RÉPUBLIQUE ALGÉRIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement supérieur et de la recherche scientifique

Université Amar Telidji de Laghouat

Faculté de Technologie
Département d'Électrotechnique

Dr. OUBBATI Youcef

Concepts et langage de programmation graphique

Support du Cours

2019-2020

Table des matières

Table des figures

Liste des tableaux

Première partie

Objectifs du module

L'objectif de l'enseignement de cette matière est de fournir à l'étudiant les concepts et langage de programmation graphique d'une manière générale :

— Présenter LabVIEW et ses fonctionnalité ;
— Comprendre les composants d'un Instrument Virtuel (appelé VI) ;
— Créer un sous-programme dans LabVIEW ;
— Travailler avec les tableaux, les graphiques, les clusters et les structures ;
— Développer différentes architectures de programmation ;
— Établir une application simple d'acquisition de données .

0.1 Pré-test

Le teste suivant nous permet d'évaluer les connaissances de l'étudiant ainsi que ses utilisations de la programmation graphique :

1. Qu'est-ce que le LabVIEW ?
2. LabVIEW est-il un langage de programmation compilé ?
3. Comment puis-je utiliser des bibliothèques externes dans LabVIEW ?
4. LabVIEW peut-il être intégré aux pratiques d'ingénierie logicielle ?
5. Comment puis-je suivre l'historique des versions VI ?
6. Les VIs LabVIEW peuvent-ils être stockées dans des fournisseurs de contrôle de code source ?
7. Puis-je fusionner des VIs LabVIEW ?
8. Comment puis-je déterminer quelles modifications ont été apportées à un VI ?
9. Comment valider une application LabVIEW ?
10. Puis-je mapper les exigences au code LabVIEW ?
11. Comment puis-je documenter le code graphique ?
12. Comment effectuer une analyse de code statique sur les VIs LabVIEW ?
13. Citez quelques applications du LabVIEW .

0.2 Les Pré-requis

Pour pouvoir suivre ce module, l'étudiant n'a pas besoin d'avoir des pré-requis en terme des connaissances, mais il doit avoir la motivation et l'envie d'apprendre le principe de la

programmation graphique .

0.3 Public Cible

Ce cours est destiné aux étudiants de la 1ère année master Automatique et Systems.

Chapitre 1

Prise en main de LabVIEW

1.1 Objectifs du chapitre

Ce chapitre vise à doter les étudiants des connaissances et compétences requises pour les rendre capable de :

— Présenter LabVIEW et ses fonctionnalité
— Comprendre les composants d'un Instrument Virtuel (appelé VI)

1.2 Initiation à la programmation graphique LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) est un environnement de développement de programme comme les langages C ou BASIC. Cependant, LabVIEW diffère de ces applications sur un point important. En effet, alors que les autres outils de programmation emploient des langages textuels.

LabVIEW utilise un langage de programmation graphique pour créer des programmes sous la forme de diagrammes. LabVIEW est un système de programmation complet comprenant différentes bibliothèques permettant d'effectuer toutes les fonctions des programmes traditionnels.

De plus, LabVIEW comprend plusieurs bibliothèques avancées permettant, par exemple, de faire de l'acquisition de données et du contrôle d'instruments . Les programmes LabVIEW sont appelés instruments virtuels, ou VIs, car leur apparence et leur fonctionnement

s'apparentent à ceux des instruments réels, tels que les oscilloscopes et les multimètres.

LabVIEW contient une grande gamme d'outils pour l'acquisition, l'analyse, l'affichage et l'enregistrement des données, ainsi que des outils pour vous aider à mettre au point votre programme. Vous pouvez utiliser LabVIEW pour communiquer avec le matériel comme, par exemple, d'acquisition de données ou d'images [1].

1.3 Historique

C'est en 1986 que la première version de LabVIEW voit le jour sur Macintosh . Il s'ensuit un travail incessant pour ajouter des fonctionnalités :

1.4 Fonctionnement de LabVIEW

La fenêtre de démarrage s'ouvre lorsque vous lancez LabVIEW. Utilisez cette fenêtre pour créer de nouveaux projets et ouvrir des fichiers existants. Vous pouvez aussi accéder à des ressources permettant d'étendre la capacité de LabVIEW et à des informations vous aidant à vous familiariser avec LabVIEW. La fenêtre de démarrage disparaît lorsque vous ouvrez un fichier existant ou que vous créez un nouveau fichier, et réapparaît lorsque vous fermez toutes les faces-avant et tous les diagrammes ouverts. Vous pouvez aussi afficher la fenêtre à partir de la face-avant ou du diagramme en sélectionnant Affichage» Fenêtre de démarrage [2] .

1.5 Création d'un nouveau VI

Lorsque vous lancez LabView, la fenêtre suivante apparaît :

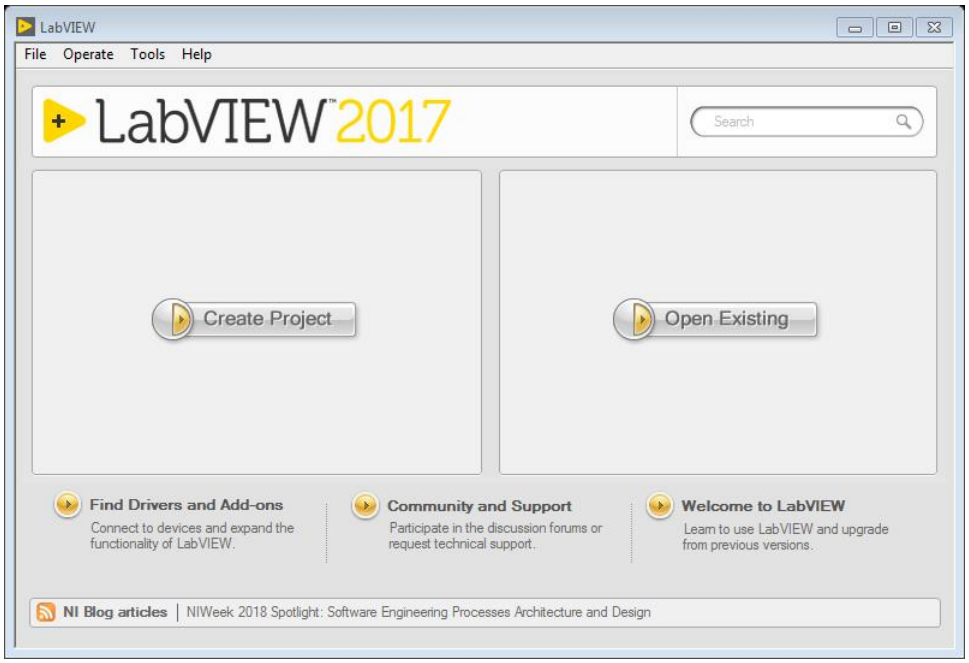


FIGURE 1.1: LabVIEV

Pour créer un nouveau VI, il suffit de cliquer sur **File - New VI** et deux fenêtres apparaîtront :

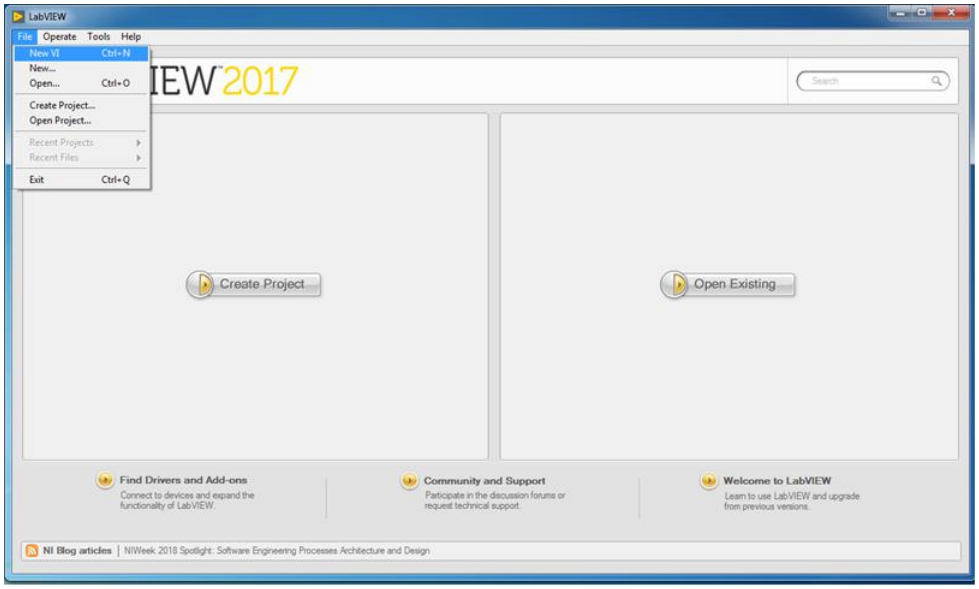


FIGURE 1.2: Créer un nouveau VI

La fenêtre avec le fond gris est la face-avant

La fenêtre avec le fond blanc est le diagramme

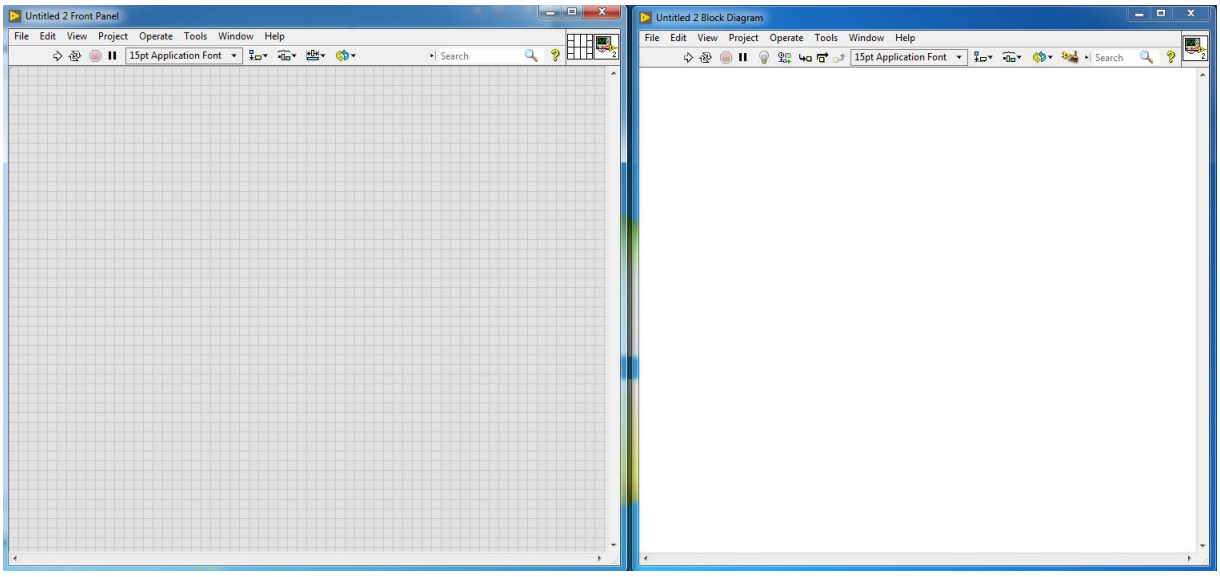


FIGURE 1.3: face-avant et blok diagramme

1.6 Vocabulaire LabVIEW

Les VI se composent de trois éléments principaux :

1.6.1 Face avant

La face-avant est l'interface utilisateur d'un VI. Pour construire la face-avant, vous utilisez des commandes et des indicateurs qui sont respectivement les terminaux d'entrée et les terminaux de sortie interactifs du VI. Les commandes et les indicateurs sont situés sur la palette Commandes. Les commandes sont des boutons rotatifs, des boutons-poussoirs, des cadrans et autres mécanismes d'entrée. Les commandes simulent les mécanismes d'entrée des instruments et fournissent des données au diagramme du VI. Les indicateurs sont des graphes, des LED et autres types d'afficheurs. Les indicateurs simulent les mécanismes de sortie d'instruments et affichent les données que le diagramme acquiert ou génère [3].

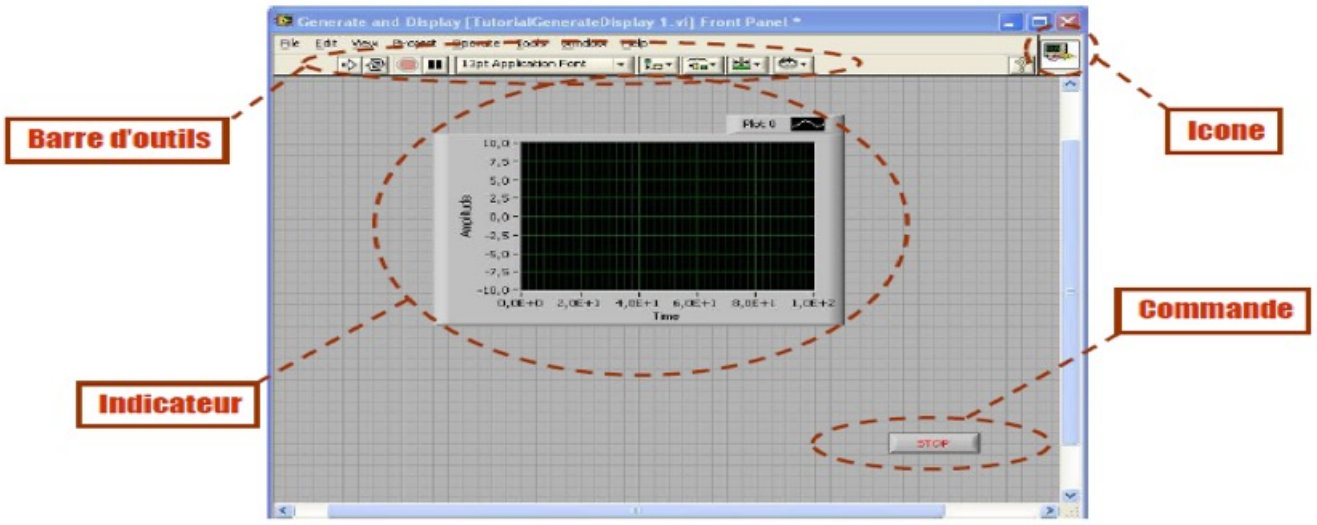


FIGURE 1.4: Interface utilisateur (face-avant)

Dans la face-avant nous trouverons tous les éléments interactifs du VI (commandes et indicateurs).

« Contrôles » = entrées
« Indicateurs » = sorties

1.6.2 Diagramme

Le diagramme contient le code source graphique, appelé aussi code G ou code du diagramme, qui détermine comment le VI s'exécute. Le code du diagramme utilise des représentations graphiques de fonctions pour contrôler les objets de la face-avant. Les objets de la face-avant apparaissent sous forme de terminaux d'icône sur le diagramme. Les fils de liaison connectent les terminaux des commandes et des indicateurs aux VI Express, aux VIs et aux fonctions. Les données circulent dans les fils de liaison des façons suivantes : des commandes aux VIs et aux fonctions, des VIs et des fonctions aux indicateur et des VIs et fonctions à d'autres VIs et fonctions. La direction dans laquelle les données passent par les nœuds du diagramme détermine l'ordre d'exécution des VIs et des fonctions. Ce mouvement de données est appelé programmation par flux de données [3].

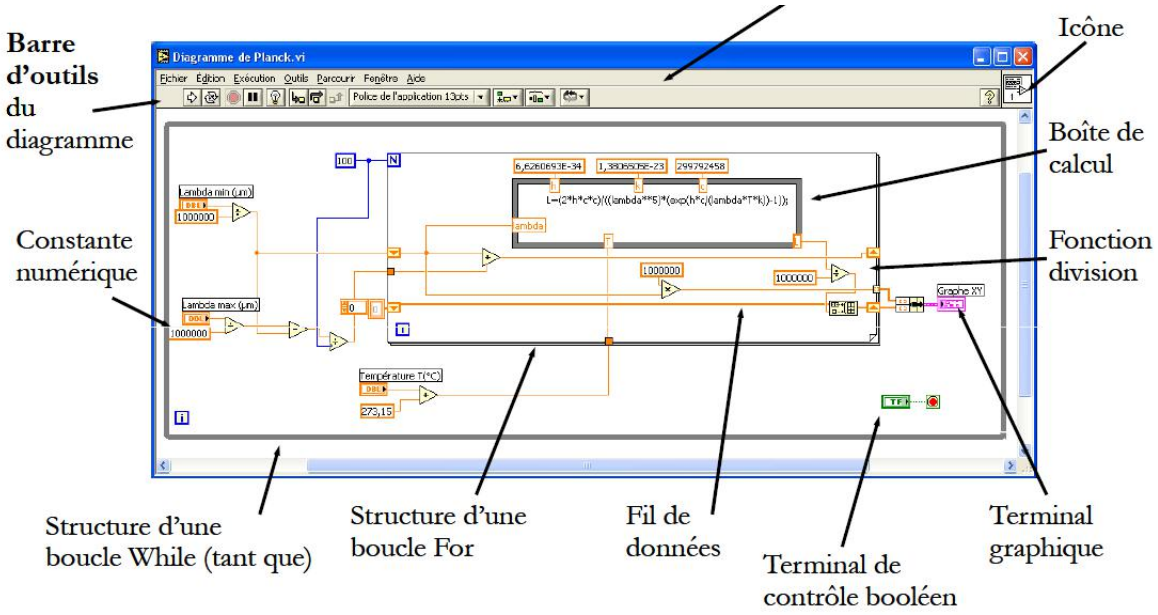


FIGURE 1.5: Fenêtre de programmation et d'affichage du code source

Dans le diagramme nous trouverons tous les éléments propres au code développé (structures, fonctions, constantes,...). Interaction entre face avant et diagramme : Ctrl+E

1.6.3 Icône/Connecteur

Chaque VIs possède une icône par défaut affiché en haut à droite de la face-avant. Cette icône est en fait celui de LabVIEW. Pour changer l'icône, il suffit de cliquer sur le bouton droit de votre souri de faire un clic droit sur celui-ci et de sélectionner l'option Edit Icon. Vous serez alors en mesure de changer le dessin de l'icône à l'aide d'un petit éditeur d'image ressemblant à Paint. De plus, pour communiquer avec les autres VIs, il faut que les contrôleurs et les indicateurs de la face-avant aient des connecteurs. Pour accéder aux connecteurs, il faut faire un clic droit sur l'icône et choisir l'option Show connectors. Vous pourrez alors ajouter des terminaux et relier ceux-ci (à l'aide de l'outil fil) avec des éléments de la face-avant. De cette manière, vous pourrez utiliser votre VI comme sous-VI dans un autre programme [4].

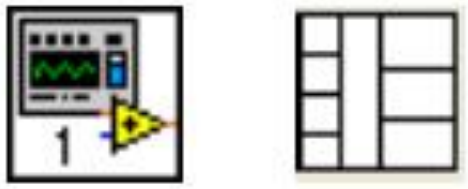


FIGURE 1.6: Icône/Connecteur

1.6.4 Environnement LabVIEW

LabVIEW est un outils d'acquisition, d'analyse et de présentation de données.

FIGURE 1.7: Environnement LabVIEW

1.6.5 Acquisition avec LabVIEW

LabVIEW permet l'acquisition de données par l'intermédiaire de diverses connectiques :

- PCI (Peripheral Component Interconnect)
- CompactFlash
- LAN (Local Area Network)
- USB (Universal Serial Bus)
- GPIB (IEEE 488) (General Purpose Interface Bus)
- PXI (PCI eXtensions for Instrumentation)
- PCMCIA (Personal Computer Memory Card International Association)
- Wi-Fi (IEEE 802.11 b/g/n) (Wireless Fidelity)
- Bluetooth
- IrDA (Infrared Data Association)
- Firewire (IEEE 1394)

— Ethernet
— Série (RS 232, RS 449, RS 422, RS 423, RS 485)
— VXI (VME eXtensions for Instrumentation.

1.6.6 Analyse avec LabVIEW

LabVIEW inclut des outils pour l'analyse des données [4] :

— Traitement du signal : Convolution, analyse spectrale, transformées de Fourier,...
— Traitement d'images : Masque, détection de contours, profils, manipulations de pixels,...
— Mathématiques : Interpolation, statistiques (moyennes, écart-type,...), équations différentielles,..

1.6.7 Présentation avec LabVIEW

LabVIEW inclut des outils d'aide à la présentation (communication) des données :

— Graphiques, tableaux, images, génération de rapport,...
— Par l'intermédiaire d'internet : outils de publication web, serveur data-socket.

Chapitre 2

Boucles et Structures

Objectifs du chapitre

Ce chapitre vise à doter les étudiants des connaissances et compétences requises pour les rendre capable de : Exploiter correctement la programmation graphique : Travailler avec les tableaux, les clusters et les structures Développer différentes architectures de programmation

2.1 Introduction

Les boucles et les registres à décalage sont des structures de données très utilisées en programmation. Celles-ci contrôlent le flux des données dans le VI et elles fonctionnent de la même manière que dans un langage de programmation textuel.

Les structures Condition et Séquence peuvent toutes deux posséder plusieurs sous-diagrammes, mais un seul diagramme est visible à la fois. Au-dessus de chaque bordure de structure, se trouve la fenêtre d'affichage du sous-diagramme, qui contient un identificateur de diagramme au centre, ainsi que des boutons de décrémentation et d'incrémentation de chaque côté. L'identificateur de diagramme indique le type de sous-diagramme couramment affiché .

2.2 Les Boucles

Dans la palette structures, nous trouvons les boucles (For, While), la structure Condition, le noeud MathScript dans lequel nous pouvons programmer dans un langage proche à MATLAB, la boite de calcul avec un langage proche du C, etc.

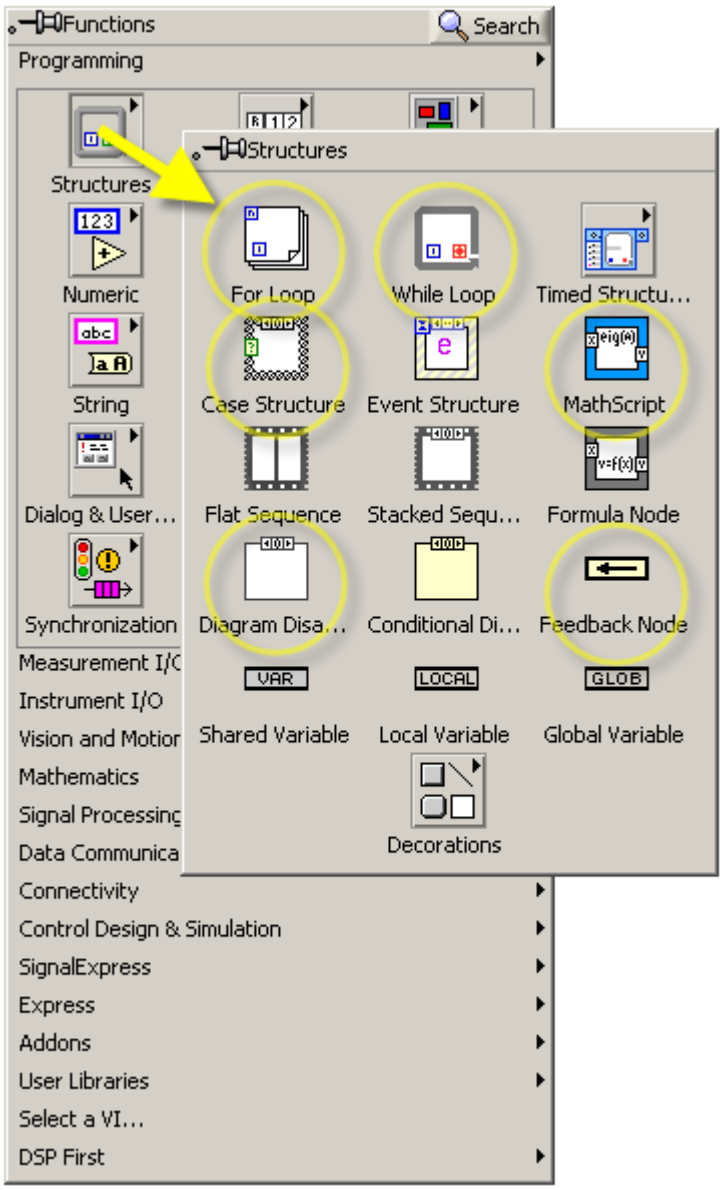


FIGURE 2.1: Palette Structures

2.2.1 Boucle For

Le langage G de LabVIEW possède 2 structures permettant de répéter l'exécution d'un ensemble de VIs ; les boucles **For** et **While**. La boucle For s'exécute un nombre de fois défini par le nombre N. L'indice i d'itération va toujours de 0 à (N-1). Comme en langage C, c'est une structure itérative permettant d'exécuter une partie de programme un nombre déterminé de fois, connu à l'avance.

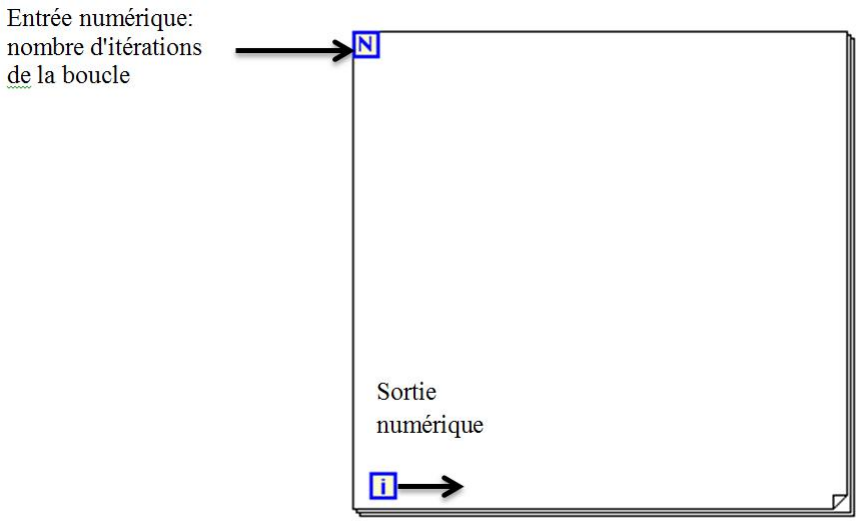


FIGURE 2.2: Boucle For

2.2.2 Boucle While

Une boucle While est une section de code qui est répétée jusqu'à ce qu'une condition soit remplie. En effet, la série d'instruction à l'intérieur du bloc sera exécutée jusqu'à ce que la condition d'arrêt soit remplie. Vous pouvez utiliser le terminal d'itération pour compter le nombre de fois que la boucle a été exécutée. De plus, vous pouvez, en faisant un clic droit sur le terminal conditionnel, choisir la condition d'arrêt entre Stop if true ou Continue if true.

FIGURE 2.3: Boucle While

2.2.3 Registres à décalage

Les registres à décalages permettent de transférer une valeur d'une itération à une autre. Ils ne sont disponibles que pour les boucles While et For. Pour ajouter un registre à décalage, il suffit de cliquer sur le bouton droit de votre souri faire un clic droit sur la bordure de la boucle et de sélectionner Add Shift Register (nœud de rétroaction). Lorsque vous ajouter un registre, deux terminaux apparaissent. Celui de gauche enregistre les informations à la fin de l'itération tandis que celui de droite transmet cette information à l'itération suivante. Il est également possible de conserver les valeurs de plusieurs itérations. Pour ce faire, il suffit d'ajouter des terminaux du côté gauche en cliquant sur le bouton droit de votre souri faisant un clic droit sur le terminal existant et en sélectionnant Add Element [4].

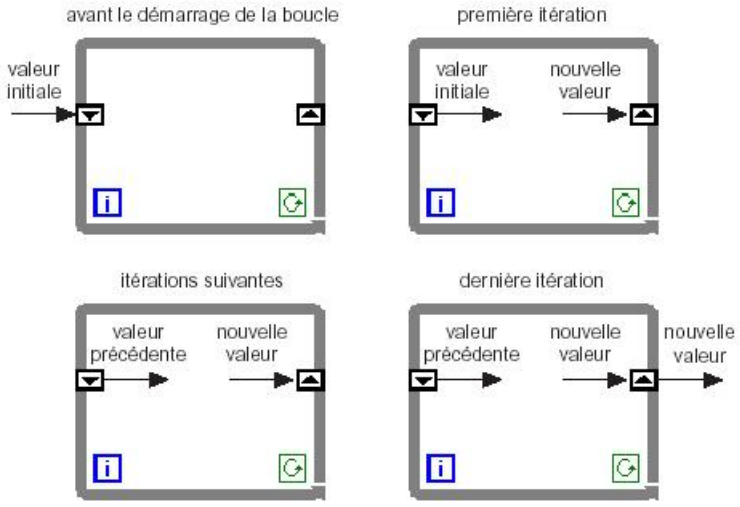


FIGURE 2.4: Registres à décalages

2.2.4 Boîtes de calculs

Grâce aux boîtes de calcul, vous pouvez entrer directement une ou plusieurs formules compliquées, au lieu de créer des sous-sections du diagramme. Entrez des formules en utilisant l'outil Texte. Il faut toujours terminer ses instructions pas un point-virgule " ;". Pour ajouter des entrés et des sorties à cette boîte, et vous cliquez sur le bouton droit de votre souri faites un clic droit sur la bordure et sélectionner Add imput ou Add output.

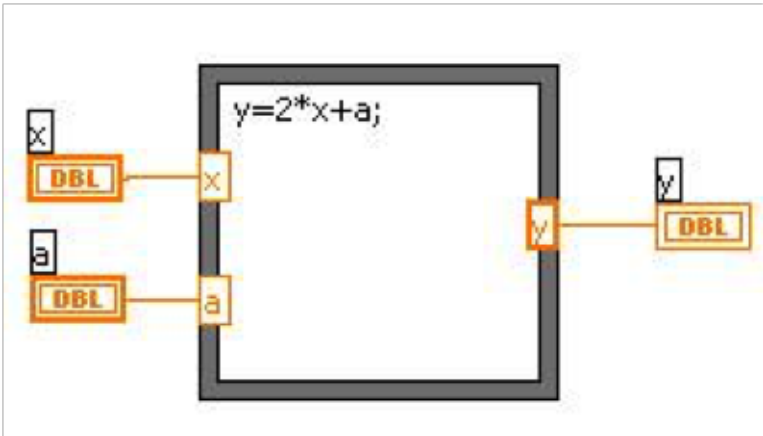


FIGURE 2.5: Boîtes de calculs

2.3 Les Structures

La structure condition contient au moins deux sous-diagrammes. La structure n'exécutera qu'un seul des sous-diagrammes selon le critère de sélection. Cette structure est très semblable à un "case" dans les langues de programmation textuelles. Le critère de sélection peut être soit un booléen, un entier ou une chaîne de caractères [4].

FIGURE 2.6: Structure

2.3.1 Structure séquence

La structure séquence permet d'exécuter des instructions dans un certain ordre. En effet, cette structure permettra d'exécuter une série d'instruction dans l'ordre déterminé par l'identificateur de diagramme. Donc, toutes les instructions dans le cadre 0 seront exécutées en premier, en second lieu les instructions de cadre 1 et ainsi de suite.

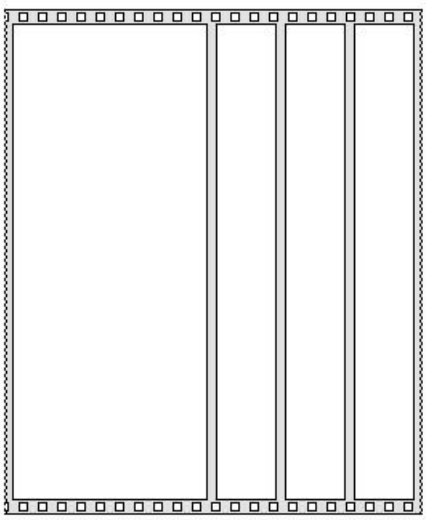


FIGURE 2.7: Structure séquence

2.3.2 Structures Événement

Une structure Événement, représentée ci-après, possède un ou plusieurs sous-diagramme ou conditions d'événement, dont un seul s'exécute lorsque la structure s'exécute.

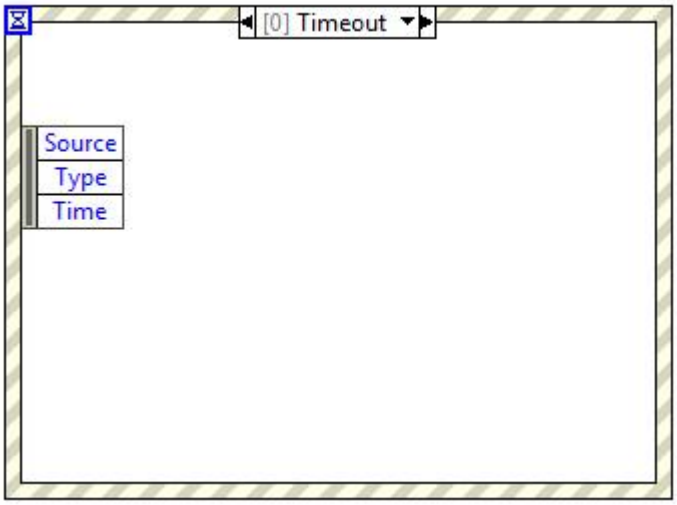


FIGURE 2.8: Structures Événement

La structure Événement attend qu'un événement se produise, puis exécute la condition appropriée pour gérer cet événement. Les événements peuvent être générés dans l'interface utilisateur, par des E/S (Entrée/Sortie) externes ou par d'autres parties de l'application. Les événements d'interface utilisateur comprennent les clics de souris, la saisie par les touches du clavier et ainsi de suite.

Les événements d'E/S externes comprennent les compteurs ou les déclenchements matériels qui avertissent qu'une acquisition de données est achevée ou qu'une erreur s'est produite.

D'autres types d'événement peuvent être générés par programmation et utilisés pour communiquer avec différentes parties de l'application. LabVIEW prend en charge les événements de l'interface utilisateur et ceux qui sont générés par programmation, mais pas les événements des E/S externes [4].

Chapitre 3

Les Graphiques

Objectifs du chapitre

Ce chapitre vise à doter les étudiants les avantages majeurs de LabVIEW qui fournir des fonctions graphiques de haut niveau immédiatement utilisables par l étudiant.

3.1 Introduction

Après avoir acquis ou généré des données, utilisez un graphe ou un graphe déroulant pour les afficher sous forme graphique. Les graphes et les graphes déroulants ont une façon différente d'afficher et de mettre à jour les données. Les VIs utilisant un graphe commencent souvent par rassembler les données dans un tableau, puis les affichent sur le graphe. Ce processus est similaire à une feuille de calcul dans laquelle vous enregistrez les données avant de les tracer.

Lors du traçage des données, le graphe supprime les données précédentes et n'affiche que les nouvelles données. Normalement, vous utilisez un graphe avec les processus rapides qui acquièrent des données en continu. À l'inverse, un graphe déroulant ajoute de nouveaux points de données à ceux qui sont déjà affichés pour créer un historique. Sur un graphe déroulant, vous pouvez afficher la mesure ou la lecture actuelle dans son contexte, avec les données déjà acquises.

Lorsque le nombre de points dépasse le nombre affichable sur le graphe déroulant, le graphe déroulant se met à défiler, les nouveaux points apparaissant à droite et les anciens

points disparaissant à gauche. Le graphe déroulant est typiquement utilisé dans le cas de processus lents avec peu de nouveaux points par seconde à tracer [5].

3.2 Types de graphes et de graphes déroulants

LabVIEW comprend les types de graphes et de graphes déroulants suivants [5] :

1. **Graphes et graphes déroulant**s :Affichent les données acquises à une vitesse constante
2. **Graphes XY :** Affichent les données acquises à une fréquence variable et les données pour les fonctions à valeurs multiples.
3. **Graphes et graphes déroulants d'intensité :** Affichent des données 3D sur un tracé 2D en utilisant des couleurs pour afficher les valeurs de la troisième dimension.
4. **Graphes numériques :** Affichent les données en tant qu'impulsions ou groupes de lignes numériques. (Windows) Graphes 3D
5. **Graphes 3D :** Affichent des données 3D sur un tracé 3D dans un objet ActiveX de la face-avant Reportez-vous au répertoire labview\examples\general\graphs pour consulter des exemples de graphes et de graphes déroulants.

3.2.1 Graphes et graphes déroulants

LabVIEW comporte des graphes et des graphes déroulants pour afficher des données acquises normalement à une fréquence constante

3.2.2 Graphes

Le graphe affiche un ou plusieurs tracés de mesures échantillonnés de manière constante. Le graphe ne trace que les fonctions à une seule valeur, par exemple dans y = f(x) avec des points distribués également le long de l'axe des X, comme les waveforms acquises qui varient dans le temps. La figure suivante montre un exemple de graphe.

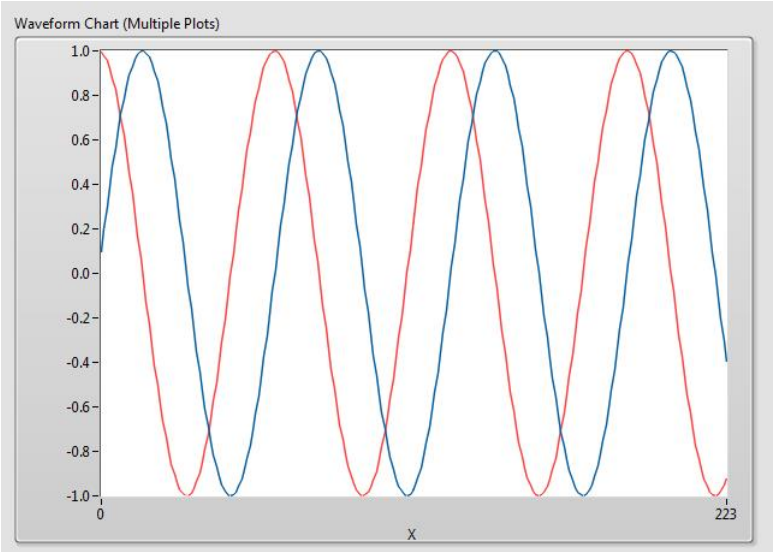


FIGURE 3.1: LabVIEW Graphes

3.2.3 Graphes déroulants

Le graphe déroulant est un type particulier d'indicateur numérique qui affiche un ou plusieurs tracés déchantillons acquis à une vitesse constante. La figure suivante montre un exemple de graphe déroulant

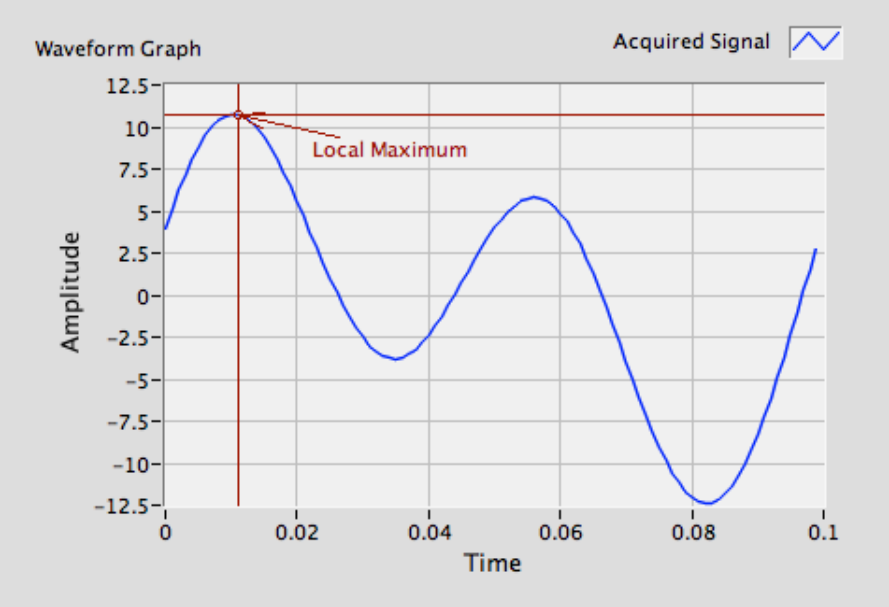


FIGURE 3.2: graphe déroulant

Le graphe déroulant conserve un historique des données, ou buffer, des mises à jour précédentes. Cliquez avec le bouton droit sur un graphe déroulant et sélectionnez Longueur de l'historique dans le menu local pour configurer le buffer. Par défaut, la longueur de l'historique des graphes déroulants est de 1024 points de données. La fréquence à laquelle vous envoyez des données au graphe déroulant détermine la fréquence de rafraîchissement du tracé dans le graphe déroulant [5].

3.2.4 Graphes XY

Le graphe XY est un objet graphique cartésien à usage général qui trace des fonctions à valeurs multiples, comme des formes circulaires ou des waveforms avec une base de temps qui varie. Le graphe XY affiche des ensembles de points, échantillonnés régulièrement ou irrégulièrement. Vous pouvez afficher des plans de Nyquist, Black-Nichols, S et Z sur un graphe XY. Les lignes et les étiquettes sur ces plans sont de la même couleur que les lignes cartésiennes et vous ne pouvez pas modifier la police de l'étiquette du plan. La figure suivante montre un exemple de graphe XY.

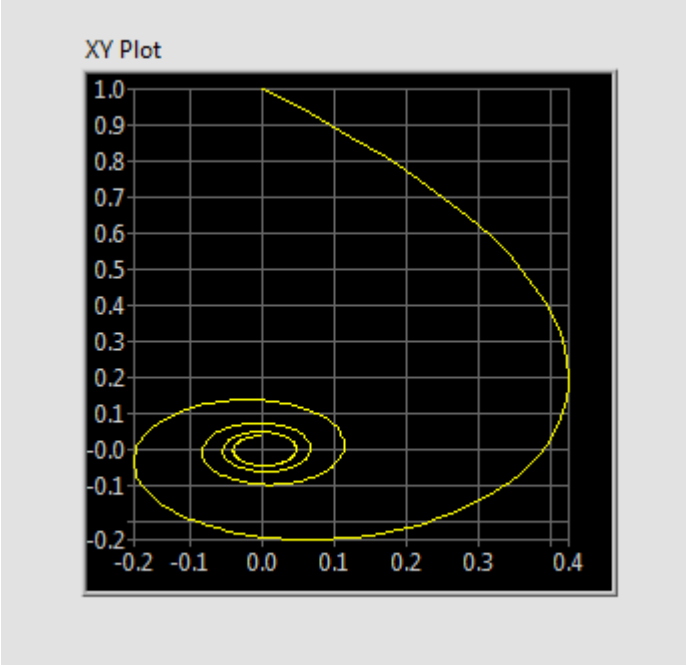


FIGURE 3.3: Graphe XY

Le graphe XY peut afficher des tracés contenant n'importe quel nombre de points. Le graphe XY accepte aussi plusieurs types de données, ce qui minimise la manipulation des

données avant l'affichage

3.2.5 Graphes et graphes déroulants d'intensité

Utilisez le graphe et le graphe déroulant d'intensité pour afficher des données 3D sur un tracé 2D en plaçant des blocs de couleur sur un plan cartésien. Par exemple, vous pouvez utiliser un graphe ou un graphe déroulant d'intensité pour afficher des données contenant des motifs, comme par exemple une carte de température ou une carte topographique, dans laquelle l'intensité représente respectivement la température et l'altitude. Le graphe et le graphe déroulant d'intensité acceptent un tableau Après avoir positionné un bloc de données sur un graphe déroulant d'intensité, l'origine du plan cartésien se décale vers la droite du dernier bloc de données. Lorsque le graphe déroulant traite de nouvelles données, celles-ci apparaissent à droite des anciennes valeurs de données. Lorsque l'affichage d'un graphe déroulant est plein, les données les plus anciennes débordent à gauche du graphe déroulant. Ce comportement est similaire au comportement des graphes déroulants. La figure suivante montre un exemple de graphe déroulant d'intensité .

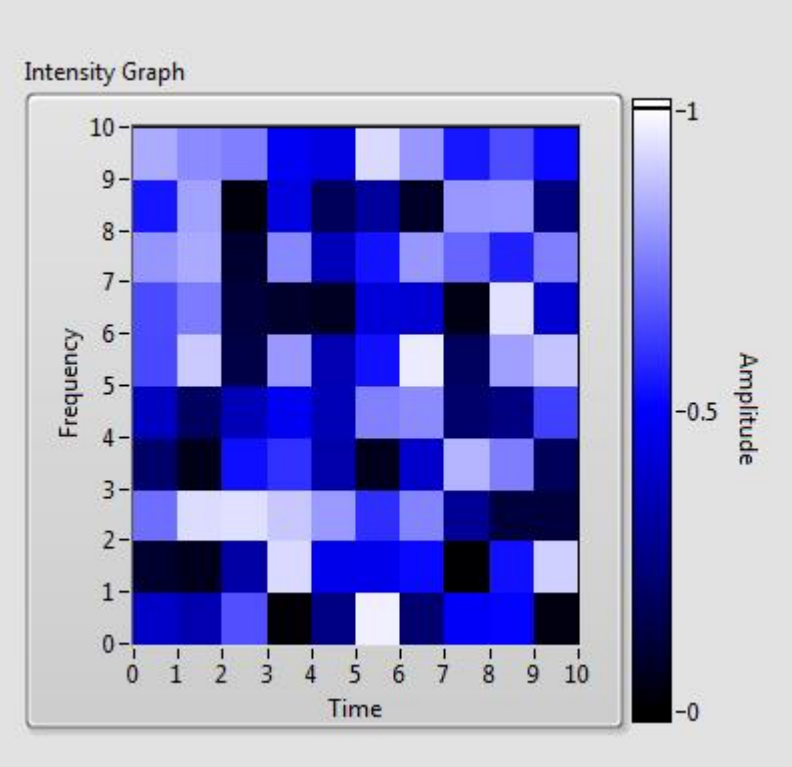


FIGURE 3.4: graphe déroulants d'intensité

3.2.6 Graphes numériques

Utilisez un graphe numérique pour afficher des données numériques, notamment lorsque vous utilisez des diagrammes temporels ou des analyseurs logiques. Le graphe numérique accepte le type de données waveform numérique, le type de données numérique et un tableau de ces types de données comme entrée. Par défaut, le graphe numérique condense les bus numériques de sorte qu'il puisse tracer les données numériques sur un seul tracé. Si vous câblez un tableau de données numériques, le graphe numérique trace chaque élément du tableau comme tracé distinct dans l'ordre du tableau.

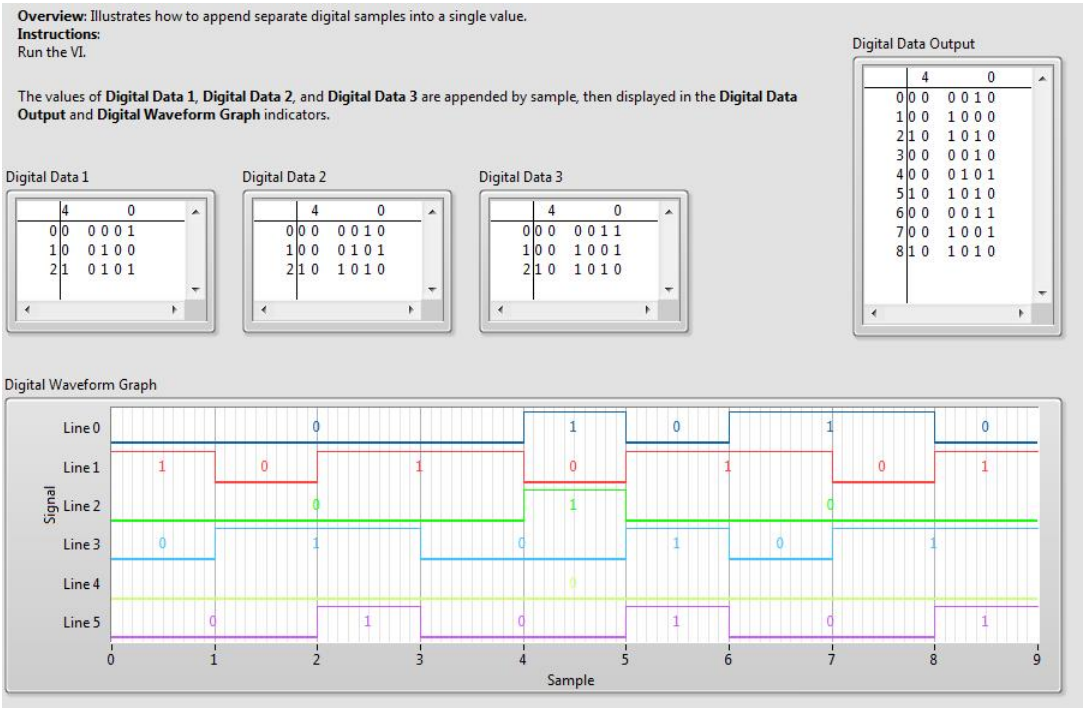


FIGURE 3.5: Graphe numérique

3.2.7 Graphes 3D

Pour de nombreux ensembles de données réelles, comme la répartition de la température sur une surface, l'analyse temps-fréquence et la trajectoire d'un avion, il vous faut visualiser des données en trois dimensions. Vous pouvez afficher des données en trois dimensions à l'aide d'un graphe 3D et changer l'affichage de ces données en modifiant les propriétés du graphe 3D.

Remarque

Les commandes de graphes 3D ne sont disponibles que sous Windows dans les systèmes de développement complet et professionnel de LabVIEW. LabVIEW inclut les types de graphes 3D suivants :

1. Graphe de surface en 3D — Trace une surface dans l'espace 3D.
2. Graphe de surface paramétrique en 3D — Trace une surface paramétrique dans l'espace 3D.
3. Graphe de courbe en 3D — Trace une ligne dans l'espace 3D.

Chapitre 4

Travaux Pratiques

TP N° 1 : Initiation à LabVIEW : Exemples d'applications

But de TP

Dans ce TP, vous apprendrez à manipuler avec le logiciel **LabVIEW** et voir quelques exemples d'application.

Partie théorique

Le programme **LabVIEW** appelé Instruments Virtuels ou Virtual Instruments (VI).

On parle d'instruments virtuels car leur apparence et leur fonctionnement sont semblables à ceux d'instruments réels, tels que les oscilloscopes et les multimètres.

- **Composantes d'un VI**

Les VI se composent des éléments principaux :

1. **Face avant** (Interface utilisateur)

Contrôles = entrées

Indicateurs = sorties

-Dans la face-avant nous trouverons tous les éléments interactifs du **VI** (commandes et indicateurs).

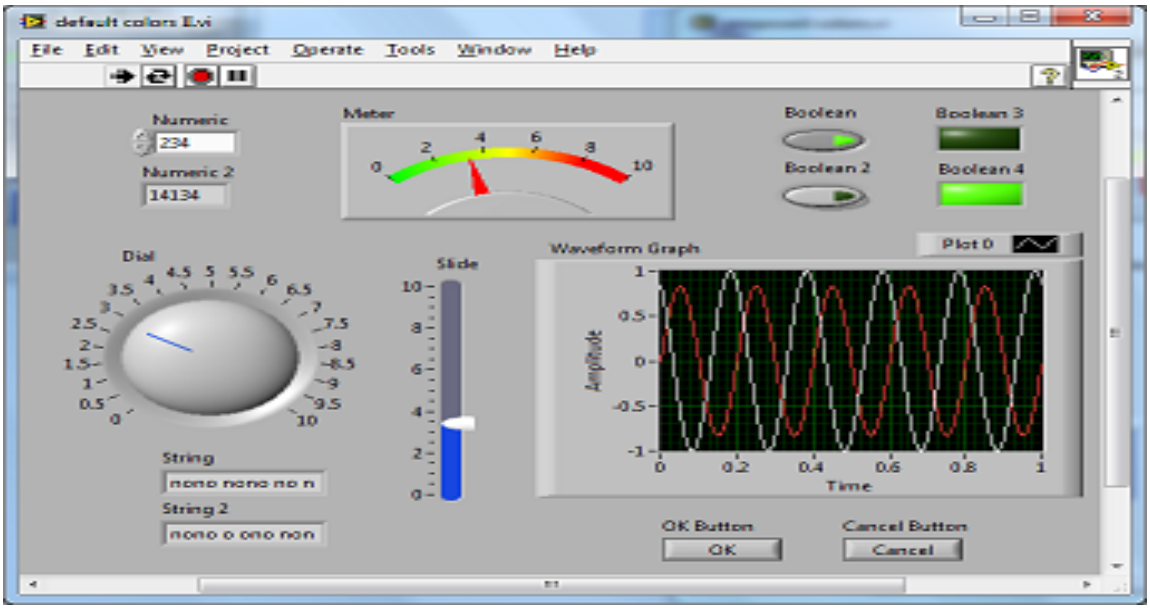


Fig.1 Face avant

Dans le diagramme nous trouverons tous les éléments propres au code développé (structures, fonctions, constantes,…etc).

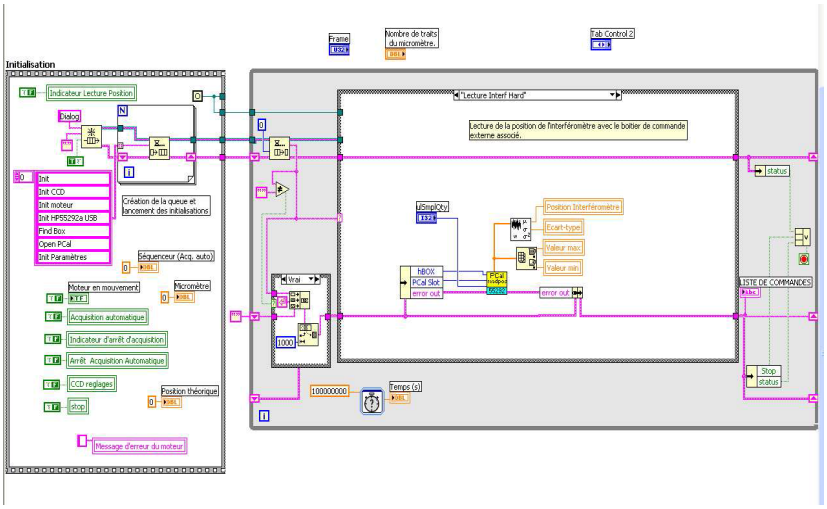


Fig.2 Diagramme (Fenêtre de programmation et d'affichage du code source)

-Interaction entre face avant et diagramme (Ctrl+E)

2. Icône/Connecteur

- Chaque VI affiche une icône, dans le coin supérieur droit des fenêtres de la face-avant et du diagramme. Une icône est la représentation graphique d'un VI.
- Un connecteur est un ensemble de terminaux correspondant aux commandes et aux indicateurs du VI qui sont accessibles.

Partie pratique

1. Type de variables : Créer un programme :

Exemple 1 :

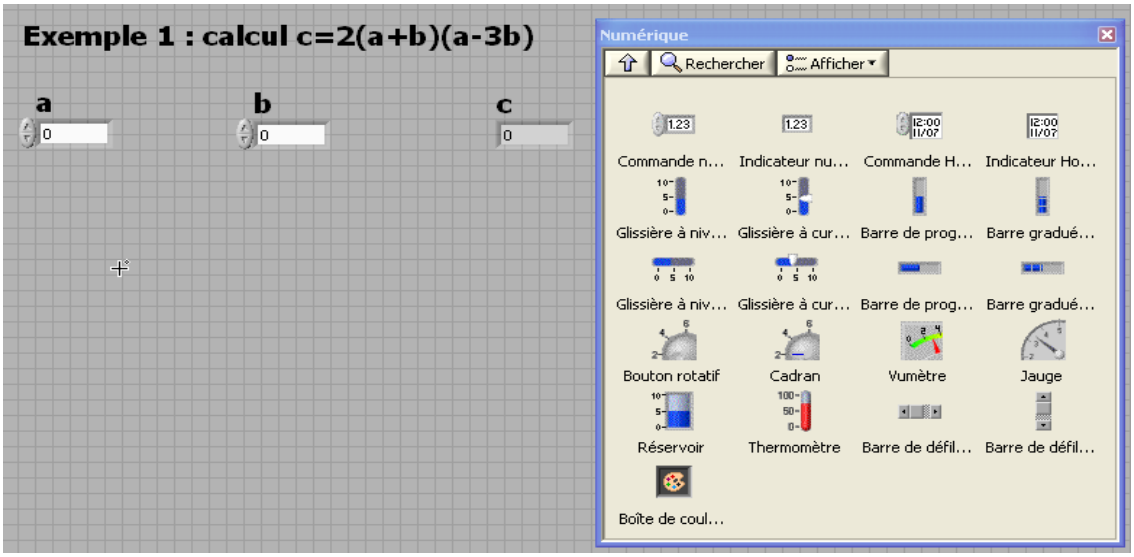
Utilisant le LabVIEW, Calculer de ***c= 2(a+b)(a-3b)*** ou a, b et c sont des nombres réels.

Sur la page de démarrage, choisir **Nouveau → Projet vide**

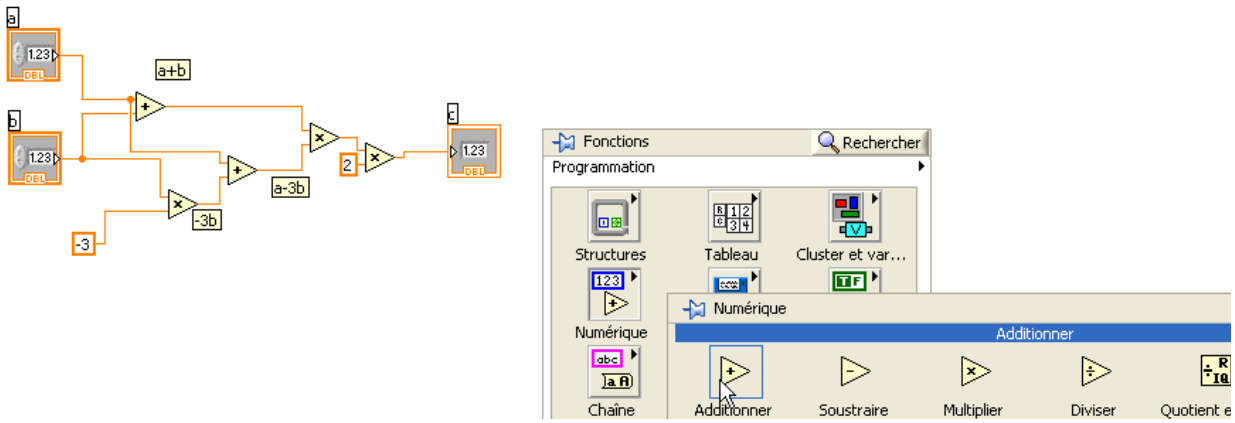
Faite Fichier → Enregistrer (ou **CTRL+s**) et donner le nom du projet : « **Exemple1** »

Un VI s'ouvre, faite Fichier → Enregistrer (ou **CTRL+s**) et donner le nom du programme : « Exemple 1 »

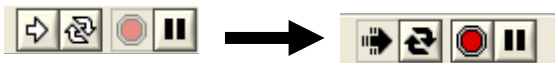
- **Sur la face avant**, ajouter **deux commandes** et un **indicateur** numérique en faisant un clic droit dans la catégorie **Moderne → Numérique** (ils sont implicitement du type double)



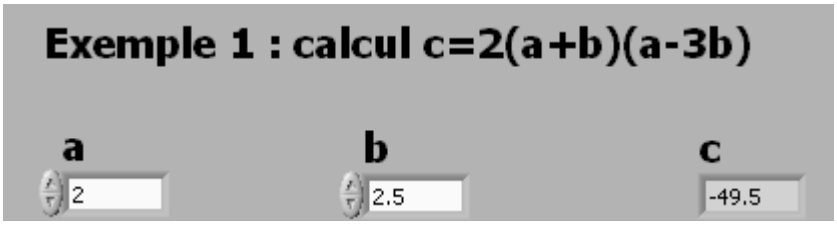
- **Sur le diagramme,** ajouter les opérateurs « + » et « X » dans Programmation → Numériques



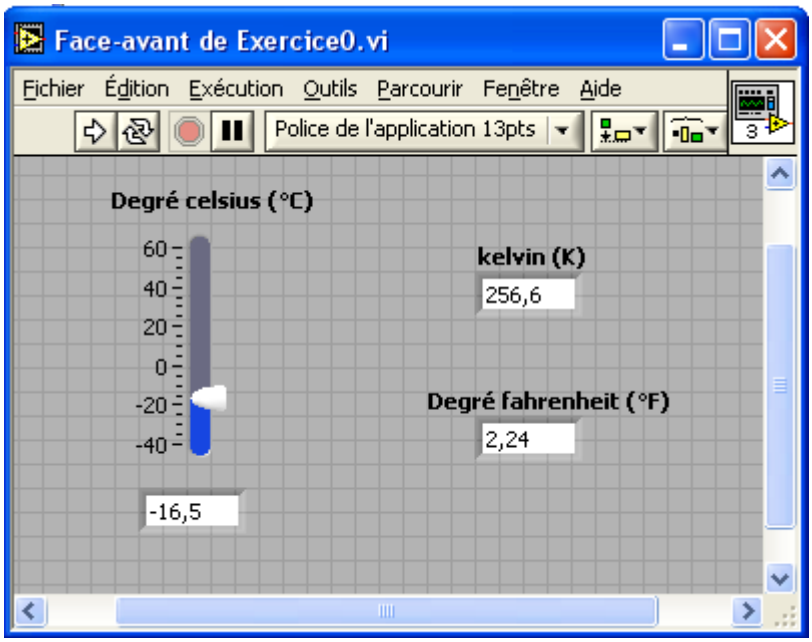
- Puis relier les différents éléments à l'aide de la bobine.
- Enregistrer le projet. Lorsque vous sauvegardez le projet tous les VIs sont sauvés.
- Tester votre programme en appuyant sur le mode exécution continue



- La face avant devient grisée (le quadrillage a disparu)



Application : Conversion de °C en °F et en K



Réaliser un VI qui permet d'effectuer une conversion de °C en K et en °F à partir de fonctions de base de labview.

- **Conversion de °C en K** : $K = °C + 273{,}15$
- **Conversion de °C en °F** : $°F = ((9 \times °C) / 5) + 32$

TP N° 2 : Initiation à LabVIEW : LES BOUCLES Exemples d'applications

But de TP

Dans ce TP, vous apprendrez à manipuler avec le logiciel LabVIEW et voir la programmation des boucles **if** (*condition*) et **for** (*factoriel*).

Partie Théorique

1) Les structures :

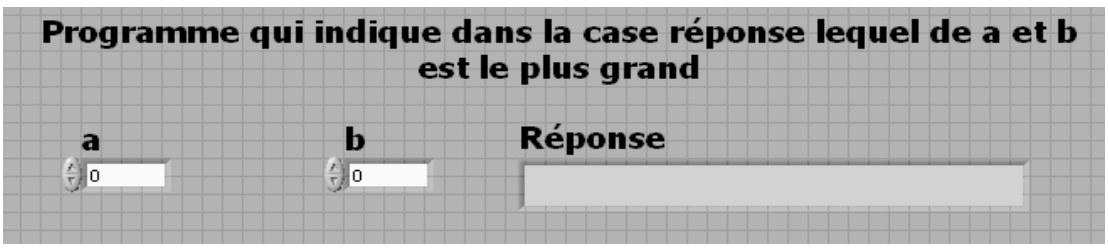
a) Structure condition :

On souhaite réaliser un programme qui teste les grandeurs de a et b

SI a>b
ALORS il affiche « *a est plus grand que b* »
SINON il teste :
SI a=b
ALORS il affiche « *a et b sont égaux* »
SINON il affiche « *b est plus grand que a* »
FIN SI
FIN SI

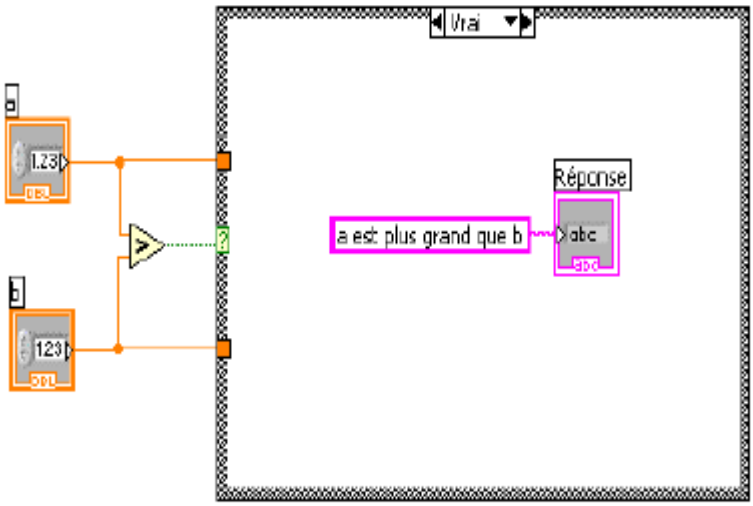
Partie pratique

- **Sur la face avant**, ajouter deux commandes « a » et « b » numériques (Moderne → Numérique) et un indicateur de chaîne « Réponse » (Moderne → Chaine et chemin)



❖ **Sur le diagramme**, ajouter la structure de condition si « a > b » dans Programmation → Structures

- Ajouter le symbole de comparaison « > » dans Programmation → Comparaison
- Dans la condition VRAI, glisser l'indicateur Réponse et créer une constante « a est plus grand que b » (bouton droit sur l'indicateur puis Créer → Constante)
- Relier les différents éléments et câbler la condition.



Dans la condition FAUX rajouter une nouvelle condition si a = b dans Programmation → Structures

Ajouter le symbole de comparaison « = » dans Programmation → Comparaison

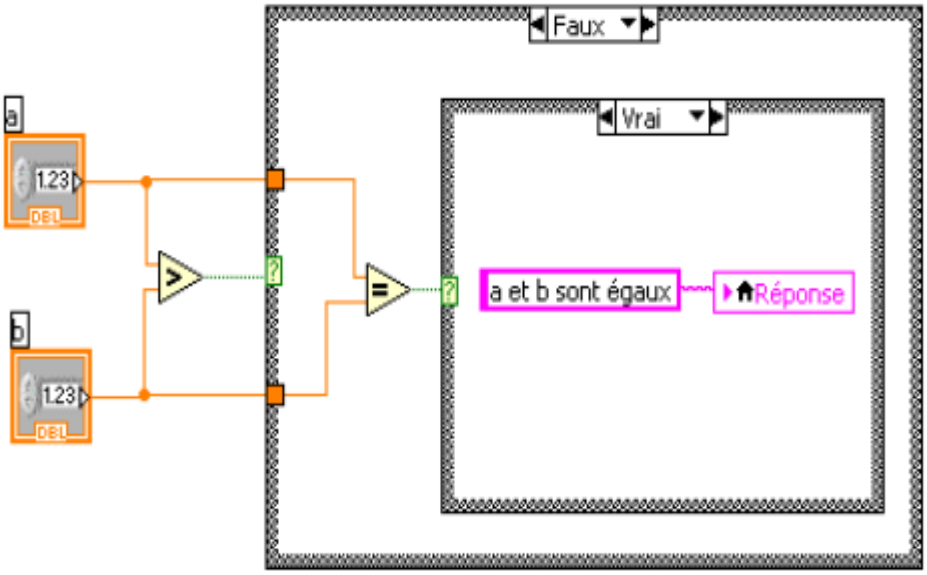
Créer deux variables locales sur l'indicateur réponse (bouton droit sur l'indicateur puis Créer → Variable locale) : 

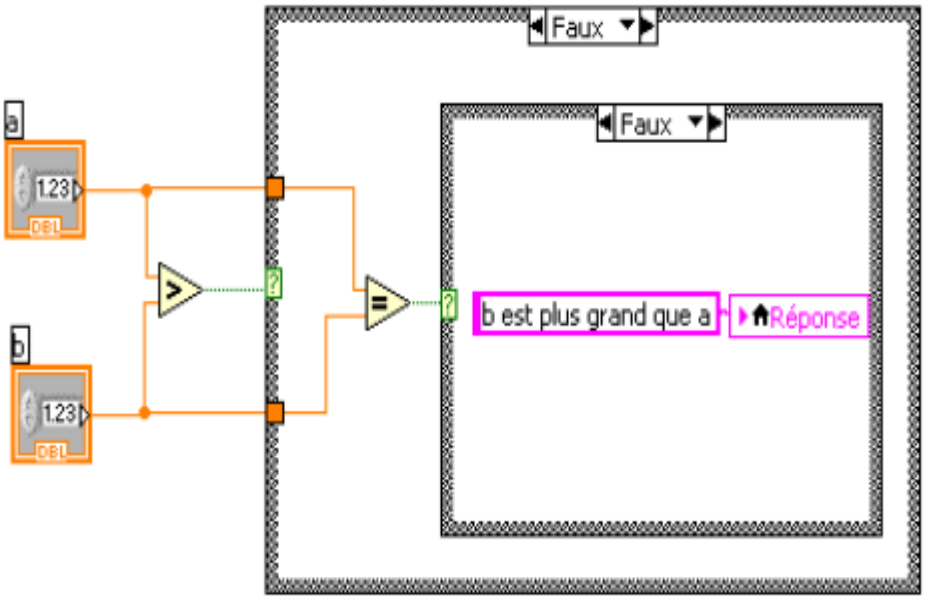
.

Dans la condition VRAI, créer la constante de chaîne « a et b sont égaux »

Dans la condition FAUX, créer la constance de chaîne « b est plus grand que a »

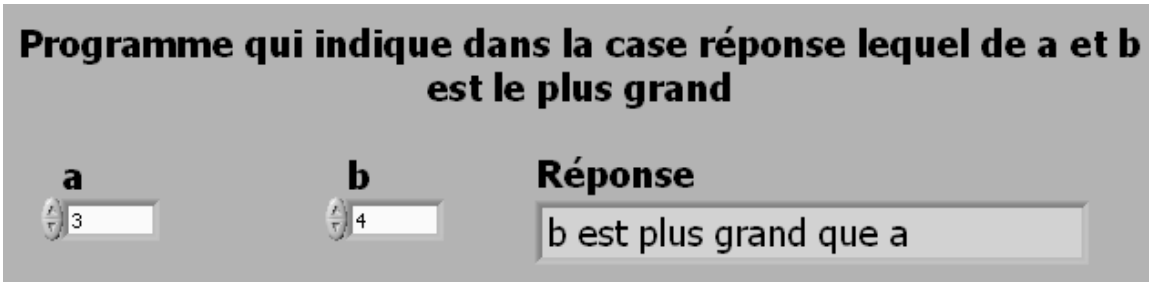
Relier les différents éléments





Enregistrer le projet. Lorsque vous sauvegardez le projet tous les VIs sont sauvées.

Tester votre programme en appuyant sur le mode « Exécution continue »

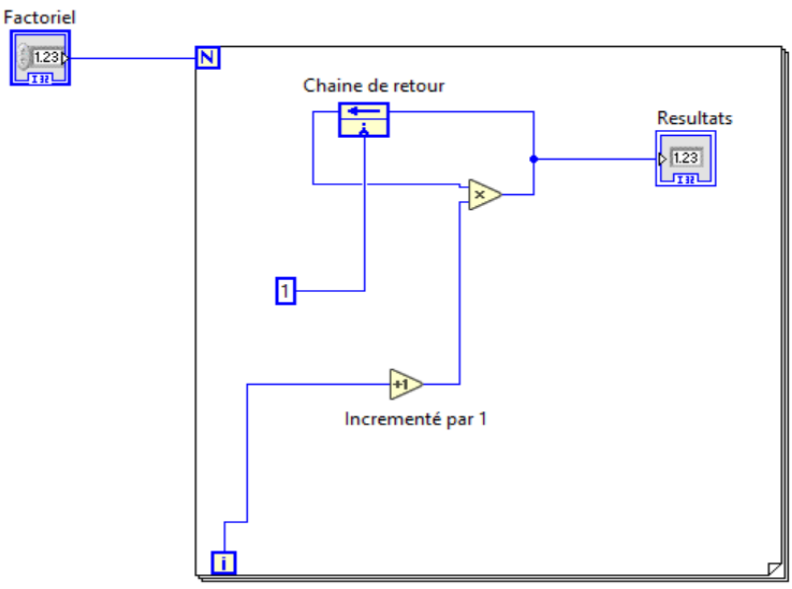


b) Structure de Boucle For :

On souhaite calculer le factoriel d'un nombre choisit par l'utilisateur.

❖ **Sur la face avant**, ajouter une commande numérique « Factoriel » et un indicateur numérique « Résultats » (Tous les numériques seront des entiers **I32**).

Sur le diagramme, créer la boucle **FOR** dans Programmation → Structures

- Ajouter un incrément dans Programmation → Numériques → Incrément
- Créer un operateur multiplication dans Programmation → Numériques → Multiplié
- Créer la chaine de retour autour de nombre multiplié
- Dans la chaine de retour, cliquer à droite et ajouter un constant = 1.
- Relier tous les éléments.
- 

Enregistrer le projet. Lorsque vous sauvegardez le projet tous les VIs sont sauvées.

Tester votre programme en appuyant sur le mode « Exécution continue »

TP N° 3 : Initiation à LabVIEW : *structure évènement* exemples d'applications

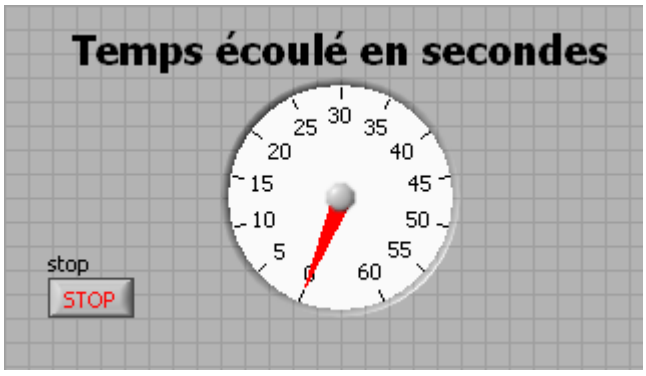
But de TP

Dans ce TP, vous apprendrez à manipuler avec le logiciel LabVIEW et voir la programmation la boucle ***while*** et La ***structure évènement***.

1) Boucle While :

Tant que l'utilisateur n'appuie pas sur stop, on affiche le temps écoulé en seconde dans une jauge.

- **Sur la face avant**, ajouter une jauge « Temps écoulé en secondes » (Moderne → Numériques) et un bouton « Stop » (Moderne → Booléen).
- Changer la valeur maximum de la jauge, soit en double cliquant directement sur le 10 et en tapant 60 ou en faisant un clic droit puis Propriétés puis onglet « Echelle».



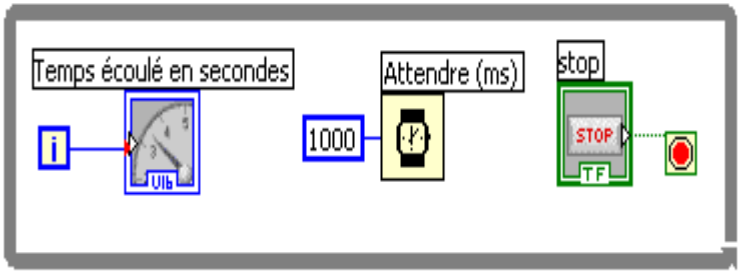
- Sur le diagramme, créer la boucle ***While*** dans Programmation → Structures

  Ajouter la fonction « **Attendre (ms)** » dans Programmation → Informations temporelles

- Créer une constante = à 1000 dans Programmation → Numériques et relier là à la fonction précédente.

- Relier la jauge à l'indice '**i**' de la boucle ('**i**' sera incrémenté de 1 si tous ce qui est dans la boucle a été effectué donc dès que 1000 ms se sont écoulées soit 1 seconde, '**i**' augmente de 1)

- Relier le bouton « Stop » à la condition d'arrêt de la boucle.



- Enregistrer le projet. Lorsque vous sauvegardez le projet tous les VIs sont sauvés.

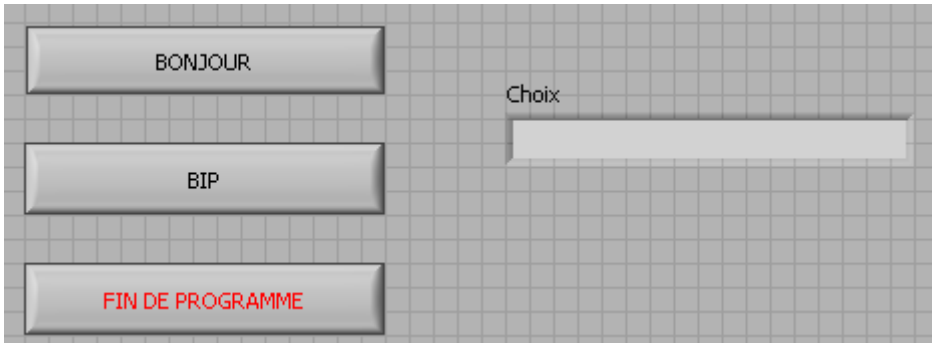
- Tester votre programme en appuyant sur le mode « Exécution Unique »

2) La structure évènement :

Exemple : On souhaite créer un programme qui :

1. Ouvre une fenêtre avec écrit « **bonjour** » si on clique sur le bouton **BONJOUR** et affiche dans la chaîne de caractères **BON CHOIX**.
2. Fait un **bip** si on clique sur le bouton **BIP** et affiche dans la chaîne de caractères **BON CHOIX**.
3. Affiche dans la chaîne de caractères : **MAUVAIS CHOIX** si on sort avec la souris de la face avant.

❖ **Sur la face avant**, ajouter trois boutons « **BONJOUR** », « **BIP** » et « **FIN DE PROGRAMME** » (Moderne → Booléen) et un indicateur de chaîne « Choix » (Moderne → Chaine et chemin).



Remarque :

- Pour changer le texte sur le bouton, double cliquer sur celui-ci et taper le nouveau texte ou clic droit Propriétés et changer le texte OFF.

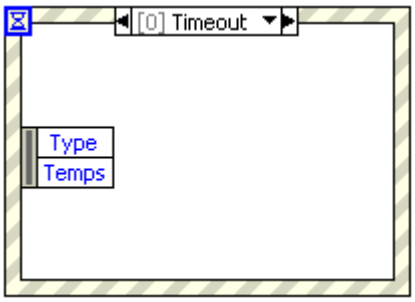
- On peut cacher le nom de variable (qui peut être différent du texte sur le bouton) donné lors de la création de celle-ci en faisant un clic droit sur la variable puis éléments visibles → étiquettes.

❖ **Sur le diagramme**, créer la boucle **While** dans Programmation → Structures

Créer dans cette boucle, une structure évènement dans Programmation → Structures

Éditer les trois évènements :

- « BONJOUR » souris relâchée pour cela faire un clic droit sur le texte Timeout puis « Ajouter une condition d'évènement … ». Dans la nouvelle fenêtre, dans la colonne « Sources d'évènement » choisir le bouton « BONJOUR » puis dans la colonne « Evènements » choisir « Souris relâchée » puis valider le tout par OK.

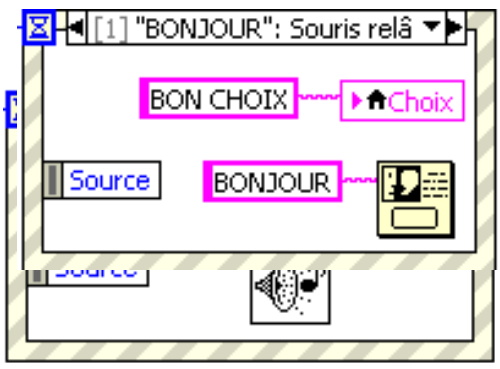


- « BIP» souris relâchée pour cela faire un clic droit sur le texte "«BONJOUR » : Souris relâchée " puis « Ajouter une condition d'évènement … ». Dans la nouvelle fenêtre, dans la colonne « Sources d'évènement » choisir le bouton « BIP » puis dans la colonne « Evènements » choisir « Souris relâchée » puis valider le tout par OK.

- Sortie de souris pour cela faire un clic droit sur le texte "« BIP» : Souris relâchée " puis « Ajouter une condition d'évènement … ». Dans la nouvelle fenêtre, dans la colonne « Sources d'évènement » choisir le bouton « <Ce VI> » puis dans la colonne « Evènements » choisir « Sortie de la souris » puis valider le tout par OK.

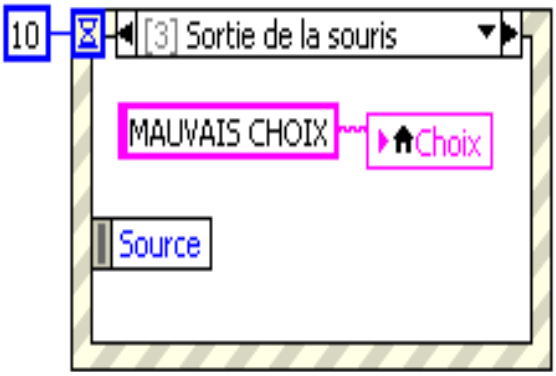
- Compléter chaque évènement :

- « BONJOUR » : Créer une variable locale de l'indicateur « Choix » puis créer une constante « BON CHOIX ». Ajouter « une boite de dialogue à un bouton » dans Programmation → Dialogue et interface utilisateur puis créer comme message, une constante de texte « BONJOUR ».



- « BIP » : Créer une variable locale de l'indicateur « Choix » puis créer une constante « BON CHOIX ». (On aurait pu copier cette partie en allant dans la condition « BONJOUR : Souris relâchée », en sélectionnant les éléments à copier (encadrer avec le bouton gauche enfoncer les éléments à sélectionner : les éléments auront un contour en pointillés ( ) puis tout en appuyant sur la touche « Ctrl » faire glisser les éléments (une copie est créée)) Ajouter « un bip» dans Programmation → Graphisme et son.

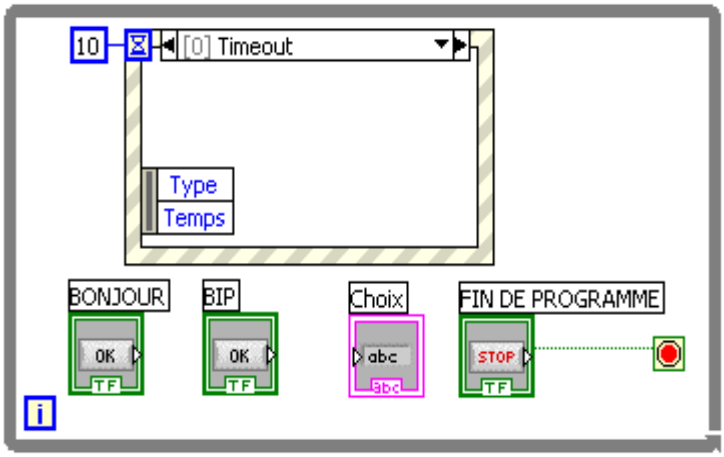
- « Sortie souris » : Créer une variable locale de l'indicateur « Choix » puis créer une constante « MAUVAIS CHOIX ».



Remarque :

- Si vous exécutez votre programme en mode « Exécution unique », celui-ci ne s'arrête pas lorsque vous cliquez sur « FIN DE PROGRAMME » car il est bloqué dans la structure évènement. Pour remédier à ce problème, il faut mettre un temps de scrutation (ex 10 ms).
- Il faut laisser l'évènement « Timeout » vide, qui correspond à ce que doit faire le programme si aucun évènement sélectionné n'est vrai.

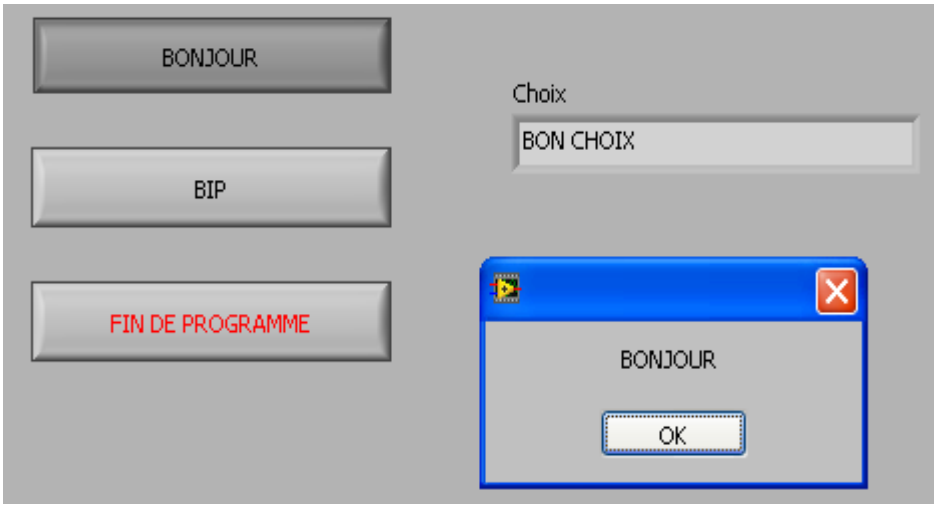
Relier le bouton « FIN DE PROGRAMME » à la condition d'arrêt de la boucle.



structure évènement.

Enregistrer le projet. Lorsque vous sauvegardez le projet tous les VIs sont sauvés.

Tester votre programme en appuyant sur le mode « Exécution Unique ».



TP N° 4 : Initiation à LabVIEW : Exemples d'applications

But de TP

Dans ce TP, vous apprendrez à manipuler avec le logiciel LabVIEW et voir la programmation la **structure bloc de séquence** et les **graphiques** :

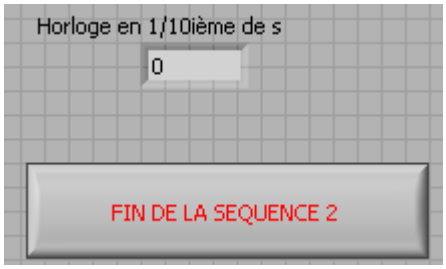
1) Structure blocs de séquence:

On souhaite créer un programme qui :

- Affiche 1ière séquence dans une boite de dialogue,
- Puis qui affiche une horloge en 1/10ième de seconde jusqu'à ce que l'on appuie sur le bouton « Fin de l'étape 2 »,
- Puis qui affiche 3ième séquence dans une boite de dialogue.

Sur la face avant,

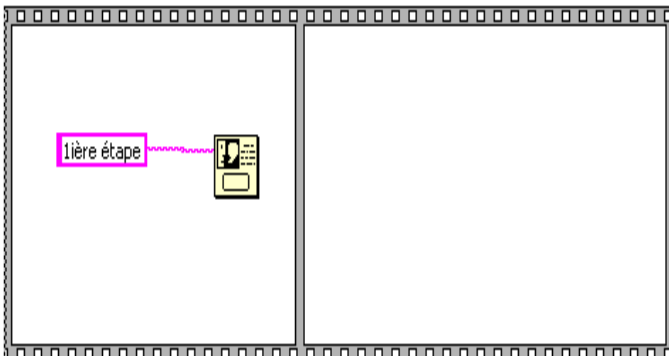
Ajouter un bouton **« *FIN DE L'ETAPE 2* »** (Moderne → Booléen) et un indicateur numérique « Horloge en 1/10 ième de s » (Moderne → Numériques)



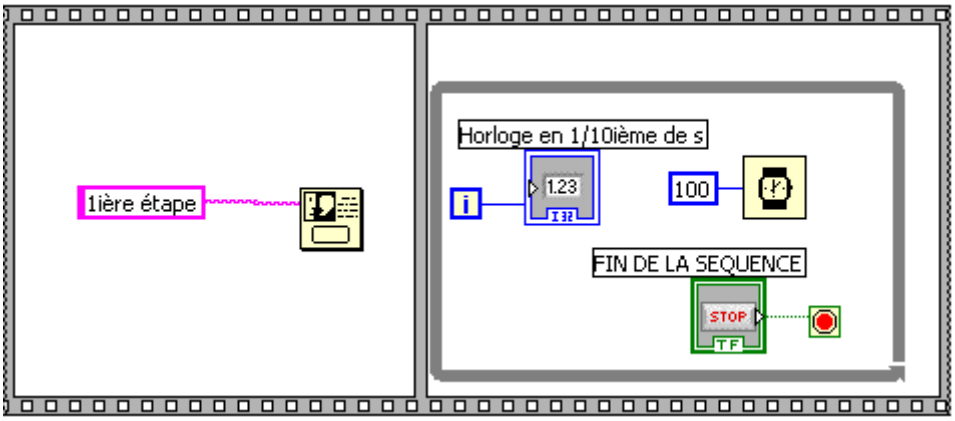
Sur le diagramme, créer la structure séquence déroulée dans Programmation → Structures

Ajouter dans cette étape « une boite de dialogue à un bouton » dans Programmation → Dialogue et interface utilisateur puis créer comme message, une constante de texte « 1ière étape»

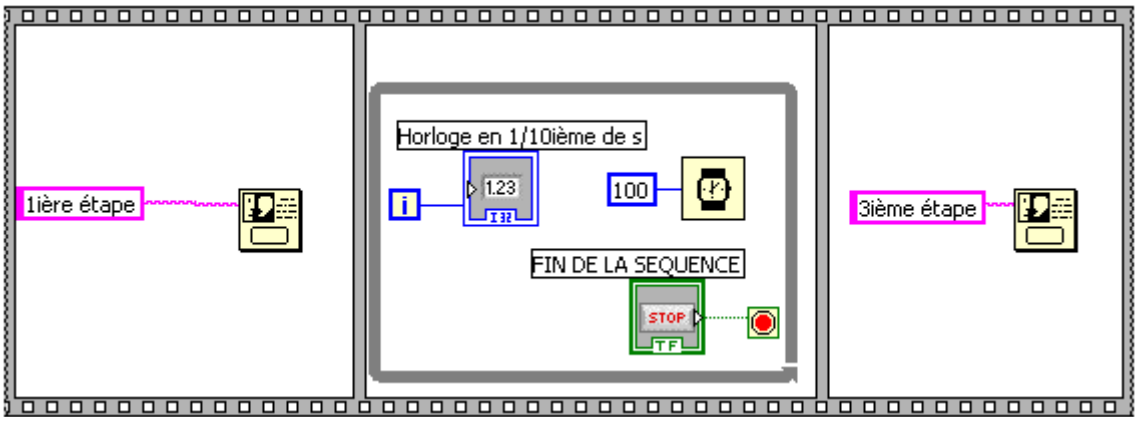
Ajouter une étape après en faisant un clic droit sur la partie verticale et en choisissant « Ajouter une étape après ». Dans cette seconde étape :

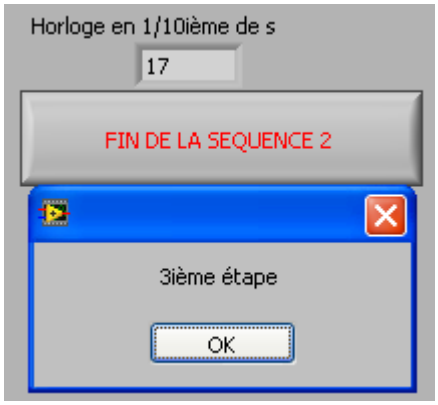


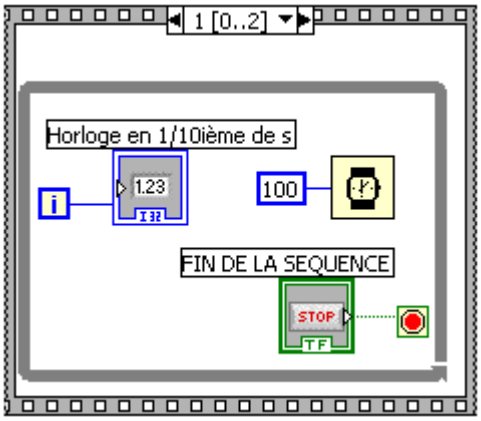
- Créer la boucle WHILE dans Programmation → Structures.
- Relier le bouton « FIN DE L'ETAPE 2 » à la condition de fin de la boucle While.
- Ajouter la fonction « Attendre (ms) » dans Programmation → Informations temporelles.
- Créer une constante = à 100 dans Programmation → Numériques et relier là à la fonction précédente.



- Relier l'indicateur « Horloge en 1/10ième de s » à numéro **i** d'itération de la boucle. Ajouter une étape après en faisant un clic droit sur la partie verticale et en choisissant « **Ajouter une étape après** ».
- Ajouter dans cette étape « une boite de dialogue à un bouton » dans Programmation → Dialogue et interface utilisateur puis créer comme message, une constante de texte « 3ième étape »







Enregistrer le projet. Lorsque vous sauvegardez le projet tous les VIs sont sauvés.

Tester votre programme en appuyant sur le mode « Exécution Unique »

Remarque :

Lorsqu'on fait des structures séquences, celle-ci peuvent prendre beaucoup de place sur le diagramme. Elles peuvent être remplacées par des séquences empilées (clic droit sur la séquence et « Remplacer par une séquence empilée »).

Exemple ci-contre : Etape 1 d'une séquence allant de l'étape 0 à 2.

2) Les graphiques :

Un système de surveillance de la température d'une étuve enregistre la température toutes les 15 mn. Le thermomètre envoie à l'ordinateur la valeur de la température sous forme d'une chaîne de caractères. Ces températures sont stockées dans un texte qui a la forme ci-contre :
On souhaite que le logiciel établisse automatiquement un rapport donnant l'évolution des dernières températures mesurées :

- ✓ les valeurs maxi, mini et la valeur moyenne de la température de la période.
- ✓ Un graphique de l'évolution de la température.

Explications :

Le programme aura la structure suivante :

a) Lecture du fichier :

On lit un fichier *.txt que l'on affiche dans l'indicateur texte lu.

b) Extraction des valeurs :

On veut obtenir un tableau des valeurs numériques des températures pour pouvoir trouver maxi, mini et moyenne.
Pour cela, on procède de la manière suivante :

FAIRE

A. Récupérer une chaîne de température dans le texte principal.
B. De cette chaîne extraire la sous chaîne exprimant la température,
C. Convertir cette sous chaîne en nombre,
D. La ranger dans un tableau de réels (doubles) « TabTemp ».

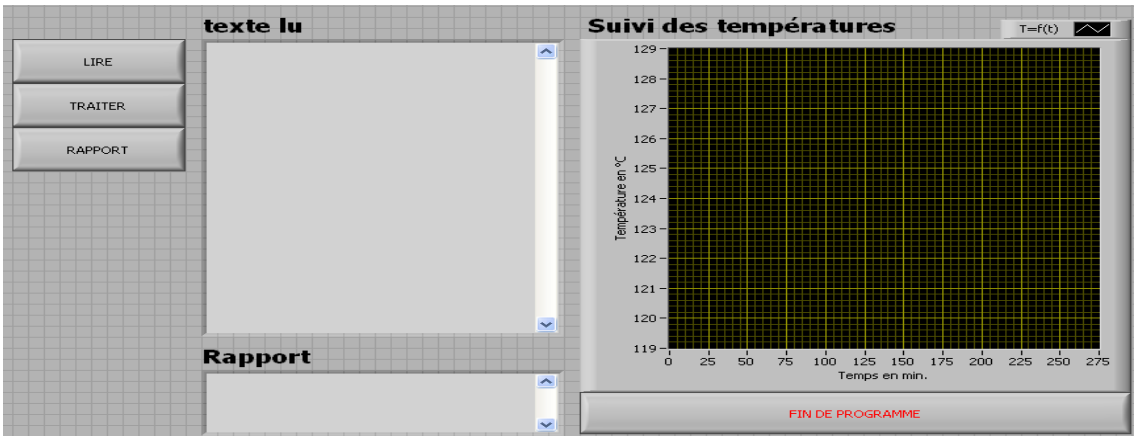
TANT QUE le texte lu n'est pas vide.

c) Constitution du rapport :

Lorsque le tableau D est complet, on applique alors les fonctions de tableaux permettant de trouver les valeurs désirées.

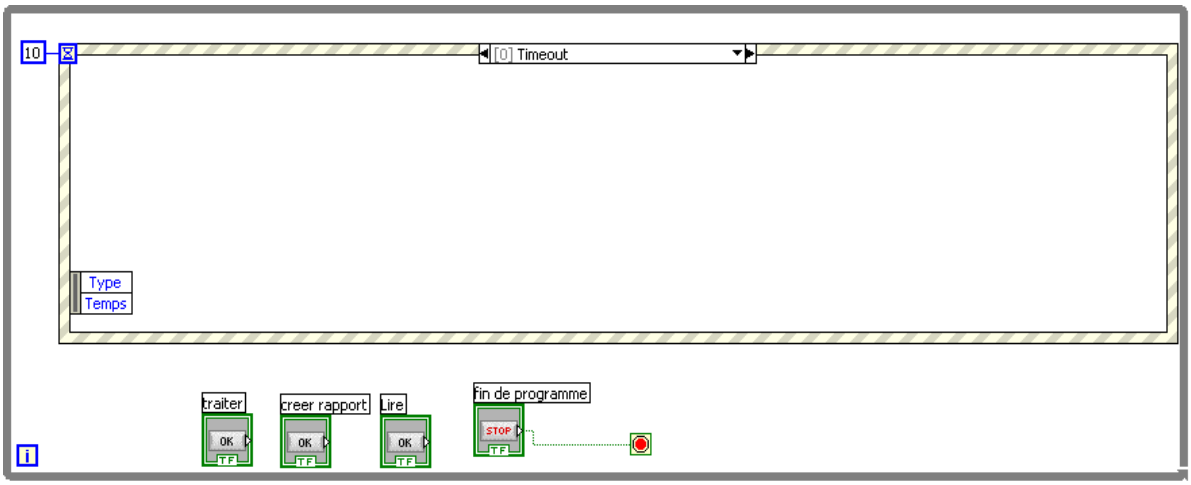
Sur la face avant, ajouter :

- Quatre boutons « LIRE », « TRAITER », « RAPPORT et « FIN DE PROGRAMME » (Moderne → Booléen) .
- Deux indicateurs de chaîne « Texte lu » et « Rapport » (Moderne → Chaine et chemin).
- Un graphe XY (Moderne → Graphe).



Sur le diagramme, créer la boucle WHILE dans Programmation → Structures Relier le bouton « FIN DE PROGRAMME » à la condition d'arrêt de la boucle.

- Ajouter dans cette boucle, une structure évènement dans Programmation → Structures
- Créer le temps de scrutation de 10 ms.



Editer les trois évènements :

- « LIRE » souris relâchée pour cela faire un clic droit sur le texte Timeout puis « Ajouter une condition d'évènement … ». Dans la nouvelle fenêtre, dans la colonne « Sources d'évènement » choisir le bouton « LIRE» puis dans la colonne « Evènements » choisir «Souris relâchée » puis valider le tout par OK.
- « TRAITER» souris relâchée pour cela faire un clic droit sur le texte "« LIRE » : Souris relâchée " puis « Ajouter une condition d'évènement … ». Dans la nouvelle fenêtre, dans

la colonne « Sources d'évènement » choisir le bouton « TRAITER » puis dans la colonne « Evènements » choisir « Souris relâchée » puis valider le tout par OK.

- « RAPPORT» souris relâchée pour cela faire un clic droit sur le texte "« TRAITER» : Souris relâchée " puis « Ajouter une condition d'évènement … ». Dans la nouvelle fenêtre, dans la colonne « Sources d'évènement » choisir le bouton « RAPPORT » puis dans la colonne « Evènements » choisir « Souris relâchée » puis valider le tout par OK.

Compléter chaque évènement :

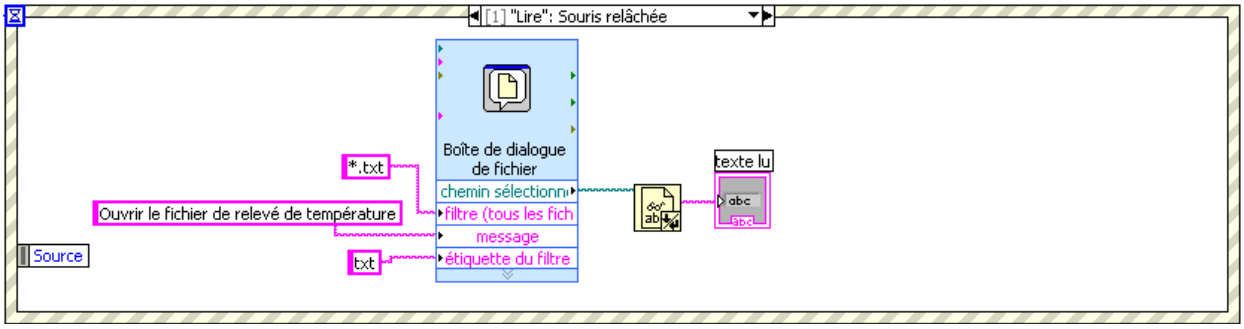
- « LIRE » :

✓ Créer une boite de dialogue de fichier (Programmation → I/O sur fichier → Fonctions de fichiers avancés).

✓ Créer 3 constantes de chaines : « *.txt », « txt » et «Ouvrir le fichier de température ».

✓ Relier la constante « *.txt » à l'entrée « filtre(tous les fichiers) », la constante « txt » à l'entrée « étiquette du filtre » et la constante «Ouvrir le fichier de température » à l'entrée « Message »

✓ Ajouter la fonction lire un fichier texte, relier l'entrée au chemin sélectionné de la boite de dialogue et la sortie à la variable « texte lu ».



- TRAITER :

✓ Créer une variable locale « Texte lu »

✓ Créer une boucle While (Programmation → Structures)

✓ Dans cette boucle :

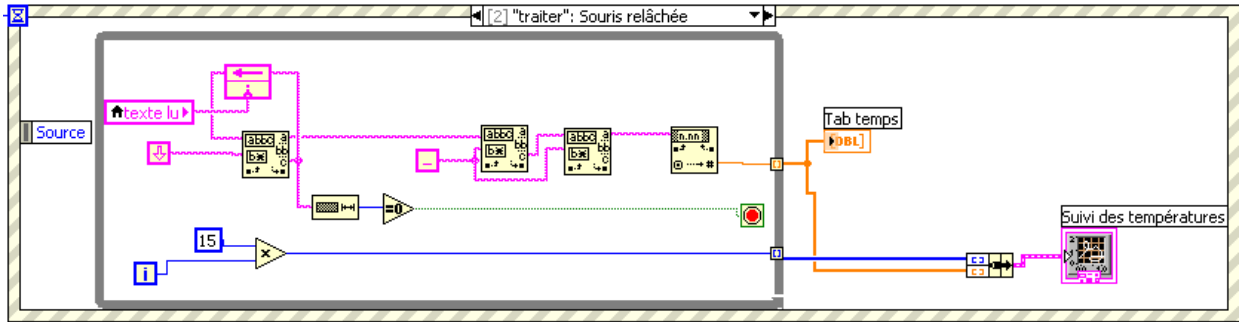
✓ Ajouter 3 fois la fonction « Rechercher une expression » (Programmation → Chaîne)

✓ Créer une « constante de retour à la ligne » et « une constante espace » (Programmation → Chaîne)

✓ Ajouter la fonction « Longueur d'une chaîne » (Programmation → Chaîne)

✓ Ajouter l'élément de comparaison « =0 » (Programmation → Comparaison)

- ✓ Ajouter la fonction « Chaîne Fract/exp en nombre » (Programmation → Chaîne → Conversion chaîne/nombre)
- ✓ Ajouter un noeud de rétroaction (Programmation → Structures). Un carré se place sur la boucle while qui correspond à la valeur d'initialisation. Quand i = 0, cette valeur est le texte lu.
- ✓ Créer une constante = à 15 (Programmation → Numériques)
- ✓ Ajouter la fonction « X » (Programmation → Numériques) (car chaque température est prise toutes les 15 minutes)
- ✓ Relier les différents éléments
- ✓ Ajouter la fonction assemblé (Programmation → Cluster et Variant)



Remarque : Il est impossible de relier les valeurs des températures au tableau car la boucle While n'envoie que la dernière valeur calculée. Pour que celle-ci conserve les valeurs pour chaque i et les range dans un tableau, il faut faire un clic droit sur le petit carré du nombre et faire « Activer l'indexation ».

- RAPPORT :

- ✓ Créer une variable locale « Tab temps »
- ✓ Ajouter la fonction « Max. et min. d'un tableau » (Programmation → Tableaux)
- ✓ Ajouter la fonction « Moyenne » (Mathématiques → Probabilités et statistiques)
- ✓ Ajouter 3 fois la fonction « Nombre en chaîne fractionnaire » (Programmation → Chaîne → Conversion chaîne/nombre)
- ✓ Pour le mini et le maxi, on prendra un chiffre après la virgule. Pour cela créer une constante = à 1 (Programmation → Numériques) que l'on reliera à l'entrée « Précision (6) » des convertisseurs.

Bibliographie

[1] F. Cottet, "LabVIEW Programmation et applications", Ed. Dunod, 2001.

[2] J. Jerome, "Virtual Instrumentation Using LabVIEW", PHI Learning Pvt. Ltd., 2010.

[3] R. W. Larsen, "LabVIEW for Engineers", Ed. PEARSON, 2010.

[4] F. Cottet, "Traitement des signaux et acquisition de données", Ed. Dunod, 1997.

[5] G. Johnson, "LabVIEW Graphical Programming, Practical Applications in Instrumentation and Control", Ed. McGraw Hill, 1997.