prendre 2 états logiques, Vraie (*True*) ou Faux (*False*). L'état par défaut est Faux (*False*), mais ceci peut être modifié à l'aide du bouton de manœuvre (*Operating Tool*), puis du choix *Make Current Value Default* de l'option *Data Operations* du menu surgissant sur l'objet.

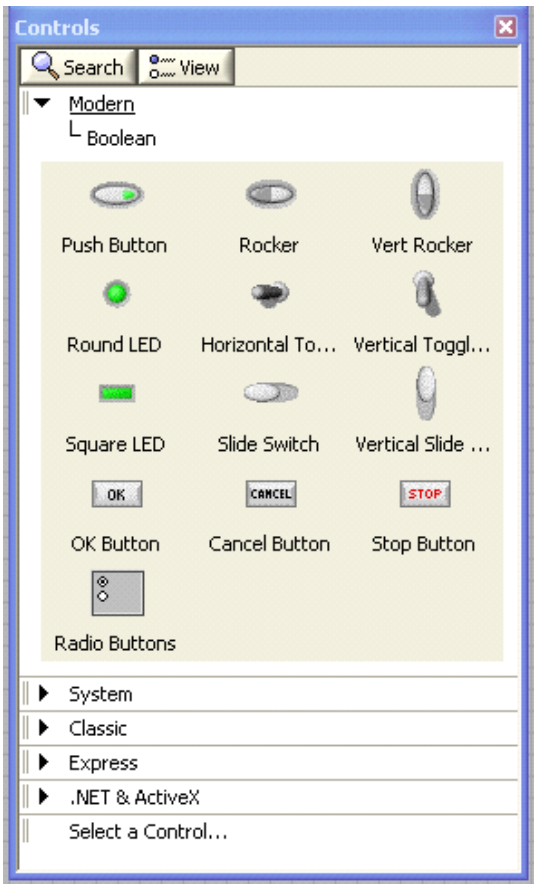

Figure 3-4 : Choix *Boolean* de la palette *Controls*

Dès l'apparition d'une commande booléenne sur le panneau avant, l'utilisateur doit définir son mode de fonctionnement, via l'option *Mechanical Action* du menu surgissant sur l'objet. Une palette propose alors six choix permettant de définir le mode de fonctionnement d'une commande booléenne pendant l'exécution :

Trois modes ne dépendent que de l'action sur le booléen :

Switch When Pressed	Permet de changer d'état à chaque pression (analogue avec un interrupteur électrique)
Switch When Released	Permet de changer d'état à chaque relâchement
Switch Until Released	Permet de changer d'état pendant la durée de la pression (analogue avec un bouton de sonnette)

Trois modes prennent en compte l'action et la lecture de l'état par le programme :

Latch When Pressed	Permet de changer d'état à chaque pression et de retourner à l'état initial ensuite
Latch When Released	Permet de changer d'état au relâchement et de retourner à l'état initial ensuite
Latch Until Released	Permet de changer d'état pendant la durée de la pression et de retourner à l'état initial ensuite

Sur le diagramme

La terminaison du diagramme associée à un booléen est un rectangle vert, à contour épais (pour une commande) ou mince (pour un indicateur), contenant les deux caractères « TF », indiquant le type booléen (Figure 3-5).

Figure 3-5 : Représentation d'un booléen sur le diagramme

La constante booléenne

Elle n'est accessible que sur le diagramme par le choix *Boolean* de la palette *Functions*. La valeur de la constante (*TRUE* ou *FALSE*) peut être modifiée à l'aide de l'*Operating Tool* de la palette d'outils.

3.1.3 Les chaînes de caractères

Les contrôleurs et indicateurs alphanumériques, manipulant le texte par chaînes de caractères (*String*), constituent un type de données de base de LabVIEW, à la différence de nombreux langages où le seul type rencontré est le caractère.

Sur le panneau avant

En mode édition, la sous-palette *String & Path* de la palette *Controls* propose un contrôleur d'entrée de chaîne de caractères (*String Control*) ainsi qu'un indicateur de sortie (*String Indicator*).

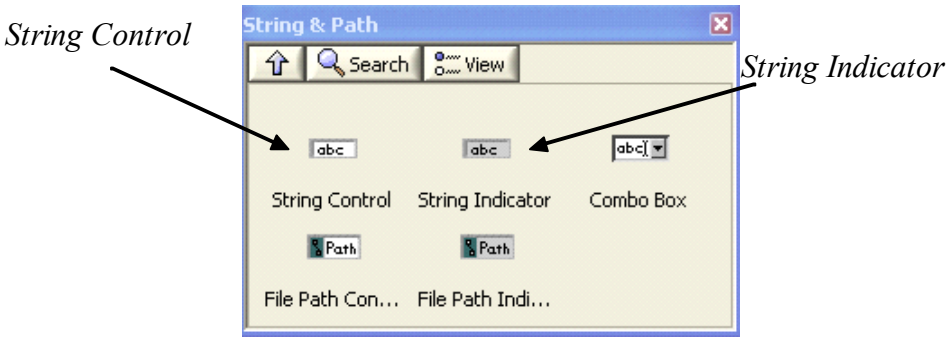

Figure 3-6 : Choix *String & Table* de la palette *Controls*

Pour entrer du texte dans un contrôleur alphanumérique, on utilise les outils d'écriture (*Labeling Tool*) ou de commande (*Operating Tool*), puis on valide l'entrée soit par un clic du curseur à l'extérieur du contrôleur (ou sur le bouton Enter de la palette d'outils), soit par la frappe de la touche ENTER du clavier numérique.
Lorsqu'une chaîne est très longue, on peut la visualiser grâce à une barre de défilement accessible via les options *Vertical Scrollbar* ou *Horizontal Scrollbar* du choix *Visible Item* du menu surgissant.

Chaque caractère alphanumérique est défini par un nombre représentable sur un octet (U8), suivant une règle de correspondance internationale, le code ASCII (***A**merican **S**tandard **C**ode for **I**nformation **I**nterchange*), où figurent également des caractères non imprimables (caractères de commande régissant le transfert d'informations, touches spéciales du clavier, telles que Retour Chariot, Retour Arrière, etc …) invisibles en affichage normal de chaînes de caractères.
L'option *Hex Display* du menu surgissant sur la chaîne de caractères permet d'afficher, sous forme hexadécimale (00..FF) le contenu de l'octet correspondant à chaque caractère.
L'option *\Codes Display* du menu surgissant sur la chaîne de caractères, permet d'insérer dans un contrôleur chaîne des caractères non imprimables, souvent nécessaires dans une chaîne de commande. Tout caractère peut

alors être entré par son code hexadécimal précédé du caractère « \ » (de \00 à \FF), ou par une lettre minuscule dans le cas de certains caractères spéciaux (comme en C) :

\r	Pour un retour chariot	\0D
\n	Pour une fin de ligne	\0A
\t	Pour une tabulation	\09

C'est cette représentation symbolique des caractères spéciaux qui s'affiche lorsque ces caractères invisibles sont présents dans un indicateur chaîne bénéficiant de cette option.

Sur le diagramme

La terminaison du diagramme associée à une chaîne de caractères est un rectangle rose, à contour épais (pour un contrôleur) ou mince (pour un indicateur), contenant les trois caractères « abc », indiquant le type chaîne de caractères.

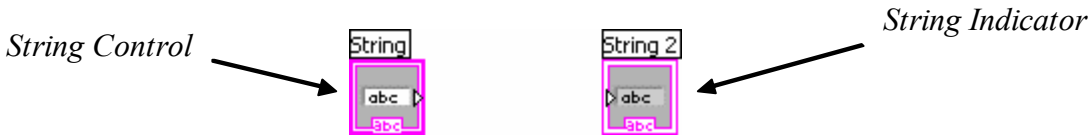

Figure 3-7 : Représentation de chaînes de caractères sur le diagramme

Les constantes alphanumériques

Elles sont accessibles sur le diagramme par le choix *String* de la palette *Functions*.

3.1.4 Les tableaux

Un **tableau** (*array*) est un ensemble d'éléments de même type (*integer*, *real*, *boolean*, etc.. Bref, tout type sauf « tableau »), ordonnés, de dimension définie et de taille variable.

LabVIEW fait grand usage de l'objet Tableau (*array*). Ce type de données existe dans la plupart des langages (Fortran, C, Pascal, …), mais les contrôleurs et indicateurs Tableau rendent ici la manipulation et l'affichage beaucoup plus aisés que dans ces langages. De plus, LabVIEW garde trace de la taille des tableaux et empêche tout débordement de mémoire. **L'utilisation de l'objet Tableau est recommandée avec LabVIEW, car le langage G possède des opérateurs et fonctions polymorphes travaillant sur les tableaux aussi bien que sur leurs éléments**.

Sur le panneau avant

Le contrôleur/indicateur Tableau, accessible en mode édition dans la sous-palette *Array, Matrix & Cluster* de la palette *Controls*, se présente sur le panneau avant sous forme d'un cadre vide de taille ajustable (destiné à recevoir l'objet définissant l'élément du tableau), à côté duquel apparaît de manière automatique un contrôleur d'indice, permettant de faire défiler les valeurs du tableau à l'écran.

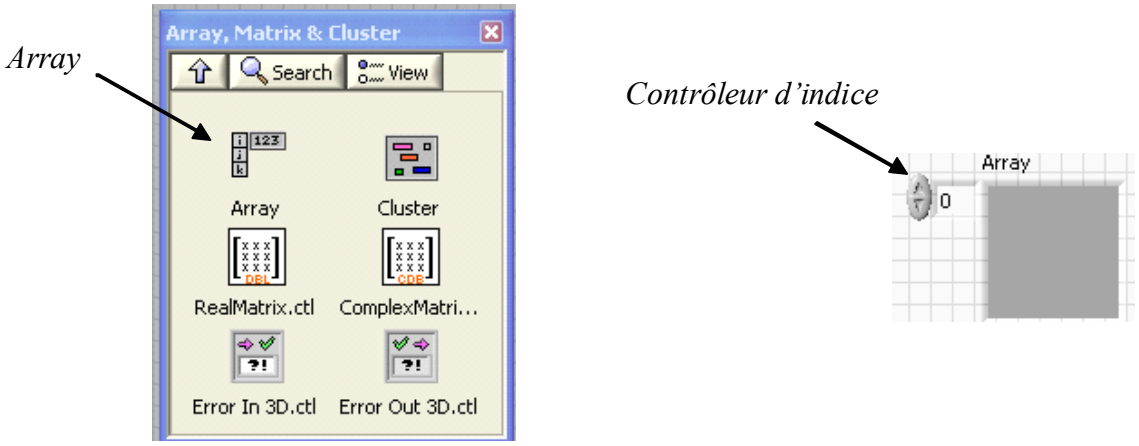

Figure 3-8 : Choix *Array, Matrix & Cluster* de la palette *Controls*

On définit le type du tableau en important dans son cadre (soit à l'aide de la palette *Controls*, soit en tirant un objet du panneau avant dans le cadre) un objet de type *Numeric*, *Boolean*, *String* ou *Cluster*, dont la catégorie (contrôleur ou indicateur) détermine celle du tableau lui-même. Par exemple, la figure ci-dessous montre ce que donne sur le panneau avant la création d'un tableau de contrôleurs réels et d'un tableau d'indicateurs booléens.

Figure 3-9 : Représentation de tableaux sur le panneau avant

On établit la dimension du tableau par extension du contrôleur d'indice (avec l'outil de positionnement placé sur un coin), ou par le choix *Add Dimension* du menu surgissant. La taille du tableau, fixée initialement par la(les) valeur(s) d'indice(s) la(les) plus élevée(s), peut être modifiée ultérieurement, et obtenue à partir de l'option *Show Last Element* du choix *Data Operations* du menu surgissant sur le contrôleur d'indice.

L'affichage de plusieurs éléments consécutifs d'un tableau est obtenu, à partir de l'élément initial défini par le (ou les) indice(s) affichés, en étirant le cadre. Les éléments apparaissant en grisé correspondent à des indices supérieurs à la taille du tableau. Il est important de noter que l'indice d'un tableau démarre à 0 et non à 1 !!!

Sur le diagramme

La terminaison associée à un tableau sur le diagramme affiche le type des éléments entre crochets, dans un rectangle à bord mince pour un indicateur ou épais pour un contrôleur. La figure ci-dessous correspond aux deux tableaux créés précédemment.

Figure 3-10 : Représentation de tableaux sur le diagramme

Il est important de noter que sur le diagramme, les fils de liaison transmettant des tableaux sont plus épais que les fils transmettant des variables simples. L'exemple ci-dessous montre

cette différence d'épaisseur de trait. Elle permet de distinguer rapidement sur le diagramme les tableaux de leurs éléments par exemple.

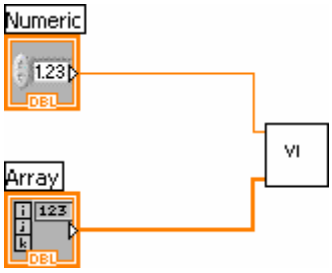

Figure 3-11 : Tableaux de réels et réels sur le diagramme. Noter la différence d'épaisseur des traits

Remarques

- Il n'est pas possible de créer des tableaux de tableaux, mais on peut contourner cette difficulté en créant des tableaux multidimensionnels, et si nécessaire en créant des tableaux d'agrégats (voir ci-dessous) de tableaux.
- L'initialisation des éléments d'un contrôleur tableau peut se faire manuellement par affichage et initialisation de chaque élément, ou par programme à l'aide de fonctions de la bibliothèque de LabVIEW spécifiques aux tableaux. La fonction *Initialize Array* permet d'initialiser un tableau dans le diagramme. Parmi les nombreuses fonctions de LabVIEW agissant sur les éléments des tableaux, deux sont particulièrement importantes : *Replace Array Element* permet de remplacer un élément et *Index Array* permet de l'extraire. Toutes ces fonctions sont accessibles par le choix *Array* de la palette *Functions*.

3.1.5 Les agrégats

Un **agrégat** (*Cluster*) est un ensemble de n éléments de types quelconques. Cet ensemble de taille définie est ordonné de 0 à n-1.

L'agrégat permet de regrouper plusieurs types différents dans une même donnée, tels les types *Record* en Pascal, ou *Struct* en C. En particulier, comme la programmation en langage G utilise des fils pour représenter le flot des données sur le diagramme, il est judicieux de rassembler dans un type agrégat des données de types différents destinées à suivre le même parcours, afin d'éviter l'emploi de nombreux fils parallèles. De plus, l'usage du type agrégat est recommandé pour manipuler en même temps des données étroitement associées. Nous l'emploierons par exemple :

- pour rassembler les coordonnées (X, Y) d'un point destiné à un affichage graphique
- pour rassembler les résultats d'une mesure physique destinée à être enregistrée dans un fichier (une tension mesurée et l'erreur sur la lecture par exemple).
- pour rassembler toutes les données correspondant à la calibration d'une carte d'acquisition (le nombre de voies à lire, leur numéro, leur gain, etc…)

Sur le panneau avant

Le contrôleur agrégat (accessible en mode édition par le choix *Cluster* de la sous-palette *Array & Cluster* de la palette *Controls*) se présente sur le panneau avant sous forme d'un cadre vide de taille ajustable, destiné à recevoir les objets constituant l'agrégat.

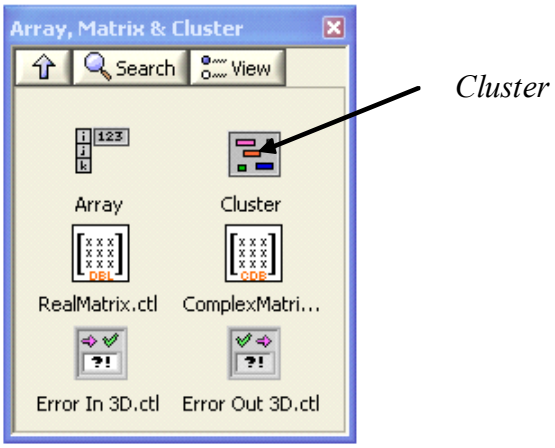

Figure 3-12 : Choix *Array, Matrix & Cluster* de la palette *Controls*

Comme pour construire un tableau, on importe dans ce cadre (à l'aide de la palette surgissant du cadre, ou en tirant un objet du panneau avant dans le cadre) des objets de types quelconques (*Numeric*, *Boolean*, *String*, *Array* ou *Cluster*), la catégorie (contrôleur ou indicateur) du premier objet importé déterminant celle des suivants et de l'agrégat lui-même.

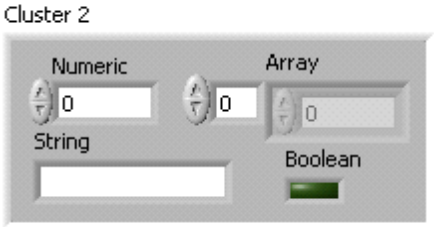

Figure 3-13 : Représentation d'un cluster sur le diagramme, comprenant un contrôleur réel, in tableau d'indicateurs réels, une chaîne de caractère et un booléen

Les constituants de l'agrégat sont ordonnés suivant leur ordre d'apparition dans le cadre (de 0 à n-1), mais l'utilisateur peut modifier cet ordre par le choix *Reorder Controls in Cluster* du menu surgissant sur le bord du cadre.

Sur le diagramme

La terminaison associée à l'agrégat sur le diagramme affiche un groupe de petits rectangles de tailles différentes, dans un rectangle à bord mince (indicateur) ou épais (contrôleur), de couleur marron si l'agrégat ne contient rien ou uniquement des contrôleurs numériques, ou de couleur rose s'il contient d'autres types de contrôleurs.

Figure 3-14 : Représentation de clusters sur le diagramme

Remarques

• L'ordre des constituants détermine sur le diagramme la position de leurs terminaisons respectives pour les fonctions d'assemblage (*Bundle*) ou de désassemblage (*Unbundle*) d'agrégats fournies par la bibliothèque

LabVIEW (accessibles directement par le choix *Cluster et Variant Palette* du menu surgissant sur la terminaison de l'agrégat dans le diagramme.

- Sur le diagramme, lors de l'utilisation des fonctions d'assemblage (*Bundle*) ou de désassemblage (*Unbundle*) les terminaisons affichent les types de tous les constituants de l'agrégat, exigeant que la composition de l'agrégat ne change pas d'une exécution à l'autre du programme.
- Par contre, lors de l'utilisation des fonctions *Bundle By Name* et *Unbundle by Name*, seules les terminaisons correspondant aux constituants choisis apparaissent et affichent leur nom, permettant ainsi à l'utilisateur de ne travailler que sur certains éléments d'un agrégat, pourvu que les éléments choisis soient présents.

3.1.6 Les graphiques

LabVIEW, dans son langage graphique G, offre à l'utilisateur des objets extrêmement performants, correspondant à des types de données numériques inexistants dans les autres langages, permettant l'affichage et la manipulation aisée des graphiques.

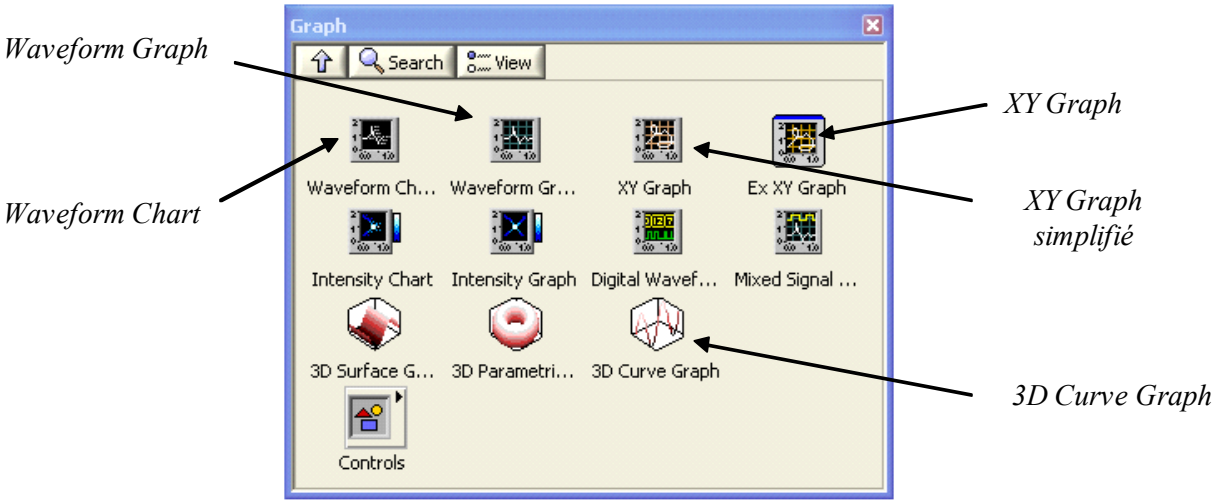

Figure 3-15 : Choix *Graph* de la palette *Controls*

La sous-palette des objets graphiques de LabVIEW est accessible sur le panneau avant, en mode édition, par le choix *Graph* de la palette *Controls*. Elle présente des afficheurs graphiques, répartis suivant 2 types d'objets (*Chart* et *Graph*) dont les propriétés essentielles sont :

Afficheurs *Chart*	Traitent l'affichage point par point et ne concernent que des tracés incrémentaux
Afficheurs *Graph*	Ne traitent que l'affichage de blocs de données Tracés incrémentaux (*Waveform Graph*), bidimentionnels (*XY Graph*) ou tridimensionnels (*3D Graph*)

A la création, ces afficheurs graphiques présentent sur le panneau avant une zone d'affichage de courbes, pourvue d'échelles verticale et horizontale, et proposent d'entrer immédiatement le nom (*Label*) du graphique à l'emplacement du curseur.

Des options par défaut sont fournies pour les échelles et le style d'affichage des points, mais l'utilisateur peut personnaliser le graphique grâce aux fenêtres accessibles par le menu surgissant.

3.1.6.1 Le *Waveform Chart*

C'est l'afficheur graphique le plus simple, qui permet d'observer l'évolution de plusieurs types de données numériques, sous la forme Y=f(i), où i est le numéro du point (axe des abscisses) et Y son ordonnée.

Sur le panneau avant

Les données reçues sont placées dans une mémoire tampon, associée au graphique (donc au programme dont le panneau avant contient l'afficheur). Une mise à jour de l'affichage **avec tracé complet du contenu de la mémoire tampon** s'effectue à chaque ajout d'une nouvelle donnée (d'où une certaine lenteur). La taille de la mémoire tampon (1024 points par défaut) peut être modifiée par le choix *Chart History Length* du menu surgissant.

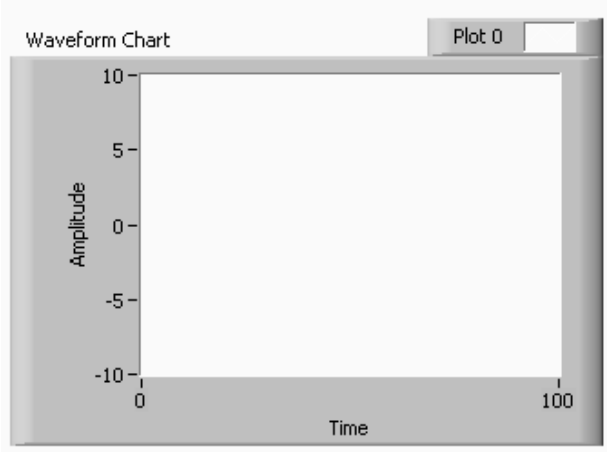

Figure 3-16 : Un *Waveform Chart* sur le « panneau avant » (le fond de l'écran apparaît normalement noir, mais il a été blanchi ici pour être facilement reproduit)

Sur le diagramme

La terminaison associée à l'afficheur graphique sur le diagramme est un rectangle dont l'intérieur va refléter le type des données fournies au graphique par le programme : nombre (entier ou réel), tableau de nombres, agrégat, tableau d'agrégats. Par exemple, le *Waveforme Graph* de la Figure 3-17 affichera des réels car le rectangle contient « DBL ».

Figure 3-17 : Un *Waveform Chart* sur le diagramme

Remarques

- Plus la taille de la mémoire tampon sera faible, plus le rafraîchissement de l'écran sera rapide.
- L'option *Digital Display* du choix *Show* du menu surgissant permet de visualiser la dernière valeur entrée sur l'indicateur digital associé à chaque courbe.
- L'option *Export Simplified Image …* du « panneau avant » peut être utile pour écrire un rapport !
- Il existe 3 modes de mise à jour de l'affichage, accessibles par l'option *Update Mode* du choix *Data Operations* du menu surgissant :

Strip Chart	C'est le mode par défaut, avec défilement des points et modifications des intervalles à chaque mise à jour
Scope Chart	C'est le mode le plus rapide, sans défilement ni modification des intervalles, avec effacement en bout de graphe et affichage des nouveaux points, comme sur un oscilloscope
Sweep Chart	La vitesse de ce mode est intermédiaire entre celle des deux précédents, sans défilement, mais avec une barre d'effacement verticale mobile, au niveau du dernier point (délimitant les nouvelles données)

3.1.6.2 Le *Waveform Graph*

C'est un afficheur graphique qui permet d'observer plusieurs types de données numériques, sous la forme Y=f(X), où l'abscisse X du point **varie par incrément fixe δX** à partir d'une origine X_0, et où Y est l'ordonnée.

Sur le panneau avant

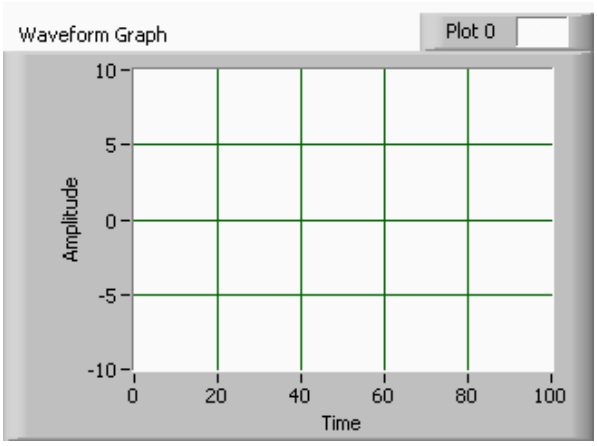

Figure 3-18 : Un *Waveform Graph* sur le panneau avant (comme pour le *Waveform Chart*, le fond de l'écran a été blanchit)

Sur le diagramme

La terminaison associée à l'afficheur graphique sur le diagramme est un rectangle dont l'intérieur va refléter le type des données fournies au graphique par le programme : nombre (entier ou réel), tableau de nombres, agrégat, tableau d'agrégats.

Les types de données acceptés par la terminaison *Waveform Graph* sur le diagramme sont très variés et font toujours intervenir un tableau. Voir l'aide en ligne pour comprendre comment connecter le tableau.

Remarque

- Une fenêtre spécifique de curseurs est accessible par l'option Cu*rsor Legend* du choix *Visible Items* de leur menu surgissant. Ces curseurs constituent un apport appréciable et leur facilité d'emploi rend plus aisé l'analyse des données affichées. Leur initialisation et leur lecture peuvent soit se faire directement à l'aide de la fenêtre, soit par programme.
- On peut directement envoyer un tableau sur un *Waveform Graph*, ce qui permet de s'en servir comme un *Waveform Chart*

3.1.6.3 Le *XY Graph*

C'est un afficheur graphique qui permet d'observer plusieurs types de données numériques, sous la forme de points définis par des couples de coordonnées (X,Y) quelconques.

Sur le panneau avan

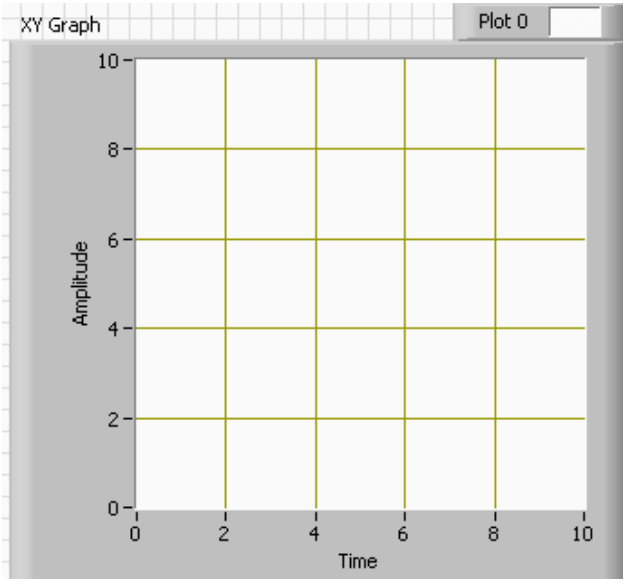

Figure 3-19 : Un *XY Graph* sur le panneau avant

Sur le diagramme

Les types de données acceptés par la terminaison *XY Graph* sur le diagramme sont très variés, font toujours intervenir au moins un agrégat et un tableau pour un tracé simple, et plusieurs pour des tracés multiples. Il existe une version simplifiée (*Express XY*) accessible par le choix *Graph* de la palette *Controls* (Figure 3-15).

Remarque

• Une fenêtre de curseurs (identique à celle du *Waveform Graph*) est également disponible par l'option *Cursor Legend* du choix *Visible Items* de leur menu surgissant. Cette fenêtre est en fait commune à tous les afficheurs de type « Graph ».

3.1.6.4 Le *3D Graph*

C'est un afficheur graphique qui permet de représenter des données numériques sous la forme de points définis par leurs coordonnées (X, Y, Z) quelconques. Son utilisation est triviale si on a compris celle des précédents *Graph*.

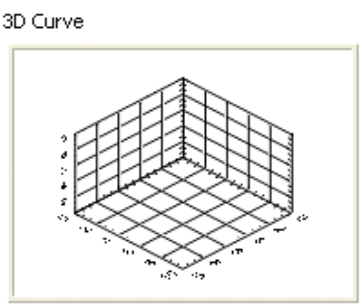

Figure 3-20 : Un *3D Curve Graph* sur le « panneau avant »

3.1.7 Les chemins d'accès aux fichiers

On a parfois besoin d'écrire sur disque certaines données numériques pour pouvoir les analyser ultérieurement. Pour cela, la création de fichiers est souvent indispensable. LabVIEW classe les fichiers de données en deux grandes catégories :

• Les fichiers *Byte Stream* sont des fichiers constitués de séquences d'octets où le pointeur d'écriture se déplace par octet. Cette catégorie comprend elle-même deux types de fichiers : les **fichiers texte** (*ASCII Byte Stream Files*) où chaque octet représente le code ASCII d'un caractère et les **fichiers binaires** (*Binary Byte Stream File*) où chaque octet correspond en général à plusieurs données.

• Les fichiers *Data Log* sont des fichiers binaires constitués de séquences d'enregistrement où le pointeur d'écriture se déplace par enregistrement.

Il est important de noter que les fichiers binaires ne sont lisibles que par un programme adéquat, tandis que les fichiers texte sont lisibles par n'importe quel logiciel de traitement de texte. L'avantage des fichiers binaires est qu'en général la taille qu'ils occupent sur le disque dur est beaucoup plus réduite que pour les fichiers texte. Ceci pourra être intéressant dans le cas de très gros fichiers, ou lorsqu'on doit sauvegarder des données en un temps très court. Pour de petits fichiers, le type texte est plutôt recommandé.

Sur le panneau avant

Les fichiers de données doivent avoir un nom conforme au système d'exploitation (les caractères tels que &, %, @, etc… sont par exemple interdits). LabVIEW permet d'accéder à ces divers types de fichiers, grâce à des contrôleurs/indicateurs spécifiques aux fichiers :

• Le contrôleur *File Path* (accessible via la sous-palette *String & Path* de la palette *Controls*) contient le chemin d'accès complet au fichier, dans l'arborescence du système de fichiers de l'ordinateur. L'information issue de ce contrôleur est nécessaire pour ouvrir ou créer un fichier dans un programme à l'aide des fonctions ou sous-programmes utilitaires de LabVIEW.

• Les contrôleurs *File Refnum* (accessible via la sous-palette *Refnum* de la palette *Controls*) contiennent le numéro de référence du fichier ouvert, attribué par le système d'exploitation dès l'ouverture d'un fichier. Ce numéro de référence est le seul moyen d'accès au fichier après son ouverture (par exemple si l'on souhaite lire ou écrire dans le fichier). Il existe deux types de contrôleurs *File Refnum* correspondant aux deux catégories de fichiers de données : le *Byte Stream File Refnum* sert à accéder aux fichiers *Byte Stream* et le *Data Log File Refnum* sert à accéder aux fichiers *Data Log*.

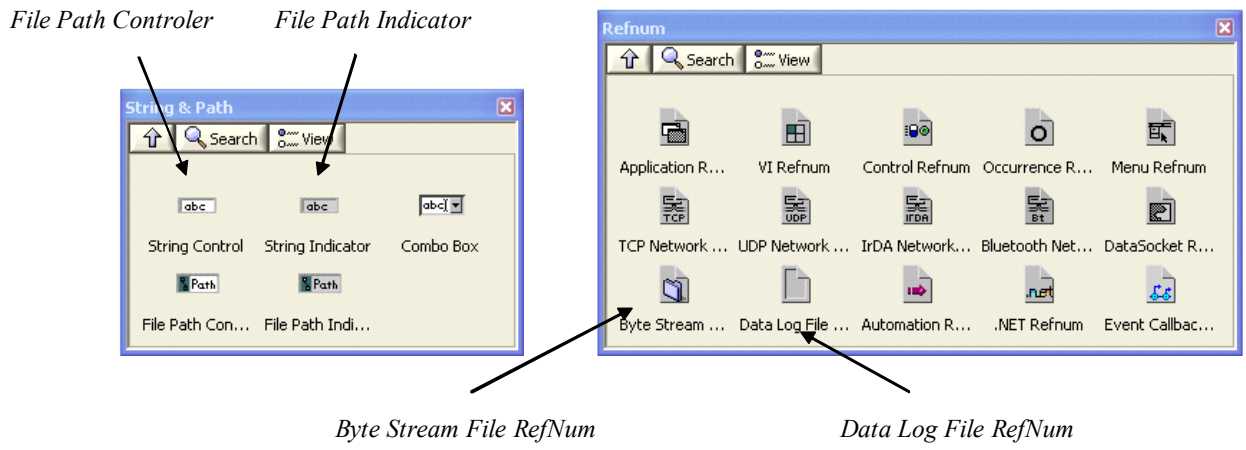

Figure 3-21 : Choix *Path & RefNum* de la palette *Controls*

Sur le diagramme

Les *Path* et *RefNum* décrits précédemment sont représentés sur le diagramme par les symboles représentés sur la figure suivante.

Figure 3-22 : Représentations de *Path* et de *RefNum* sur le diagramme

Constantes

Il existe dans LabVIEW des constantes de type *RefNum* accessibles par le choix *File I/O* de la palette *Functions*.

Remarques

- LabVIEW permet de sauver automatiquement l'ensemble des données du panneau avant d'un programme (*Data Logging* accessible par le menu *Operate*). Ceci est à manipuler avec la plus grande précaution car les risques de saturation de mémoire vive sont importants…
- Dans le cas d'un *Data Log*, les variables de type *RefNum* contiennent le type des données des enregistrements, afin de vérifier l'accord entre ce type et celui des données fournies ou extraites par fonctions élémentaires d'entrée/sortie de LabVIEW. La description du type de données contenues dans les enregistrements d'un fichier *Datalog File* est fournie par un agrégat (*Cluster*) contenant les divers constituants de l'enregistrement placé dans le cadre du *Data Log File Refnum*.

3.2 Les objets ou concepts présents uniquement sur le diagramme

Ces objets sont accessibles à partir de la *Functions Palette* du diagramme.

3.2.1 Les divers types de variable

Comme tout langage de programmation évolué (Fortran, C, ..), le langage G permet l'accès direct (aussi bien en lecture qu'en écriture) aux données contenues dans un contrôleur du panneau avant grâce au concept de « variable locale » et de « variable globale ». Une variable sera dite « **locale** » si, lors de l'utilisation du VI comme sous-programme, la variable n'est connue qu'à l'intérieur de ce sous-programme. Elle sera dite « **globale** » si sa valeur est également connue à l'extérieur du sous-programme.

L'utilisation de ces variables permet de simplifier notablement les diagrammes, en réduisant le nombre de fils de terminaisons nécessaires.

3.2.1.1 Les variables locales

Les données correspondant à une variable locale sont stockées dans un des contrôleurs du panneau avant du VI dans lequel la variable est définie.

On peut créer une variable locale à partir d'un contrôleur existant par le choix *Create Local Variable* du menu surgissant sur cet objet. La terminaison de la variable locale apparaît alors sur le diagramme sous la forme d'un rectangle de couleur correspondant au type du contrôleur

choisi et contenant son nom comme indiqué sur la figure ci-dessous pour un entier, une chaîne de caractères et un booléen.

Figure 3-23 : Terminaisons de variables locales sur le diagramme

On utilise alors la variable locale comme une variable « normale », en la reliant aux autres nœuds du programme à l'aide des terminaisons ad hoc.

Ces variables ont l'avantage de pouvoir se soustraire de la notion de "flux d'information" qui est la base de la programmation LabView. Elles permettent notamment :

– d'envoyer de l'information à un contrôleur, ou de récupérer de l'information d'un indicateur ;

– de faire sortir une information d'un nœud (une boucle par exemple) avant la fin de l'exécution de ce nœud.

Remarques

• La catégorie de la terminaison (écriture ou lecture) est modifiable par l'option *Change to read Local* (pour une lecture) ou *Change to Write Local* (pour une écriture) du menu surgissant.

• Un autre moyen de créer une terminaison de variable locale est le choix de *Local Variable [LOCAL]* dans la sous-palette *Structures* de la palette *Functions*.

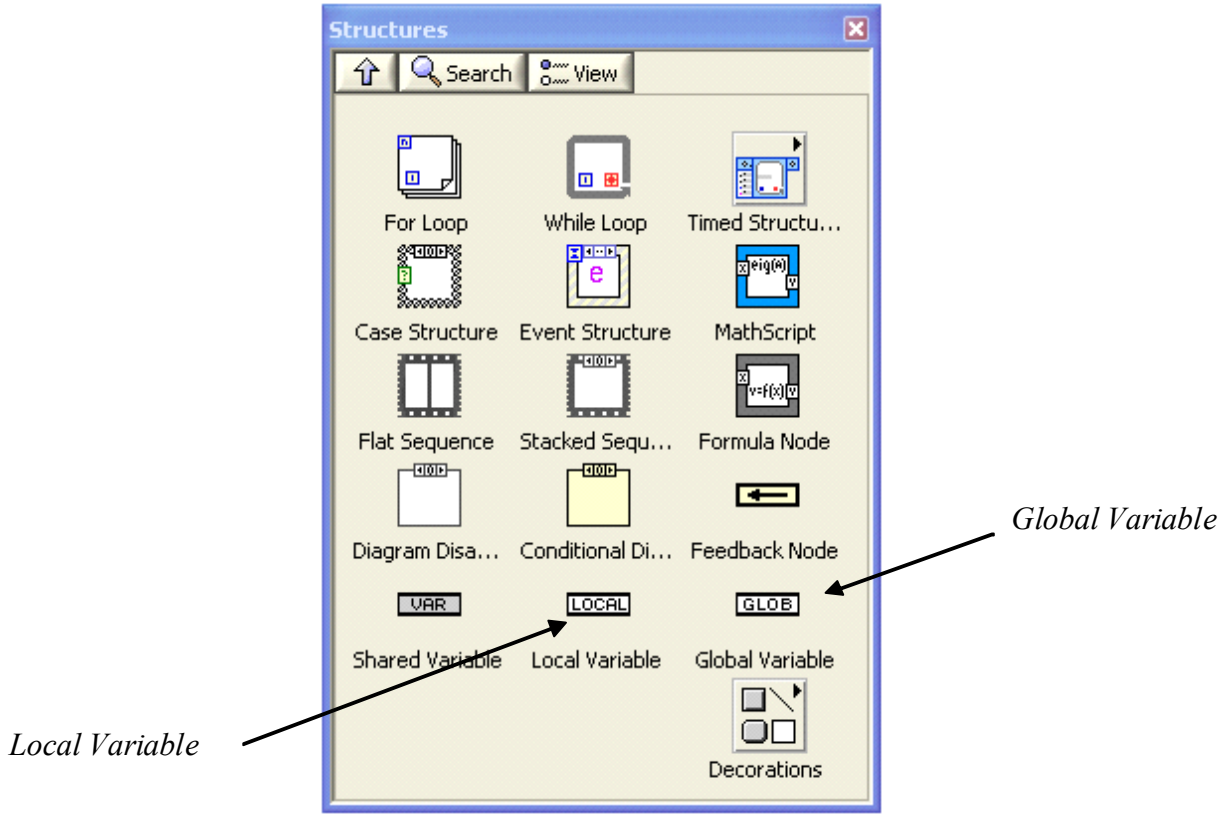

Figure 3-24 : Choix *Structures* de la palette *Functions*

La terminaison rectangulaire qui apparaît porte le nom du premier objet introduit sur le panneau avant ou la forme décrite sur la figure ci-contre si le panneau est encore vide.

Le nom de la variable locale peut alors être choisi dans la liste des contrôleurs du panneau avant accessible par l'option *Select Item* du menu surgissant sur la terminaison. Le nom du contrôleur sélectionné apparaît à l'intérieur du rectangle de la variable locale dont le contour prend la couleur correspondant au type du contrôleur

Figure 3-25 : Terminal d'une variable locale non affectée

3.2.1.2 Les variables globales

Les données correspondant à une variable globale sont stockées dans un des contrôleurs du panneau avant d'un VI spécial destiné à contenir des objets accessibles à tous les sous-programmes.

Le choix *Global Variable [GLOB]* de la sous-palette *Structure* de la palette *Functions* (Figure 3-24) fait apparaître sur le diagramme la terminaison rectangulaire ci-contre.

Figure 3-26 : Terminal d'une variable globale non affectée

Le choix *Open Front Panel* du menu surgissant sur cette terminaison fait apparaître le panneau spécifique mentionné ci-dessus (de nom *Global i*) destiné aux variables globales. Il ne comporte pas de diagramme mais uniquement un panneau avant.

L'accès aux variables globales peut alors se faire par le choix *Select a VI…* de la palette *Functions*, parmi les contrôleurs du panneau de variables globales, grâce à l'option *Select Item* du menu surgissant sur la terminaison rectangulaire.

Les variables globales permettent de partager de l'information entre différentes vi en s'affranchissant des contraintes de flux.

Remarque

- La catégorie de la terminaison (écriture ou lecture) est modifiable par l'option *Change to read Global* (pour une lecture) ou *Change to Write Global* (pour une écriture) du menu surgissant.

3.2.2 Les constantes

Il existe toute une série de constantes pouvant être employées à la place d'un contrôleur si la valeur qui lui est associée doit rester constante. Elles sont accessibles dans les diverses palettes de la palette Functions et ont pour certaines déjà été décrites dans ce manuel. La liste non exhaustive de ces constantes est donnée ci-dessous.

Liste des principales constantes disponibles :

Constantes numériques	e, π, h, ..
Constantes chaîne de caractères	Retour Chariot, Fin de Fichier, ..
Constantes pour la gestion des fichiers	*Not a Path*, *Not a RefNum*, *Empty Path*
Codes d'erreur	Valeurs numériques correspondant à des erreurs particulières (dépassement de fin de fichier, ..)

3.2.3 Les divers types de nœuds

La figure ci-dessous regroupe les divers types de nœuds accessibles par le choix *Structures* de la palette *Functions*.

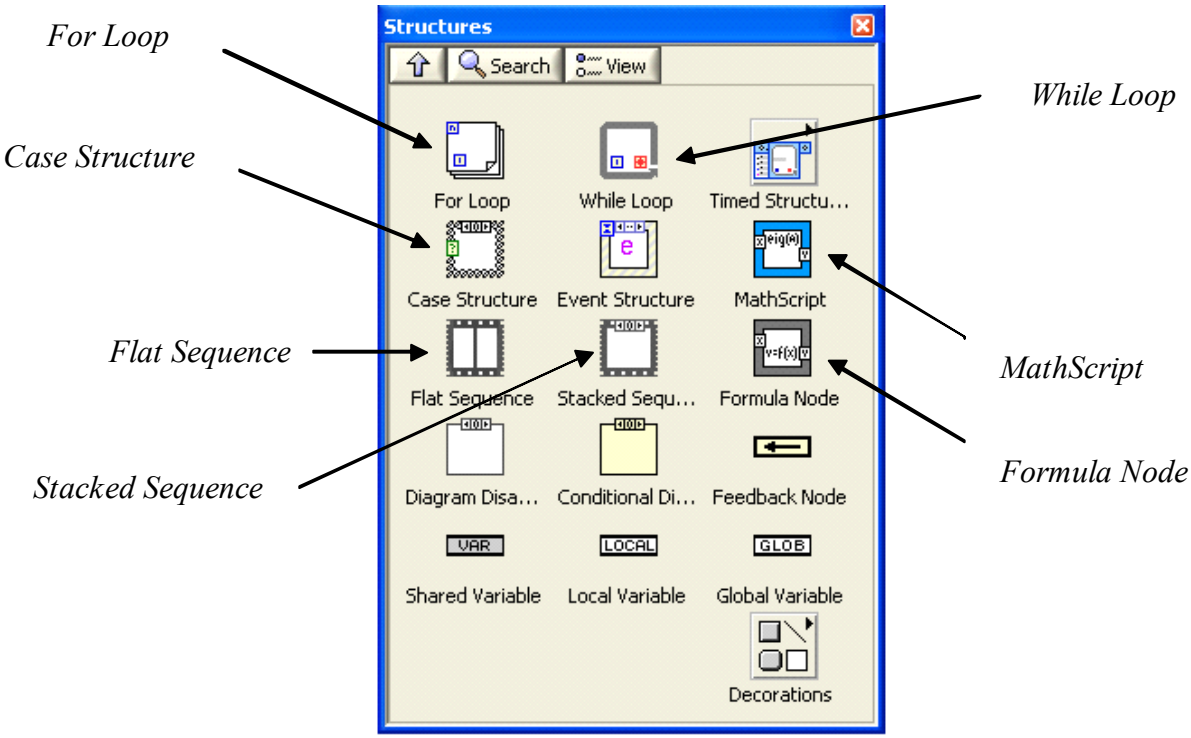

Figure 3-27 : Choix *Structures* de la palette *Functions*

3.2.3.1 Le nœud pour formules (*Formula Node*)

La programmation graphique du langage G se prête mal aux calculs faisant de nombreux appels à des fonctions élémentaires. C'est pourquoi le langage G propose le **nœud pour formules** (*Formula Node*) permettant d'exécuter des calculs présentés sous la forme d'instructions écrites en C.

Le choix *Formula Node* de la palette *Structures* (cf Figure 3-27) propose un curseur rectangulaire pointillé contenant les caractères « fx » que l'on positionne sur le diagramme en cliquant et étirant en glissant la souris. Il apparaît alors sous la forme d'un cadre rectangulaire vide comme indiqué sur la Figure 3-28.

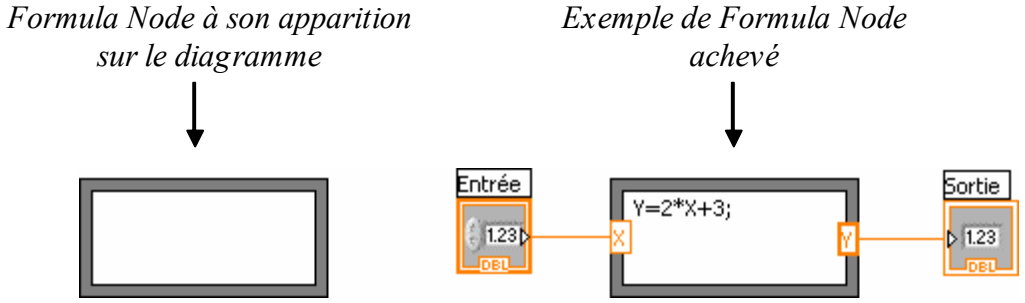

Figure 3-28 : Terminaison du *Formula Node*

Il est alors nécessaire d'affecter une terminaison à chaque variable d'entrée (choix *Add Input* du menu surgissant sur le bord du cadre) et de sortie (choix *Add Output*) utilisée dans la formule, en situant de préférence les entrées sur le bord gauche et les sorties sur le bord droit. Chaque terminaison est représentée par un rectangle (à bord mince pour une entrée, à bord épais pour une sortie) à l'intérieur duquel il faut écrire le nom de la variable correspondante. Enfin, la formule doit être écrite dans le cadre suivant la syntaxe du C (en terminant chaque ligne par « ; » notamment). Tous les opérateurs numériques et logiques du C peuvent être utilisés dans l'écriture de la formule.

Dans l'exemple de la figure ci-dessus, X est la variable d'entrée et Y la variable de sortie.

Remarques

- En cas d'erreur, le menu surgissant permet de changer la catégorie d'une terminaison (choix *Change to Output/Input*) ou de la supprimer (choix *Remove*).
- Les calculs seront effectués en nombres réels avec la plus grande précision disponible sur le PC.
- Il existe une 2ème possibilité de faire des calculs symboliques. Elle utilise la structure *MathScript* (Figure 3-27). L'avantage de cette structure est qu'elle permet de manipuler des tableaux et qu'elle utilise le langage MathLab™.

3.2.3.2 La structure séquentielle (*Stacked Sequence Structure*)

On est parfois amené à exécuter une partie d'un programme avant une autre (par exemple pour initialiser une carte d'acquisition avant de l'utiliser). Pour cela, la structure séquentielle permet de définir un ordre dans le déroulement du programme. On y accède par le choix *Sequence* de la palette *Structures* (Figure 3-27). Elle est formée d'un cadre ressemblant à de la pellicule photographique que l'on étire de la même façon que le *Formula Node* décrit précédemment.

Figure 3-29 : Terminaison de la structure *Stacked Sequence*

L'intérieur du cadre est un sous-diagramme destiné à recevoir la partie du programme devant s'exécuter dans la séquence. On ajoute les séquences successives par le choix *Add Frame Before/After* ou *Duplicate Frame* du menu surgissant sur le bord du cadre. Dès que la structure comporte plus d'une séquence, la partie supérieur du cadre porte un rectangle d'index contenant le numéro de la séquence affichée (0 pour la 1ère séquence).

Comme un seul sous-diagramme à la fois est visible sur l'écran du PC, les boutons de changement de séquence sont nécessaires pour visualiser l'ensemble des séquences.

On peut supprimer la séquence affichée par le choix *Delete This Frame* du menu surgissant, ou la structure complète par le choix *Remove Sequence*.

Variables locales aux structures séquentielles : les Sequence Locals

Si l'on doit communiquer une variable d'un sous-diagramme à un sous-diagramme suivant, des terminaisons correspondant à des variables locales (*Sequence Locals*) peuvent être créée par le choix *Add Sequence Local* du menu surgissant sur le bord du cadre (une flèche indique le sens de passage des données dès qu'une liaison est établie). Les données des tunnels d'entrée sont disponibles dans tous les sous-diagrammes, mais une seule séquence est autorisée à fournir des données à un tunnel de sortie. Les données d'un tunnel de sortie ne sont disponibles pour le reste du programme qu'à l'issue du déroulement de toutes les séquences. Dans l'exemple de la figure ci-dessous, on ajoute 2 à un nombre avant de l'envoyer à un 1er VI. La réponse de ce VI est alors envoyée à un 2ème VI.

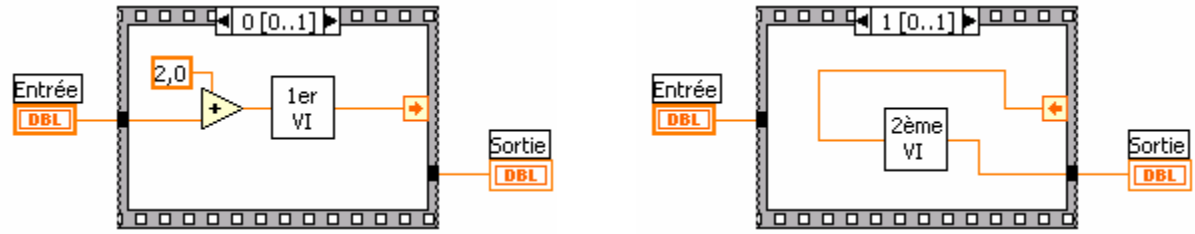

Figure 3-30 : Exemple d'utilisation d'une structure *Stacked Sequence*

Remarques

- L'utilisation des variables locales et globales dans les séquences permet de simplifier notablement les diagrammes, en réduisant le nombre de fils passant d'un niveau à un autre.
- Si l'on doit exécuter deux VI l'un après l'autre, au lieu de les mettre dans deux étapes consécutives d'une même séquence, on peut tirer un fil de données du 1er VI vers le 2ème. L'exécution du 2ème VI ne se fera que lorsque celle du 1er sera terminée. Même lorsqu'on ne transmet rien sur ce fil de données…

Il existe un 2ème type de structure séquentielle (*Flat Sequence*) que l'on accède de la même façon (Figure 3-27). La seule différence est de pouvoir visualiser en même temps le contenu de tous les sous-diagrammes. Son utilisation doit être réservée aux séquences comportant peu d'instruction.

3.2.3.3 La structure de choix (*Case Structure*)

Cette structure est l'équivalent en langage G des structures *if(condition) then .. else .. endif* en Fortran ou *if(condition){..}else{..}* en C, ainsi que de la structure *case* en C. On est amené à l'utiliser lorsque le programme dépend d'une condition.

On y accède par le choix *Case* de la sous-palette *Structures* de la palette *Functions* (cf Figure 3-27). Elle est formée d'un cadre rectangulaire que l'on étire de la même façon que le *Formula Node* et les *Sequence* décrits précédemment. La partie supérieure du cadre porte un rectangle d'index contenant l'indicateur de cas (*False* par défaut) et des boutons de changement de cas, tandis que le bord gauche porte une terminaison de sélection (contenant un « ? » par défaut). Par défaut, la terminaison de sélection est de type booléen, tandis que la structure comporte deux sous-diagrammes qui correspondent aux deux états d'un booléen (*False* et *True*). L'intérieur du cadre est un sous-diagramme destiné à recevoir la partie du programme correspondant au cas affiché.

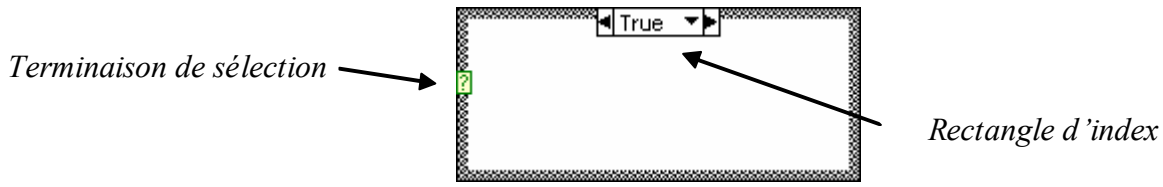

Figure 3-31 : Terminaison de la structure *Case*

Les données des tunnels d'entrée sont disponibles dans tous les sous-diagrammes. **S'il existe un tunnel de sortie, tous les sous-diagrammes doivent charger cette variable**.

Comme la structure *Case* comporte toujours plusieurs sous-diagrammes mais qu'un seul est visible à la fois sur l'écran, les boutons de commande de changement de cas permettent de changer de sous-diagramme.

On peut supprimer le cas affiché par le choix *Delete This Case* du menu surgissant ou la structure complète par le choix *Remove Structure*.

Remarques

- Si l'on envoie un nombre vers la terminaison de sélection (au lieu d'un booléen), elle prend un type numérique et le rectangle d'index affiche alors le numéro du cas. On peut alors créer plus de deux cas en ajoutant des cas successifs par le choix *Add Case* ou *Duplicate Case* du menu surgissant sur le bord du cadre. Comme LabVIEW arrondit les valeurs de sélection et limite lui-même ces valeurs au nombre de cas, il est sage de prévoir un cas spécial pour dépassement de limite.
- Si l'on envoie vers la terminaison de sélection un nombre de type *Enumération*, le rectangle d'index affiche la chaîne de caractères correspondante au lieu du numéro du cas (ce qui peut faciliter l'identification des sous-diagrammes). On a alors l'équivalent de la structure *case* du C.

3.2.3.4 La boucle inconditionnelle POUR (*For Loop*)

Comme en Fortran ou en C, c'est une structure itérative permettant d'exécuter une partie de programme un nombre déterminé de fois. On l'utilise lorsque l'on connaît le nombre d'itérations à exécuter.

Le choix *For Loop* de la sous-palette *Structures* (Figure 3-27) propose un curseur rectangulaire pointillé contenant les caractères « Ni » que l'on positionne sur le diagramme en cliquant et étirant en glissant la souris. Il se présente finalement sous la forme d'un cadre rectangulaire vide contenant par défaut deux terminaisons comme indiqué sur la figure ci-dessous. L'intérieur du cadre est un sous-diagramme destiné à être exécuté N fois.

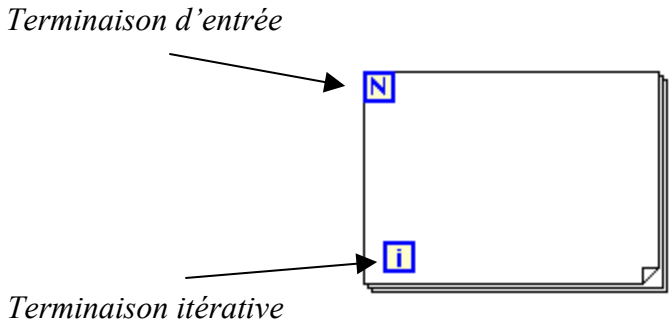

Figure 3-32 : Terminaison de la boucle *For*

La terminaison d'entrée reçoit le nombre d'itération à exécuter (nombre entier compris entre 0 et 2^{31}-1). La terminaison itérative fournit dans la boucle le numéro de l'itération en cours (nombre entier compris entre 0 et N-1).

La boucle *For* autorise par défaut l'indexation automatique (*Auto Indexing*) : c'est la possibilité d'indexer et de stocker automatiquement des tableaux dans les tunnels situés sur le bord de la structure. Il faut donc traiter avec attention (ou mieux éviter) le cas N=0 fournissant des tableaux vides.

Variables locales aux boucles For : les Shift Registers

Si l'on a besoin de transmettre la valeur d'une variable d'une itération de la boucle à une autre, on peut utiliser des variables locales à la boucle *For* appelées registres à décalage (*Shift Registers*). Elles sont introduites par le choix *Add Shift Register* du menu surgissant sur le bord de la structure. Il apparaît alors deux terminaisons, celle de droite recevant la valeur de la variable en fin d'itération pour la transmettre à celle de gauche qui fournira la valeur de la variable au sous-diagramme au début de l'itération suivante.

L'initialisation du registre à décalage (primordiale pour éviter les ennuis..) se fait en reliant une variable (ou une constante) à la terminaison de gauche. C'est cette valeur qui sera prise lors du 1er passage dans la boucle.

Dans l'exemple de la figure ci-dessous, on utilise une boucle for pour calculer la somme des N premiers nombres entiers.

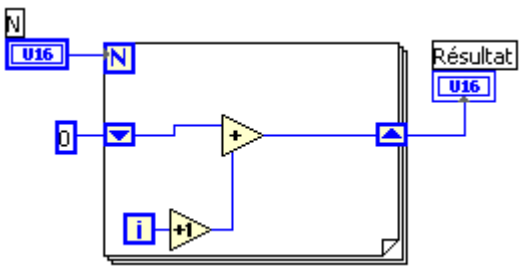

Figure 3-33 : Exemple d'utilisation d'une boucle *For* : calcul de la somme des N premiers nombres entiers

On peut supprimer une boucle *For* (en laissant son contenu sur le diagramme) par le choix *Remove For Loop* de son menu surgissant.

Remarque

• On peut supprimer l'indexation automatique par le choix *Disable Indexiong* du menu surgissant sur le tunnel. Cela permet de transmettre au reste du programme la dernière valeur d'une variable, et non le tableau de ces valeurs à chacun des N passages dans la boucle.

3.2.3.5 La boucle conditionnelle TANT QUE (*While Loop*)

C'est une structure permettant d'exécuter une partie de programme tant qu'une condition reste vraie. Elle correspond à la boucle *do ... while (condition)* du Pascal par exemple. On l'utilise par exemple lorsqu'on effectue des mesures, sans savoir a priori quand on doit s'arrêter.

On y accède par le choix *While Loop* de la sous-palette *Structures* de la palette *Functions* (cf Figure 3-27). Elle est formée d'un cadre rectangulaire que l'on étire de la même façon que les boucles décrites précédemment. L'intérieur du cadre est un sous-diagramme destiné à recevoir la partir du programme à exécuter tant que la condition de sortie le permet.

Figure 3-34 : Terminaison de la boucle *While*

La boucle *While* contient deux terminaisons par défaut : une terminaison itérative fournissant à l'intérieur de la boucle le numéro de l'itération en cours (0 pour la 1ère) et une terminaison conditionnelle, testée à la fin de chaque tour de boucle et provoquant l'arrêt des itérations et la sortie de la boucle si son état est *False*.

Par défaut, la boucle *While* n'indexe pas les tableaux mais fournit aux tunnels de sortie les données de la dernière itération (c'est donc le contraire de la boucle *For*). Néanmoins, ce défaut peut être modifié par l'usage du choix *Enable Indexing* du menu surgissant sur le tunnel.

On peut supprimer une boucle *While* (en laissant son contenu sur le diagramme) par le choix *Remove While Loop* de son menu surgissant.

Variables locales aux boucles While : les Shift Registers

Pour transmettre les valeurs d'une variable d'une itération à la suivante, on peut utiliser des variables locales à la boucle *While* appelées registres à décalage (*Shift Registers*), comme dans le cas de la boucle *For*. Elles sont introduites par le choix *Add Shift Register* du menu surgissant sur le bord de la structure. Il apparaît alors deux terminaisons, celle de droite recevant la valeur de la variable en fin d'itération pour la transmettre à celle de gauche qui fournira la valeur de la variable au sous-diagramme au début de l'itération suivante.

L'initialisation du registre à décalage (primordiale pour éviter les ennuis..) se fait en reliant une variable (ou une constante) à la terminaison de gauche. C'est cette valeur qui sera prise lors du 1er passage dans la boucle.

Remarques

- Par défaut, la boucle s'effectue une seule fois si la terminaison conditionnelle n'est pas reliée.
- Par contre, elle s'effectuera de manière ininterrompue si la terminaison est reliée à un booléen dans l'état *True*
- Un moyen simple de contrôler le déroulement d'une boucle *While* à partir du panneau avant consiste à introduire dans son sous-diagramme une autre boucle *While* ne contenant qu'un contrôleur booléen (dont on a défini l'action mécanique, par exemple à l'aide de *Latch When Pressed*) relié à sa terminaison conditionnelle. Chaque pression sur le bouton permet le passage à l'itération suivante de la boucle principale. Il reste simplement à trouver un moyen de sortir de la boucle principale …
- En cas d'indexation automatique (*Auto Indexing*), le nombre d'itérations n'étant pas limité, la taille d'un tableau d'entrée peut être dépassée et les valeurs lues au-delà seront les valeurs par défaut du tableau. C'est pourquoi il est déconseillé d'utiliser des tableaux en entrée d'une boucle *While* en indexation automatique. De même, toujours en cas d'indexation automatique, la taille d'un tableau de sortie augmente tant que la boucle continue. Il faut donc veiller à prévoir des limites d'arrêt raisonnables, pour ne pas saturer la mémoire du PC par exemple. En résumé, l'indexation automatique est potentiellement source d'ennuis dans une boucle *While*. Il faut donc l'éviter dans la mesure du possible ou à défaut faire extrêmement attention…

4 Quelques trucs indispensables pour survivre

Ce chapitre fourre-tout recense le florilège de certains « trucs » et recettes à utiliser en cas de blocage, à utiliser avant d'appeler l'enseignant …

Programme non exécutable

Si la flèche *Run* est brisée comme indiqué sur la figure ci-dessous, cela signifie que votre programme ne peut pas fonctionner. Dans un langage tel que le Fortran ou le C, cela correspondrait à une erreur de compilation ou de construction de l'exécutable.

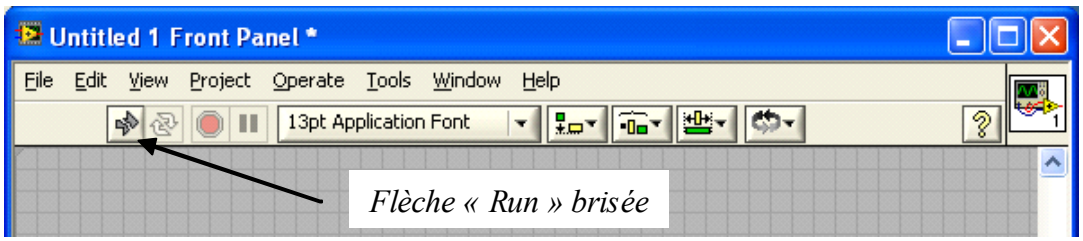

Figure 4-1 : Barre de menus avec un programme non exécutable

• Pour trouver l'erreur, il suffit de cliquer sur la flèche brisée : la liste des erreurs apparaît. Double-cliquez sur l'une d'elle et vous atteindrez directement l'erreur (il est conseillé de corriger les erreurs dans l'ordre car une erreur peut dépendre des erreurs précédentes).

• Si c'est un VI qui est encadré, cela peut signifier que vous avez modifié le connecteur d'entrée/sortie de ce sous-programme. Il faut alors le réinstaller dans le programme qui l'appelle (par le choix *Select a VI …* de la palette *Functions*).

• Si vous ne voyez pas le fil brisé (bien que LabVIEW vous dise que votre erreur vient de là), il est sûrement caché derrière un objet. L'option *Remove Bad Wires* (CTRL-B) du menu *Edit* peut être utile.

Un fil ne se connecte pas

• C'est vraisemblablement que vous cherchez à connecter un indicateur vers un contrôleur (ou le contraire) alors qu'il faut faire l'inverse : un fil sert à envoyer des données d'un contrôleur vers un indicateur.
⇒ L'option *Change to Control/Indicator* du menu surgissant sur l'objet peut être utile pour changer le type de la variable

Impossible de modifier un diagramme

• Une 1ère possibilité est que le VI soit en mode d'exécution. Il faut alors utiliser l'option *Change to Run/Edit Mode* du menu *Operate*
• Il est également possible que le VI soit verrouillé. Pour le déverrouiller, utiliser l'option *Show VI Info …* du menu *Windows*, puis *Unlocked*.

Remarque

• On peut utiliser le verrouillage lorsqu'on veut empêcher toute modification d'un VI particulièrement délicat à mettre au point .. Par exemple, les VI d'acquisition qui vous sont fournis sont verrouillés.

5 Quelques notions rudimentaires pour utiliser KaleidaGraph™

Dans tous les cas, l'aide en ligne est bien faîte, prendre l'habitude de la consulter avant de demander à l'enseignant.

- **Importer un data ascii** : Choisir *File\Import\Text* puis OK. On peut importer un fichier à plusieurs colonnes ainsi.

- **Tracer un graphe X,Y** : Choisir *Gallery\Linear\Scatter* ou *Line* selon le style voulu, puis choisir quelle colonne on trace en fonction de quelle autre. Pour ajouter un autre graphe, procéder de même depuis le graphe initial. Pour modifier l'aspect du graphe, double-cliquer sur la légende associée au data à changer avec le curseur flèche. Pour changer les axes ou le titre, double-cliquez dessus.

- **Ajouter des barres d'erreur** : sur le graphe, *Plot\Error Bars*, cliquer sur *Yerr* si on veut par exemple une barre d'erreur en Y, puis choisir soit un pourcentage de la valeur, soit, si on a une colonne qui contient la barre d'erreur, choisir cette colonne en cliquant sur l'onglet horizontal, puis *data column*, puis choisir la colonne.

- **Manipuler le graphe :** on utilisera les outils. L'outil avec deux carrés blancs imbriqués permet de faire des zooms, et en le double-cliquant de revenir au graphe de départ. L'outil avec des carrés et des droites dedans doit être utilisé seulement avec précaution, car il permet de sélectionner une partie du data et d'exclure le reste.

- **Générer une colonne à partir d'une formule mathématique :**
  On génère d'abord un axe X en sélectionnant une colonne, puis en utilisant *Functions\Create Series*.
  On entre ensuite depuis la fenêtre *Formula Entry* (appelée via Ctrl F) la formule qui génère l'axe Y : par exemple, si on veut 2 cos(X), que les colonnes X et Y sont la numéro 0 et numéro 1, on tape la formule c1=2*cos(c0) puis *RUN*.

- **Faire une opération mathématique sur une colonne :**
  On entre depuis la fenêtre *Formula Entry* (appelée via Ctrl F) la formule à appliquer. Par exemple, si on veut additionner deux colonnes placées en 0 et 1, on tape c2=c0+c1 puis *RUN*.

- **Faire un ajustement (fit) :**
  - *pour un ajustement avec une des fonctions prédéfinies* (linéaire, polynôme, exponentiel, logarithmique, loi de puissance), on se place sur le graphe où le data est tracé. On clique sur *Curve Fit* puis la fonction choisie. On sélectionne ensuite le data à fitter. Le fit est automatiquement tracé. Pour visualiser les paramètres correspondant, revenir sur *Curve Fit*, sélectionner la fonction, cliquer sur *View\Results*. Puis cliquer sur *Clipboard*. On peut alors les visualiser via *Windows\clipboard*.
  - *Pour un ajustement avec une fonction quelconque*, cliquer sur *Curve Fit\General\Edit General* puis *Add*, puis sélectionner en haut à droite la fonction

New Fit, la renommer en bas à gauche, puis *Edit*. Entrer une formule du type (ici comme exemple, un fit avec une lorentzienne) : m4 + m1/(((m0 - m2)/m3)^2 + 1) ; m1=1 ; m2=50 ; m3=10; m4 = 0. Attention, le programme utilise la notation m0 comme variable (c'est-à-dire x), et utilise la notation m1, m2, …, m9 comme paramètres ajustables (ce sont les valeurs de ces paramètres que kaleidagraph va essayer d'optimiser – m0 est de nature totalement différente). Dans cette formule, m4 est une ligne de base, m1 l'amplitude, m2 le centre, m3 la mi-largeur. Il faut fournir au logiciel des valeurs initiales afin qu'il puisse commencer sa procédure d'optimisation. Il faut choisir ces valeurs le plus proche possible de la réalité pour permettre au fit de converger. Une fois cette formule définie, le fonctionnement est le même que pour un fit avec fonction prédéfinie.