Table des Matières

Table des Figures

1. INTRODUCTION

Labview est un langage de programmation graphique dont les programmes sont appelés Instruments Virtuels plus communément VI pour Virtual Instruments. Ces VIs comportent 3 composants : la Face Avant, la Fenêtre Diagramme et l'Icône et ses connecteurs.

Il existe de nombreuses fonctions définies dans labview sous forme de bibliothèques de VIs, aussi labview permettra de développer des applications complexes utilisant des fonctions mathématiques, de traitement de signal, d'analyse de données, et les moyens de communications actuels avec les instruments (DAQ, GPIB, RS232, Ethernet, USB …).

Le but de ce manuel est de vous aider à comprendre l'architecture de labview et de ses composants : outils (palettes, Vis intégrés), Face Avant (FA) et diagramme, options … et d'être capable de réaliser à court terme des projets utilisant les principales capacités de programmation du logiciel, c'est-à-dire :

- travailler avec des tableaux, clusters et les différentes structures
- créer des Sub-Vis (sous programmes ou fonctions)
- construire une application utilisant l'acquisition de données (DAQ) ou la communication avec un ou des instruments.

2. Ouverture – Nouveau VI

2.1. Fenêtre de choix

Double cliquer sur le raccourci Labview sur le bureau s'il est présent sinon :

→ Démarrer

→ Programmes

→ National Instruments

→ Labview

Une fenêtre apparaît. Celle-ci vous permet de créer un nouveau VI ou un projet, d'ouvrir un VI, de configurer Labview ou de rechercher de l'aide.

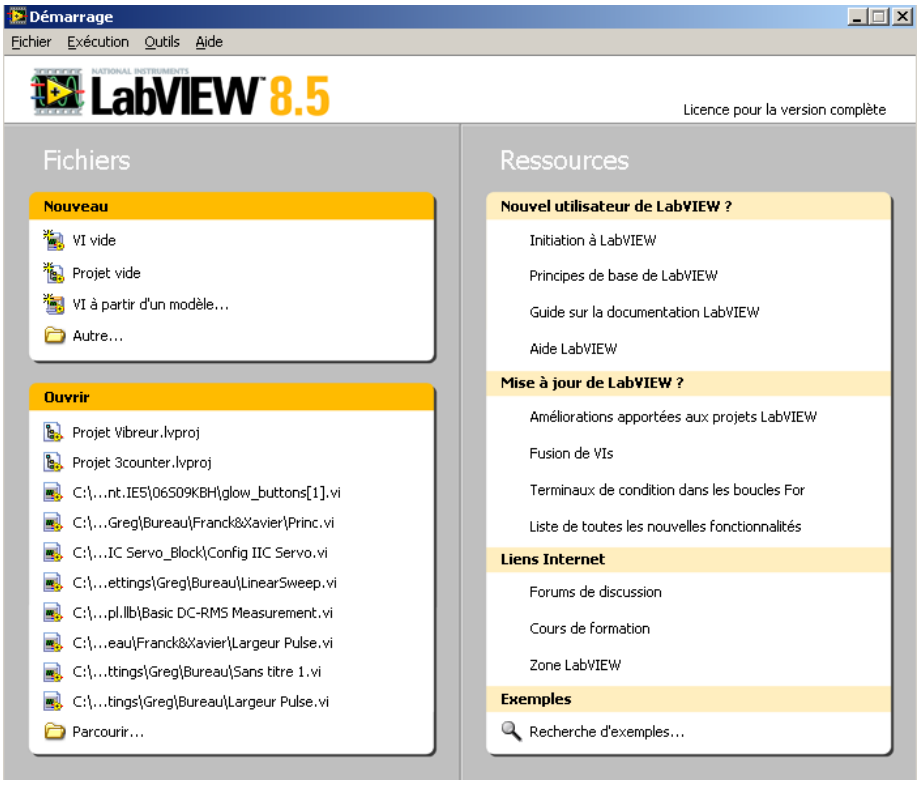


Figure 1 : Fenêtre ouverture

▪ Cliquer sur VI à partir d'un modèle : une nouvelle fenêtre s'ouvre :

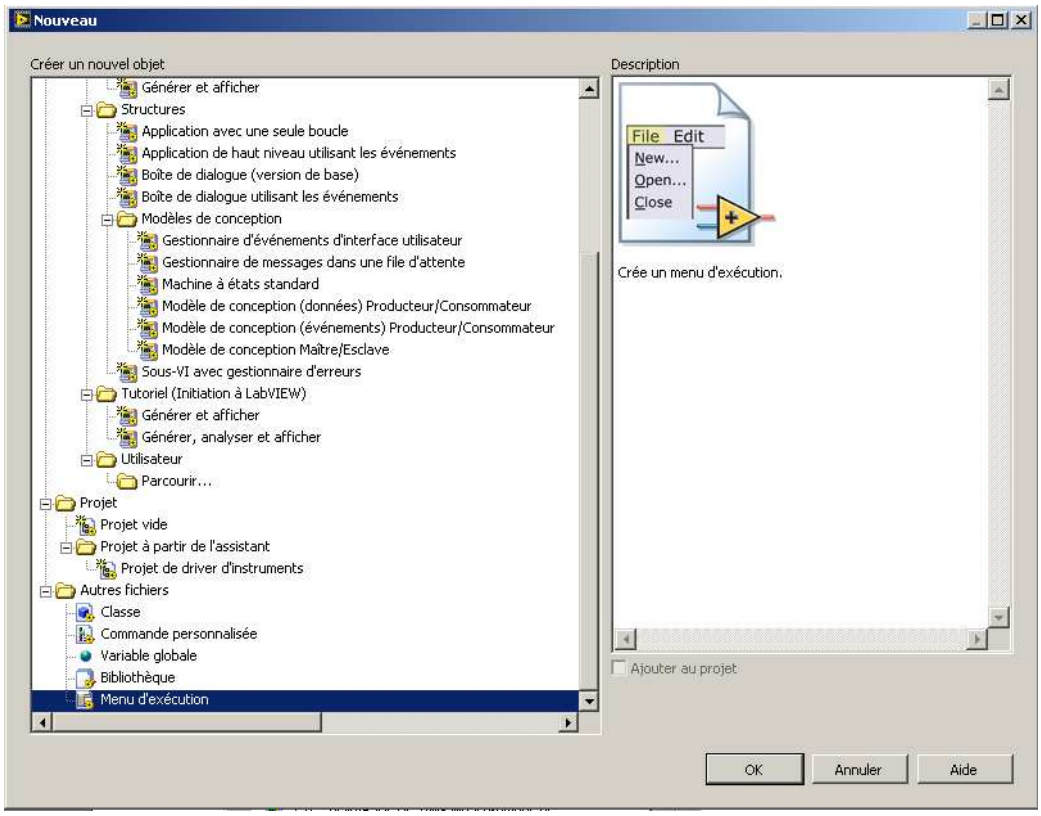


Figure 2 : Nouveau VI

A partir de cette fenêtre vous pouvez choisir quel type de VI vous allez créer, il existe différents modèles de VIs suivant le type de programme que vous voulez créer. Une description vous permet de comprendre le modèle. Dans ce manuel nous ne travaillerons pas à partir de modèles.

▪ Fermer la fenêtre et développer le menu 'Outils'.

Dans ce menu, l'utilisateur peut lancer Measurement and Automation eXplorer (MAX) qui est le logiciel «drivers» des cartes NI, pour l'instant cliquer sur Options Labview.
Les différentes options de labview peuvent être modifiées ici, notamment la représentation des palettes d'outils, le choix de visualiser ou non sous forme d'icônes les terminaux, le routage automatique.

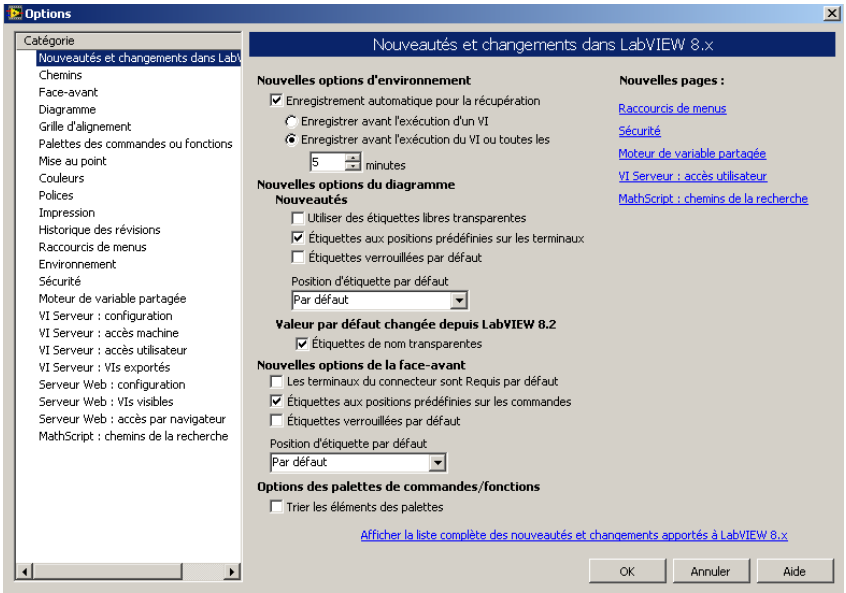


Figure 3 : Fenêtre Option

- Fermer la fenêtre et aller sur ‘Exemples’ ‘Recherche d’exemples’.

Une partie intéressante de labview est l’intégration d’une bibliothèque d’exemples. Cliquer sur Recherche d’exemples.

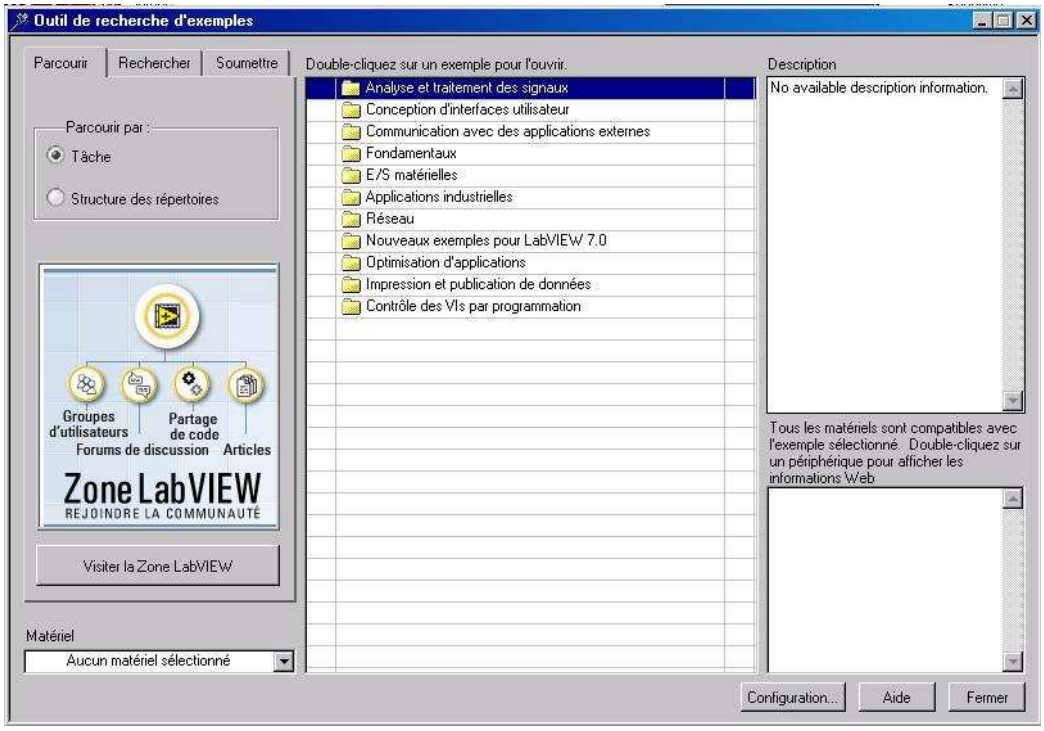


Figure 4 : Fenêtre Recherche d’exemples

Depuis cette fenêtre de nombreuses options s’offrent à vous, recherche d’exemples en parcourant divers types de programmes, recherche d’exemples par mots clés avec la possibilité d’inclure une recherche sur le site de NI et de soumettre vos exemples à NI. En programmation labview n’hésiter donc pas à rechercher des exemples pour les inclure dans vos programmes afin d’éviter une perte de temps dans la création d’un programme existant.

2.2. Nouveau VI

Ouvrir un VI vide : pour cela fermez l’outil de recherche d’exemple, aller sur ‘VI vide’. 2 fenêtres s’ouvrent, suivant les options choisies elles peuvent se superposer. Une des fenêtres est la Face Avant (FA) c'est-à-dire la fenêtre que verra l’utilisateur lorsque le programme fonctionnera, elle contient les Contrôles (Entrée utilisateur) et les Indicateurs (Sorties du code), ceux-ci se retrouvent sur le diagramme qui contient le code et les composants reliés entre eux par des «Fils».

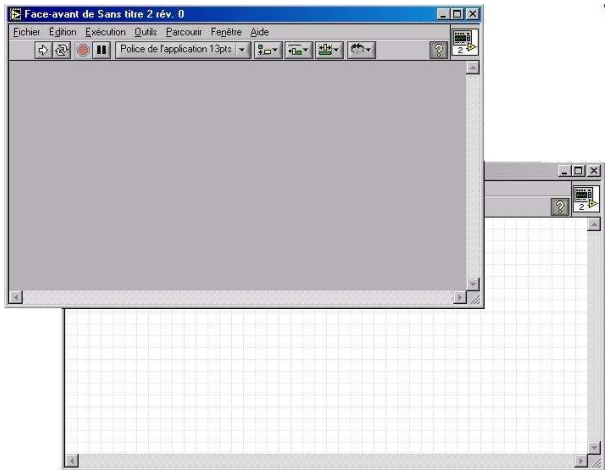


Figure 5 : Face-avant et diagramme

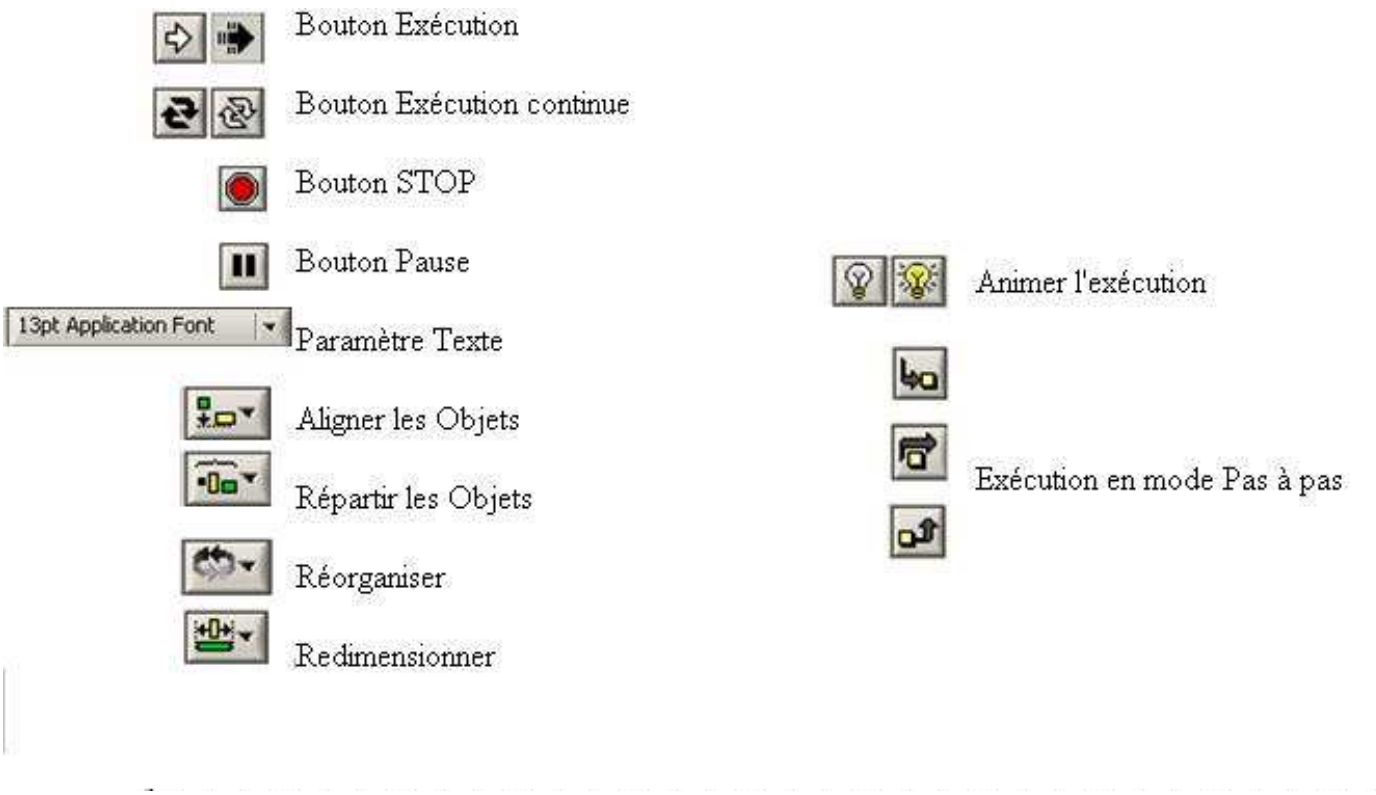


Figure 6 : Bouton de commande du diagramme

Suivant la fenêtre sur laquelle vous vous trouvez 2 palettes sont disponibles : palette de contrôles pour la FA et palette de fonctions et d'outils pour le diagramme.

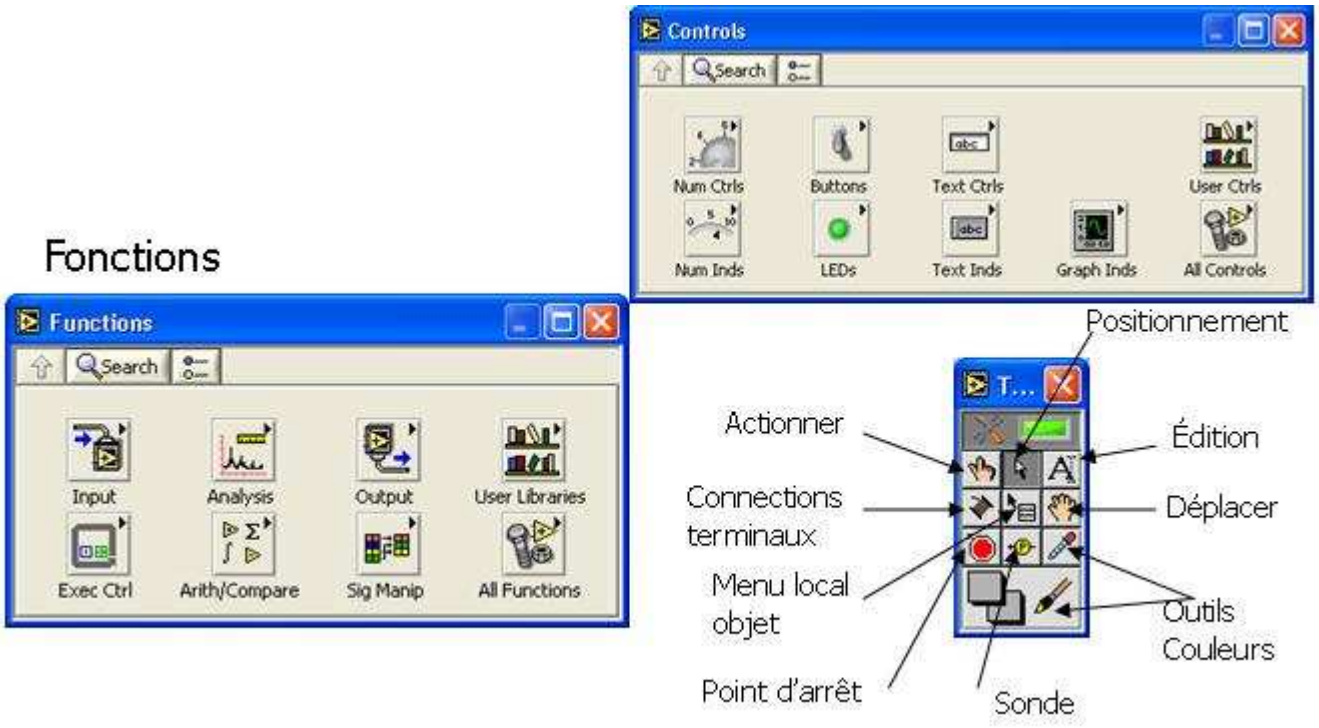


Figure 7 : Palettes de Fonctions - Contrôles et outils

La visualisation de ces palettes est différente suivant les options définies. Si les palettes ne sont pas visibles, click droit de la souris dans les fenêtres. (La palette présentée ici est la palette de la version 7 vous permettant de faire la relation entre icônes et fonctions), la version 8 apparaît comme l'utilisateur le veut.

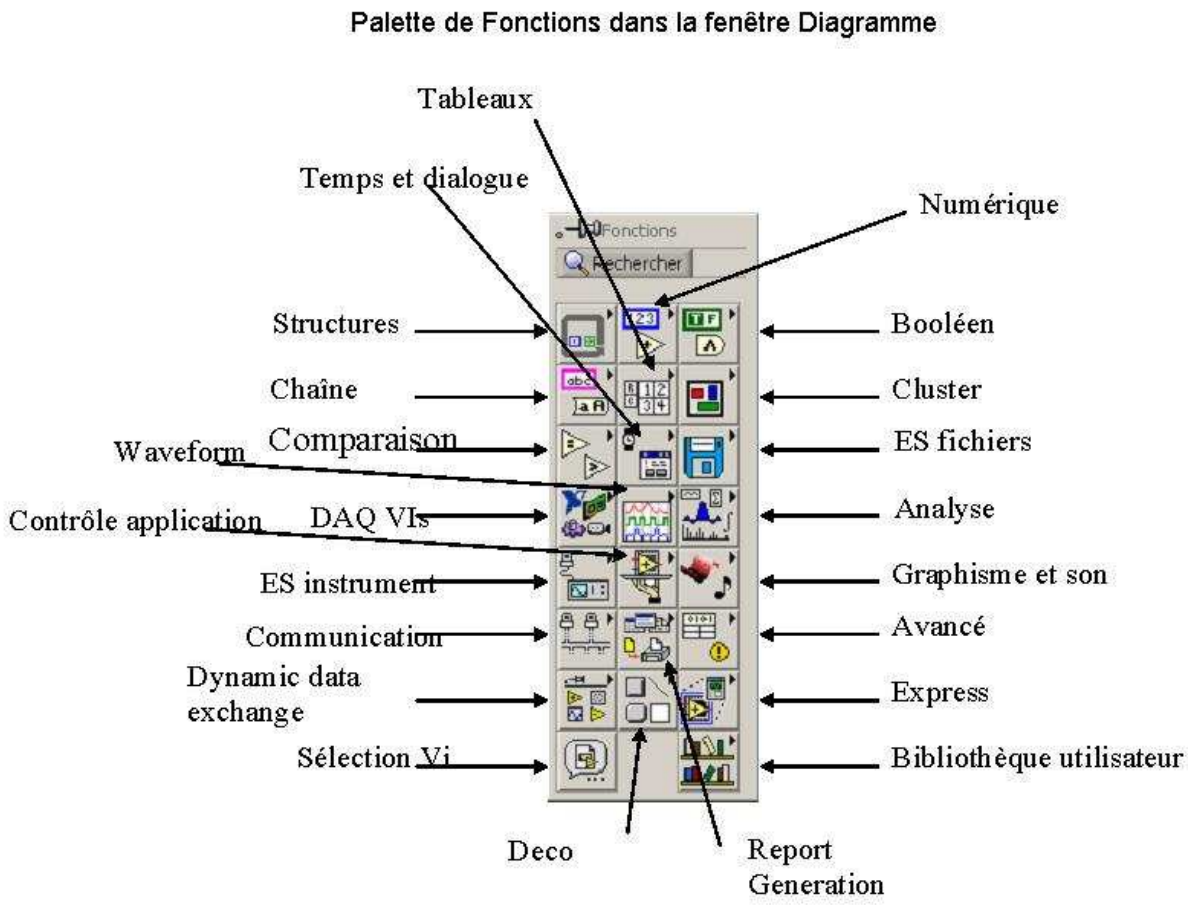


Figure 8 : Palette de Fonctions

2.3. Règles de connexion

Au niveau du diagramme quelques règles de connexion seront à respecter notamment en essayant de croiser le moins possible les câbles et à compacter le plus possible le code.

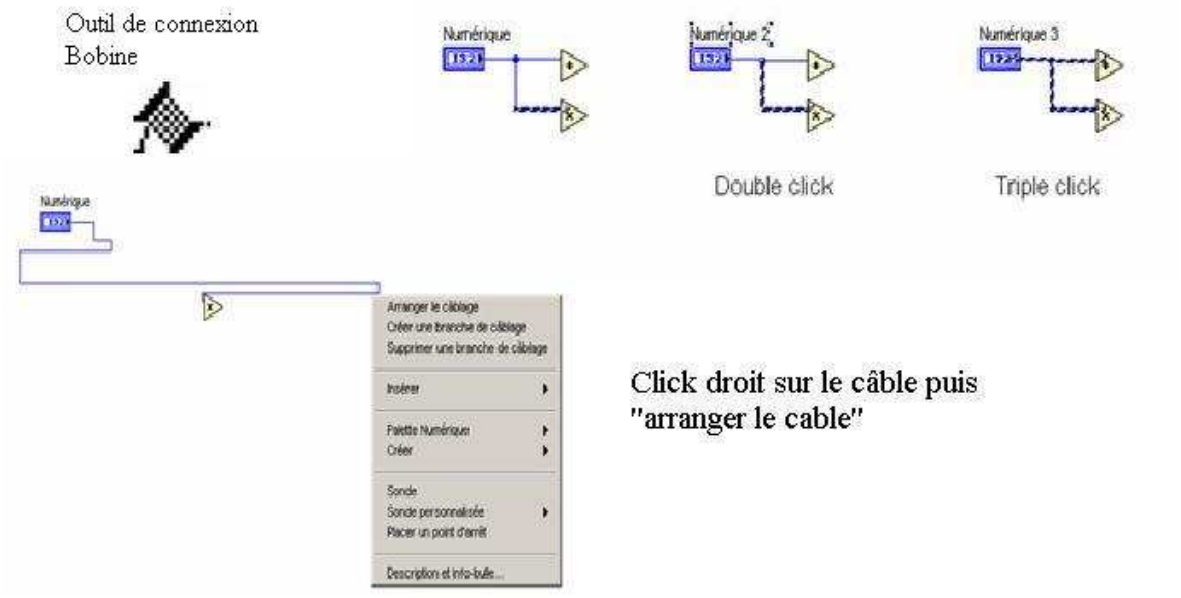


Figure 9 : connexion

3. CREATION D'UN VI

3.1 VI

Pour comprendre comment nous pouvons programmer sous labview nous allons étudier un exercice qui est le suivant :

Créer un VI simulant une prise de température à partir d'un capteur à sortie analogique linéaire 0-10Volts pour -15°C et 150°C. Ce VI affichera la température en °C, °F et K. (rappel : °F=°C*(9/5)+32).

L'algorithme sera le suivant :
Début
Lecture commande V
Tc=16.5 * V - 15
Tf=Tc * (9/5) + 32
Tk=Tc + 273.15
Fin

Il existe plusieurs façons de réaliser ce programme. De nombreuses fonctions sont disponibles, vous serez amenés à faire des choix basés sur votre connaissance des différentes fonctions.
Faire un click droit sur la face avant afin de créer une commande numérique, de la même façon placer 3 indicateurs numériques.

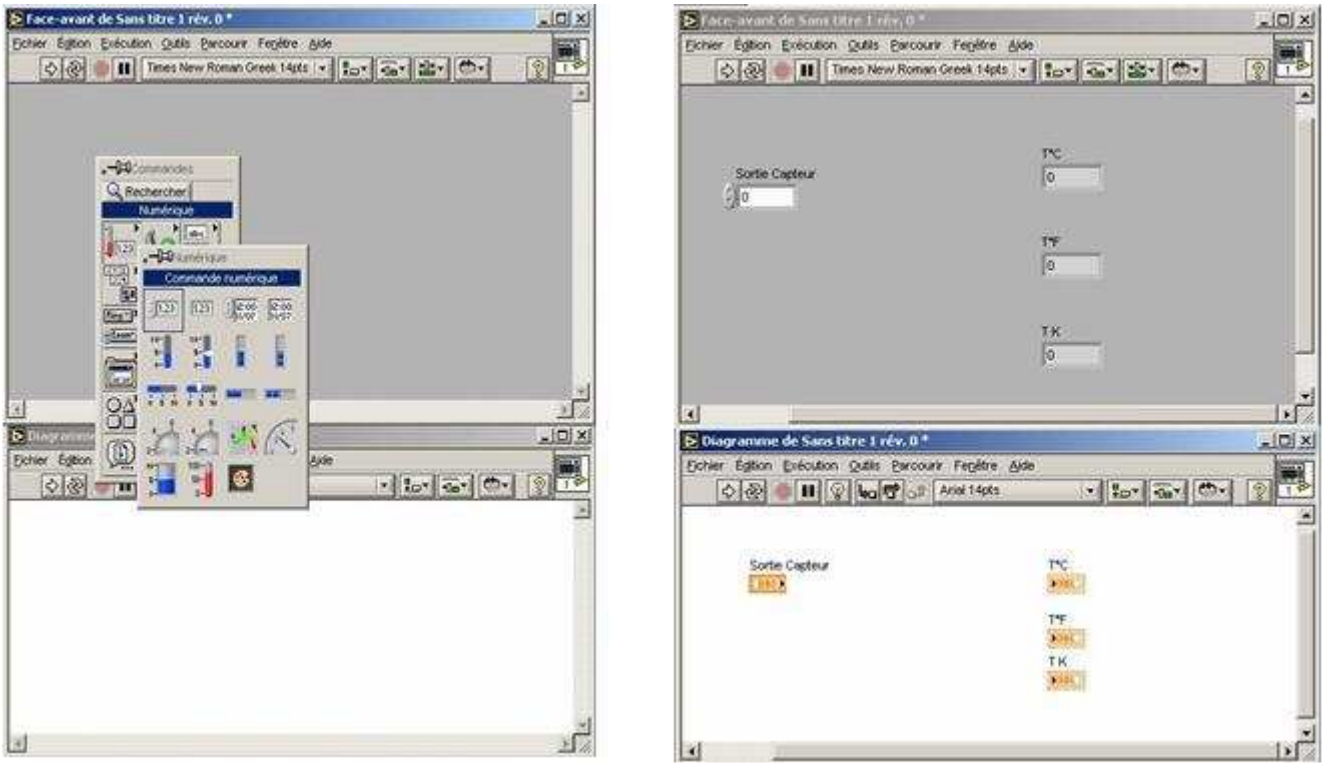


Figure 10 : Placement de commandes et indicateurs

Observer que dès que vous créez une commande ou un indicateur en FA vous retrouvez cette commande ou indicateur au niveau du code. Observer bien la **petite flèche noire** qui vous indique si nous sommes en présence d'un indicateur (valeur à rentrer dans l'indicateur) ou d'une commande (valeur sortante de la commande) et le contour gras de la commande. Sous labview les couleurs, l'épaisseur des fils, le contour nous permettent d'avoir des informations essentielles. Se référer aux annexes.

Maintenant plusieurs méthodes s'offrent à vous, utilisez dans un premier temps les outils simples numériques, les nœuds d'expression puis la boite de calcul (la boite de calcul se situe dans les Structures).

Figure 11 : Palette Fonctions Numériques

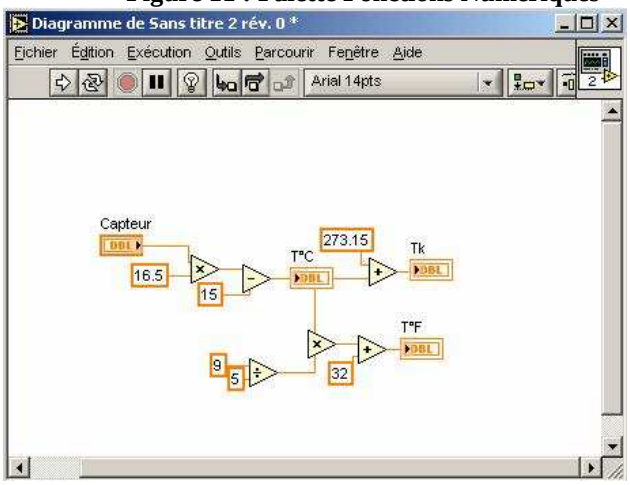


Figure 12 : Code 1

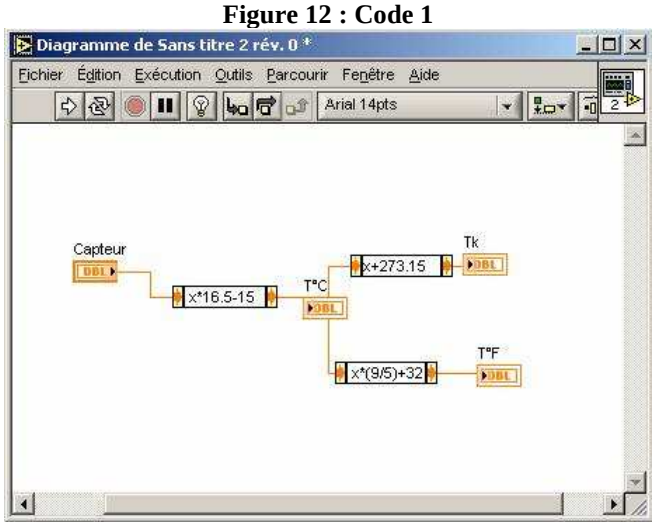


Figure 13 : Code 2

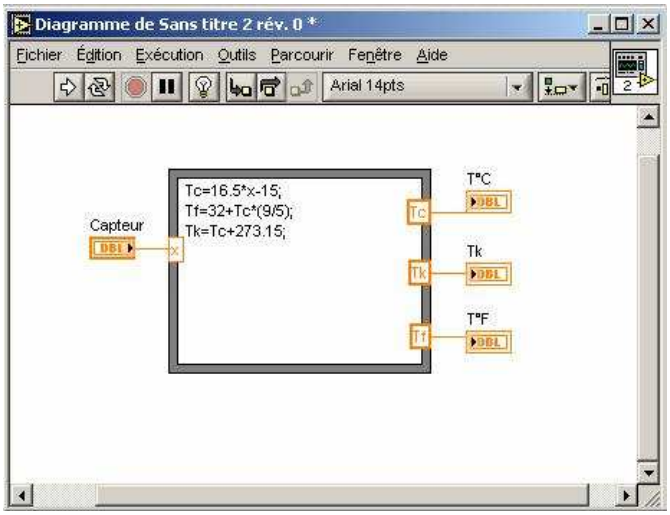


Figure 14 : Code 3

Vous avez remarqué que les commandes ou indicateurs peuvent être de différentes couleurs ; sous Labview à chaque type est associée une couleur (se référer à la plaquette de ressource d'aide).

A ce stade nous pouvons penser que le programme est fini, nous simulons un capteur, or celui-ci est défini entre 0 et 10 V, pourtant l'utilisateur peut rentrer une commande inférieure à 0V et supérieure

à 10V. Chaque commande possède des **propriétés** dont certaines sont directement accessibles à l'aide d'un **click droit** sur la commande.

- Modifier les paramètres de la commande sortie capteur afin que l'utilisateur n'entre pas de données hors gamme.

→ Click droit sur la commande
→Propriétés
→Gamme

ou → Click droit sur la commande
→Gamme des données

En programmation Labview ayez toujours l'aide contextuelle ouverte, cela vous facilitera la compréhension des fonctions. (CTRL+H).

3.2. Sous VI

Un sous VI est équivalent à un sous programme ou fonctions d'autres langages de programmation. Il permet l'utilisation d'un code récurrent dans le programme principal, de plus il requiert moins de mémoire, les sous VIs permettent de déboguer plus facilement un programme et peuvent être utilisés par plusieurs programmes. Ils permettent aussi de simplifier le diagramme.

Pour construire un sous VI, **2 solutions** s'offrent à vous : (ne faites rien pour l'instant)

- Sélectionnez la partie de code à mettre en sous programme puis Edition – Créer un sous - VI, le sous - VI est créé à l'intérieur du code avec ses connexions. **Attention le sous - VI n'est pas enregistré.**

- Aller sur l'icône en haut à droite de la face avant, click droit Visualiser le connecteur, des connecteurs apparaissent, sélectionner l'outil Bobine, aller sur la FA, click gauche sur une commande puis un connecteur et cela pour les indicateurs aussi. Vous venez de créer directement un sous VI.

Ces solutions sont à utiliser suivant votre façon de programmer. Si vous êtes dans votre partie principale et vous vous apercevez que du code va être répétitif utilisez la première méthode. Si par contre dès le départ une fonction sera utilisée à de nombreuses reprises alors créez votre sous - VI comme un VI et reliez les connecteurs.
Les sous VI peuvent être des programmes à part entière, la différence réside dans le fait qu'un connecteur est créé. En double cliquant sur l'icône vous pouvez modifier la représentation de celui-ci.

- Créer un sous VI à partir de votre programme. Sélectionnez la partie à mettre en sous programme : puis Edition - Créer un sous - VI

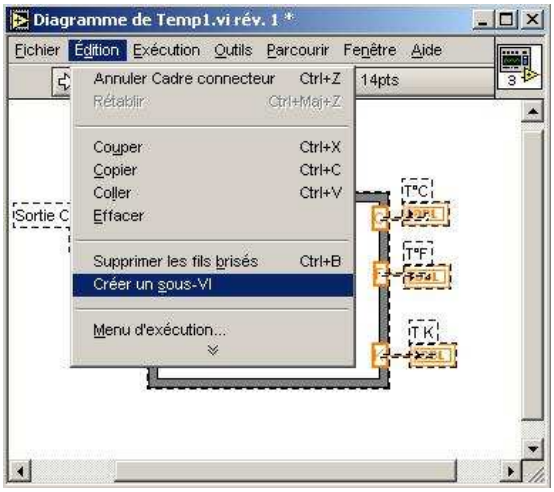


Figure 15 : Création Sous VI à partir du diagramme VI principal

Une des particularités de labview est qu'il est multitâche, c'est-à-dire que plusieurs parties de code peuvent s'exécuter en même temps. Nous pouvons placer ce même sous VI plusieurs fois en parallèle dans le code pour qu'ils s'exécutent en même temps. Pour cela nous devons modifier une propriété d'exécution.

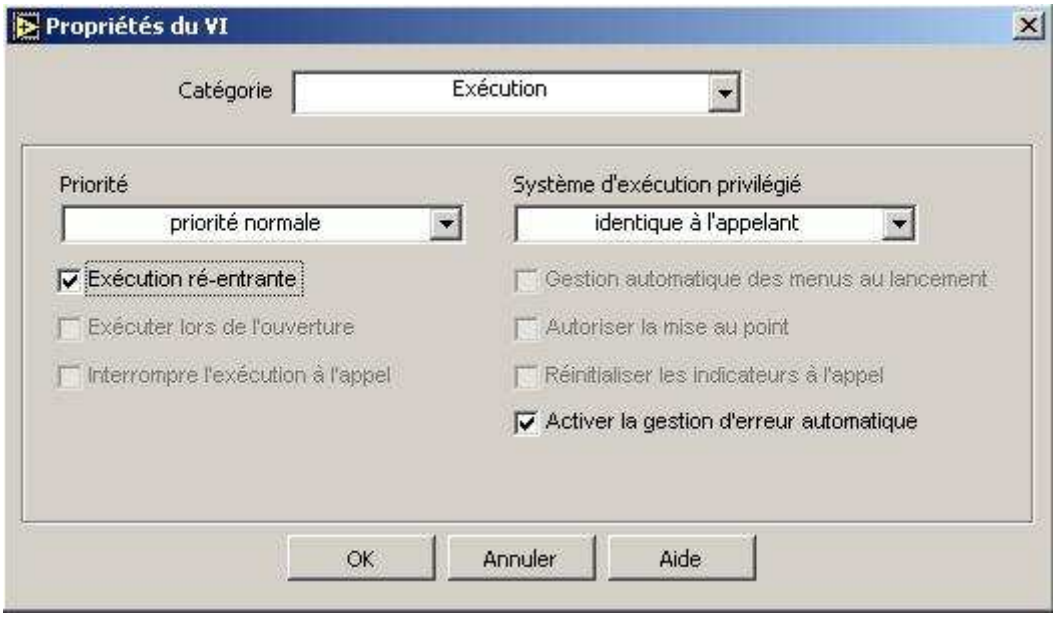


Figure 16 : Propriétés d'un VI ou sous VI

Le sous VI sera mis en Exécution réentrante ainsi plusieurs instances de celui-ci s'exécuteront en même temps si le code le demande.

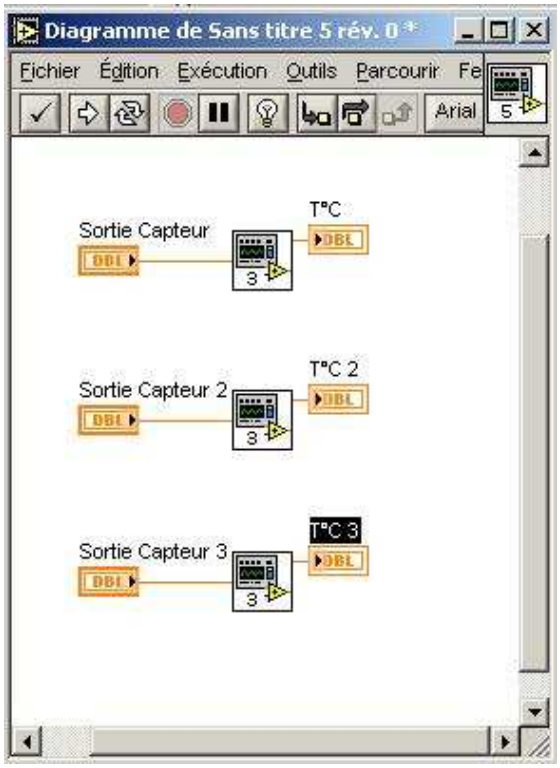


Figure 17 : plusieurs instances d'un sous VI

Nous pourrons visualiser en même temps les 3 sorties températures.

4. LES STRUCTURES : While, For, Séquence, Condition, Evènement.

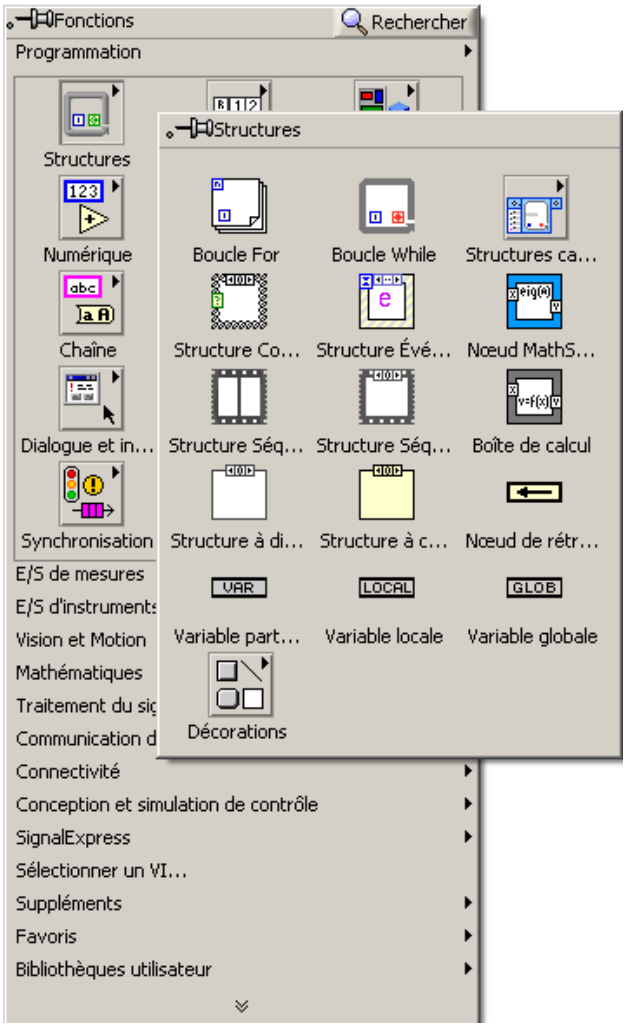


Figure 18 : Palette des Structures

4.1. While :repeat until

Même boucle que les autres langages de programmation tout en ayant des particularités supplémentaires : registre à décalage et indexation.

- Reprendre votre programme et sélectionner la boucle pour entourer votre code.

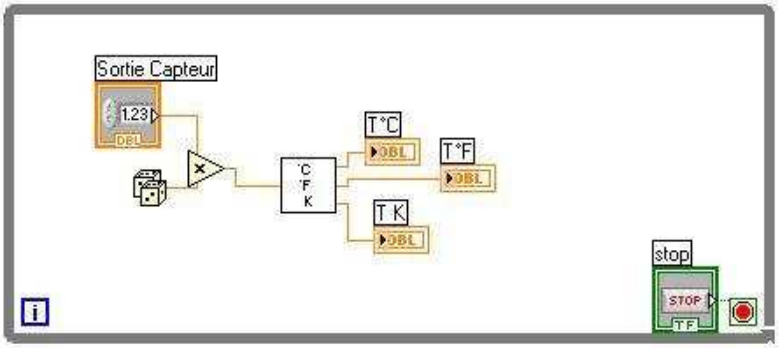


Figure 19 : Boucle While

L'algorithme est le suivant :

Début
Repeat
    Lecture capteur
    Calcul des températures et affichage
Until Stop=true
Fin

Vous êtes obligés de créer une commande STOP pour pouvoir arrêter la boucle. Tant que STOP est faux la boucle continue. Les boutons STOP possèdent différentes actions mécaniques.

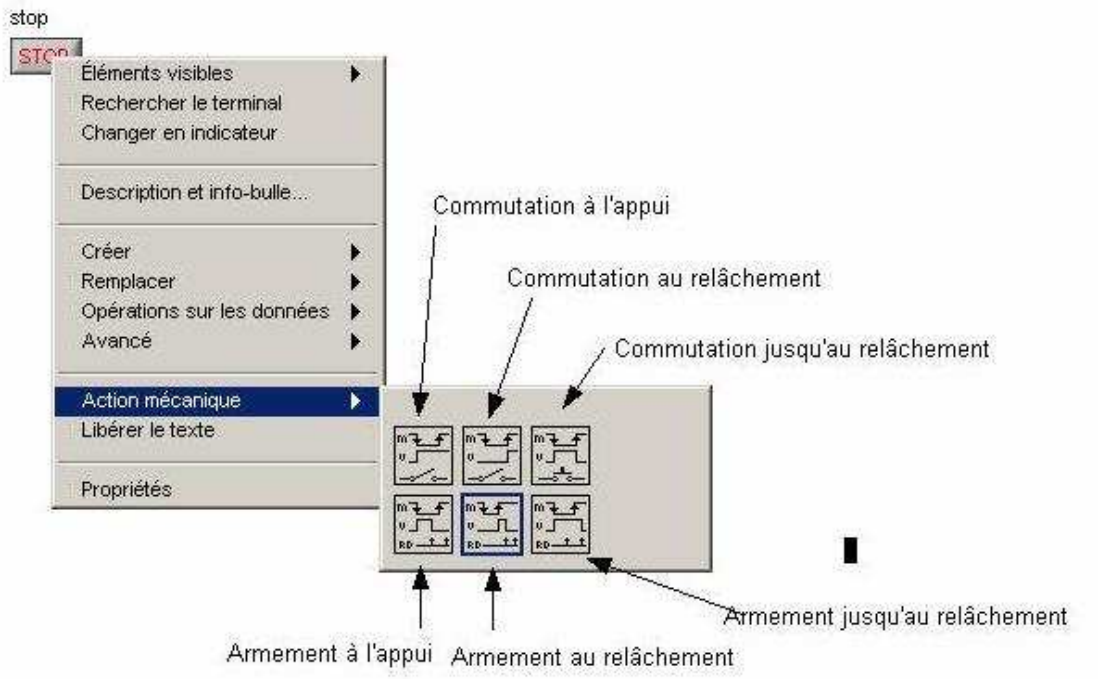


Figure 20 : Actions mécaniques de boutons

Si vous ouvrez le gestionnaire des tâches Windows pendant que votre VI est en exécution vous vous apercevez que 100% du CPU est utilisé (si monocoeur) du fait que Labview teste à chaque tour de boucle le plus rapidement possible le bouton STOP. Afin de **libérer** quelques **ressources** incluez dans la boucle une **temporisation** de 10 ms et observez le gestionnaire des tâches. Il serait intéressant d'avoir après la fin de la boucle toutes les valeurs de température, pour cela reliez vos indicateurs à la boucle While, un carré plein apparaît sur la boucle (**tunnel**), faites un click droit puis Activez l'indexation.

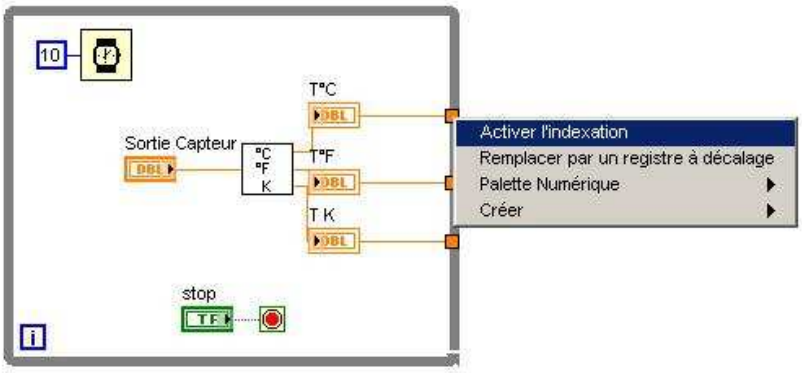


Figure 21 : Indexation

Vous avez directement un tableau de toutes vos valeurs en sortie. Remarquez **l'épaisseur du fil reliant le tableau**. (Remarquez le carré plein devient un carré dans lequel se trouve des crochets indiquant que nous sommes en présence d'un tableau).

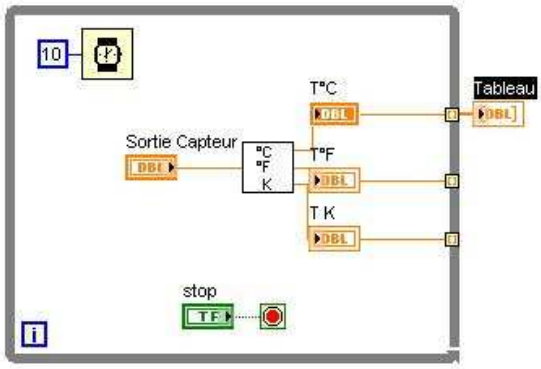


Figure 22 : Tableau en sortie d'indexation

Une autre particularité intéressante est le **registre à décalage** : click droit sur la boucle While, Créer un registre à décalage.

Réalisez le code suivant :

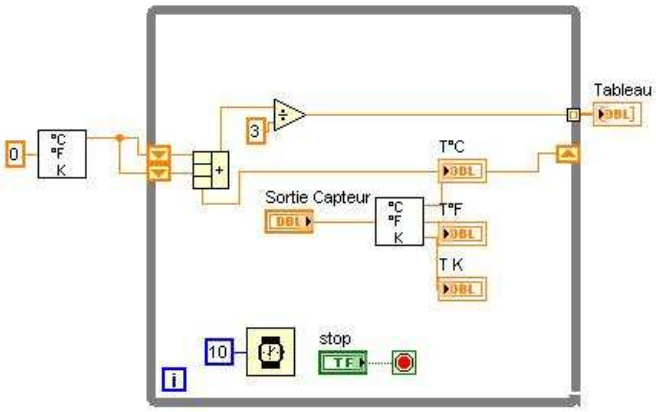


Figure 23 : Registre à décalage

Le registre à décalage nous permet ici de faire une moyenne sur les 3 **dernières valeurs**. Attention à initialiser les registres à décalage si nécessaire.

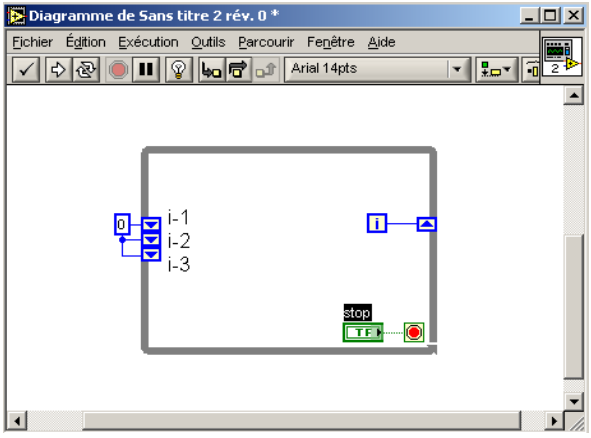


Figure 24 : Explication registre à décalage

Attention à la condition d'arrêt qui est traitée avant la fonction de la boucle, à l'appui sur le bouton Stop conduit à la fin de l'exécution en cours et à une autre exécution, ce comportement n'est pas un comportement souhaitable, pour résoudre ce problème nous utiliserons une structure If pour tester le bouton et permettre ou non l'exécution de la fonction.

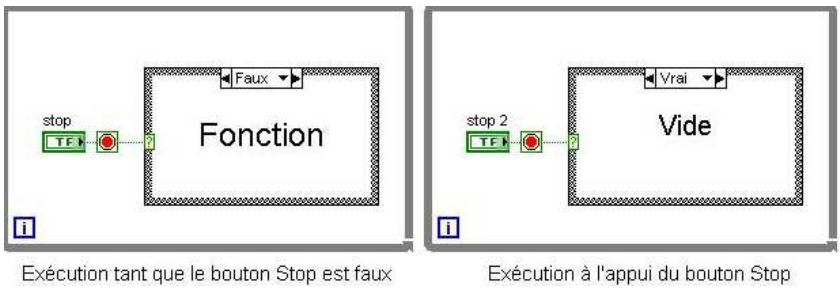


Figure 25 : Condition d'arrêt

Attention aussi aux boucles infinies provoquées par une mauvaise compréhension du flux de données.

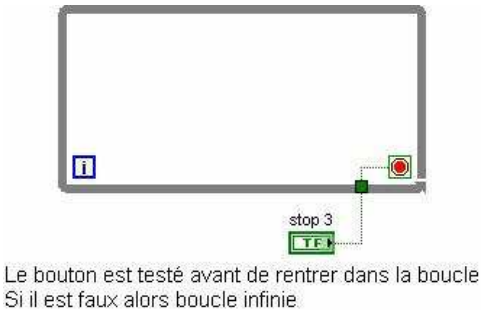


Figure 26 : Boucle infinie

4.2. FOR

La boucle For est comparable à la boucle While sauf que la fin de la boucle est connue, nous retrouvons l'indexation et les registres à décalages.

- Réaliser le code suivant :

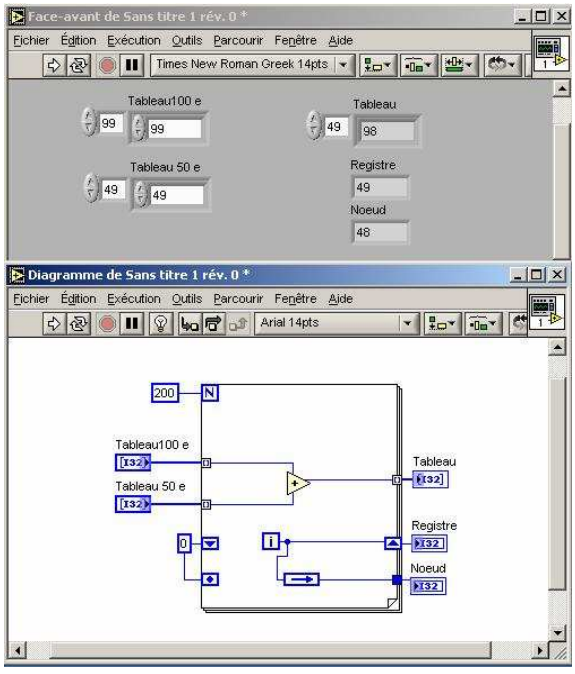


Figure 27 : Boucle For et propriétés

Création des tableaux :
→ diagramme
→fonctions tableaux
→ constante tableau
→ fonctions numériques
→ commande ou indicateur numérique suivant le cas.

Observez bien le résultat des indicateurs.

Conclusion : l'indexation sur les boucles For est prioritaire si N > au nombre de lignes des tableaux, le plus petit des tableaux est prioritaire sur les autres. Le nœud de rétroaction s'arrête à N-2 tandis que le registre à décalage à N-1.

N est le terminal de décompte qui indique combien de fois le diagramme doit être répété.

i est le terminal d'itération qui indique le nombre de fois que la boucle s'est exécutée.

4.3. Condition

La structure peut être considérée comme **un IF ou un CASE**, elle n'est pas limitée à un seul type de données, par contre vous devez définir **un choix par défaut** si toutes les conditions ne sont pas implémentées dans la structure.

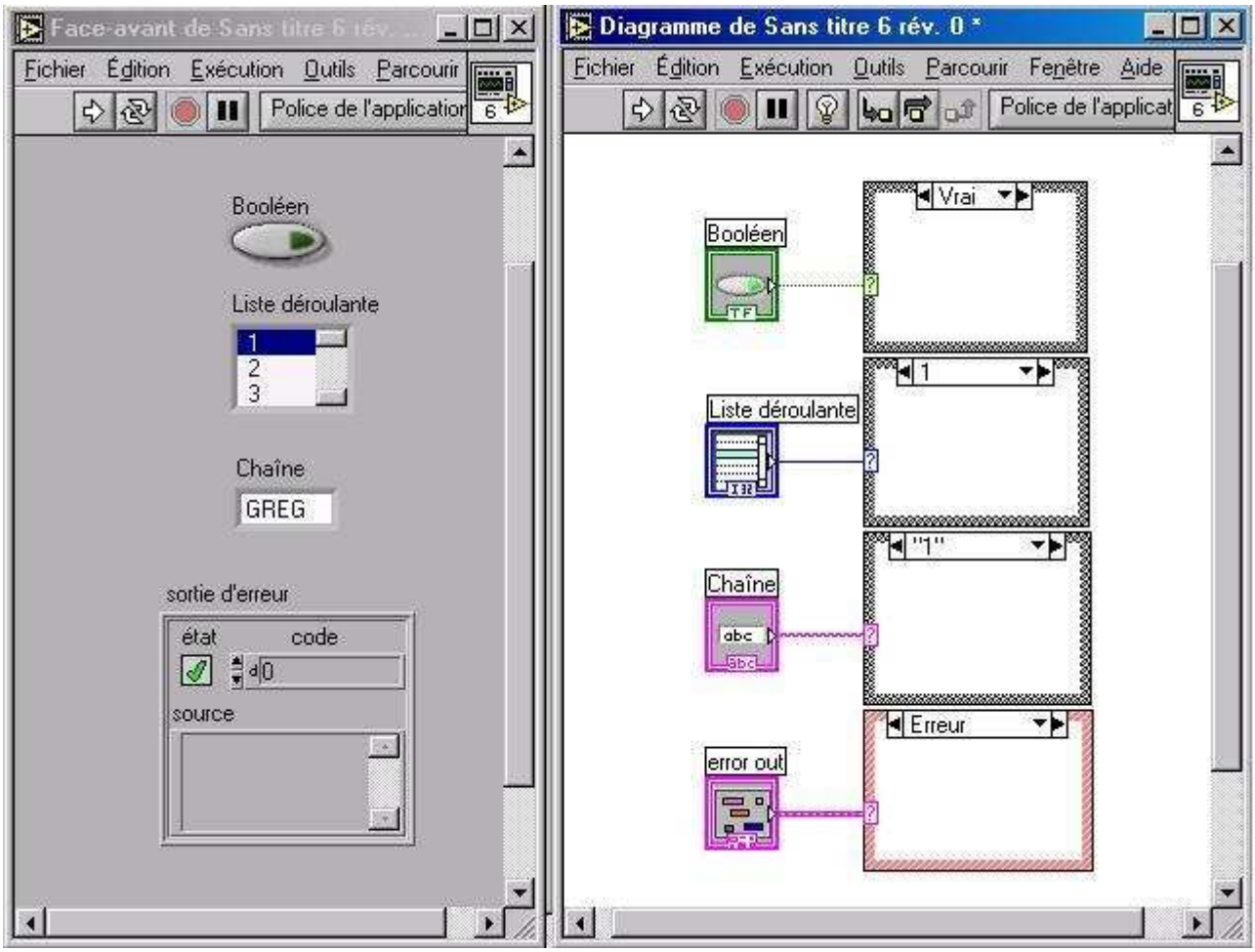


Figure 28 : Condition

4.4. Séquence

La structure Séquence permet de séparer des parties de code qui devront s'exécuter séquentiellement, mais aussi de simplifier le code en superposant ces parties. 2 types de séquence existent depuis la version 7 : séquence empilée (ancienne version) et séquence déroulée. Afin de faire passer des variables d'une séquence à une autre il est possible de créer ce qui est appelé une **variable locale de séquence** (click droit sur la séquence Créer une variable locale de séquence).

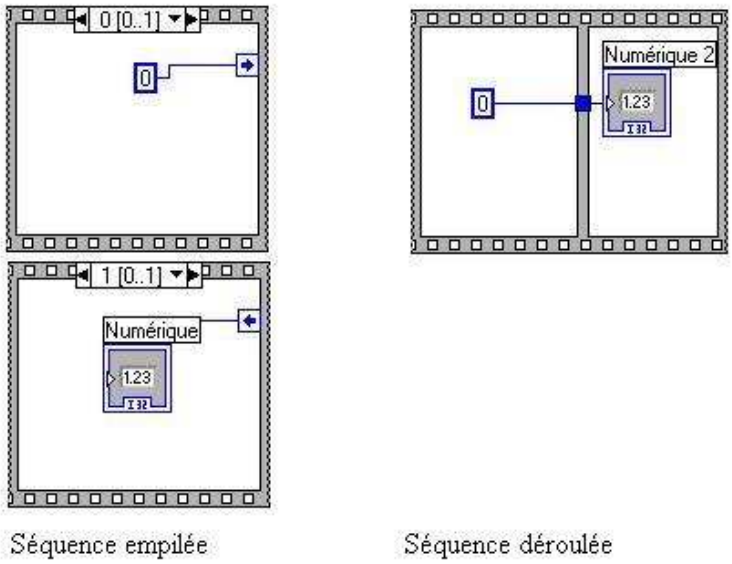


Figure 29 : Types de Séquence

4.5. Structure événement

La structure évènement est une des structures devenue incontournable sous labview. Cette structure permet de gérer **tout évènement utilisateur sur la FA**. Elle attend qu'un évènement utilisateur se produise avant d'exécuter le code approprié ou se désactive après un time out défini. Chaque structure événement possède une file d'attente où seront stockés les événements qui seront traités l'un après l'autre. Sans cette structure votre programme est obligé d'interroger l'état des objets de la Face Avant dans une boucle pour vérifier si un changement a eu lieu, cette interrogation consomme du temps processeur et peut manquer des événements si ceux-ci se produisent trop rapidement.

Elle se présente ainsi :

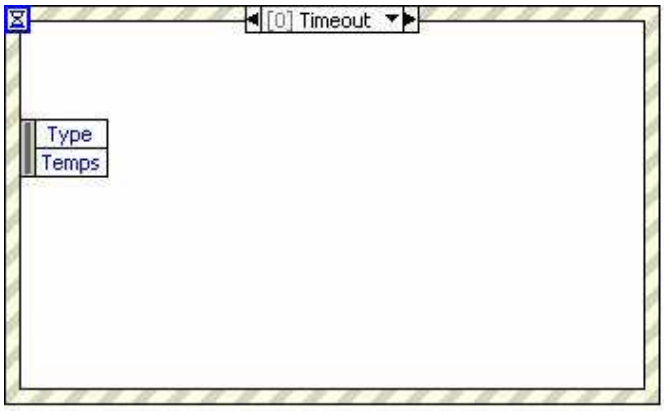


Figure 30 : Structure événement

Le sablier en haut à gauche sert de temps d'attente avant de passer dans l'événement «timeout», si aucun temps n'est imposé la structure attend jusqu'à ce qu'un événement se produise.

Afin d'ajouter des conditions évènement click droit sur la structure – ajouter une condition d'évènement.

Figure 31 : Ajouter une condition

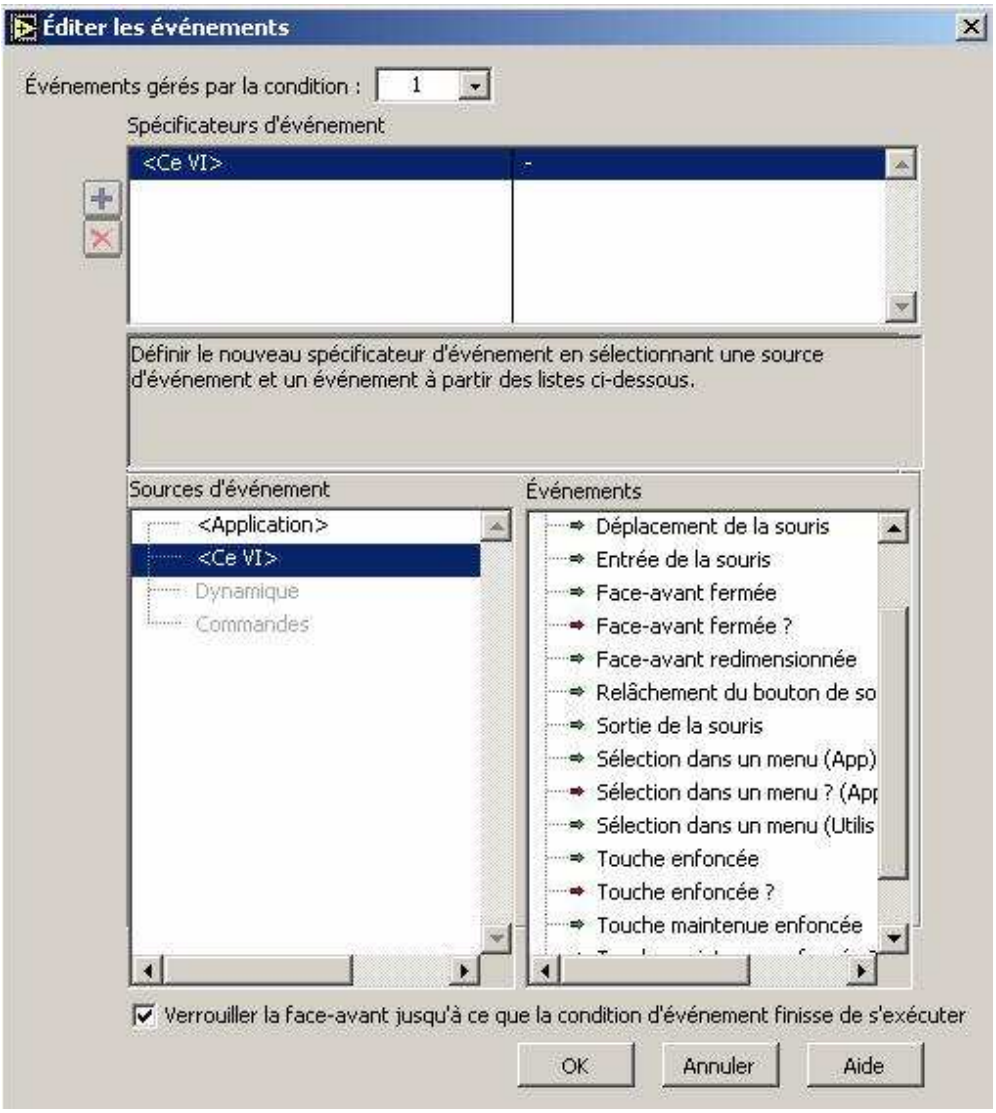


Figure 32 : Editer les événements

Vous voyez que certains événements sont en double, il existe pour certains événement une possibilité de filtrage (flèche rouge et ?) c'est-à-dire que vous avez la possibilité d'annuler l'événement en cours. Vous pouvez ou non verrouiller la Face Avant tant que l'exécution du diagramme de l'événement n'est pas terminée mais dans ce cas certaines commandes ne répondront plus à l'utilisateur.

Chaque structure événement possède une file d'attente, chaque événement sera exécuter l'un après l'autre, si un événement permis a lieu alors qu'un diagramme d'un autre événement est en exécution il sera mis en file d'attente. Il n'est pas possible de supprimer cette file d'attente.

- Exercice : utilisez la structure évènement dans votre programme capteur de sorte que l'utilisateur ferme la FA, une boite de dialogue s'ouvre demandant si nous voulons fermer ou pas la fenêtre. (boite de dialogue dans fonction Temps et Dialogue)

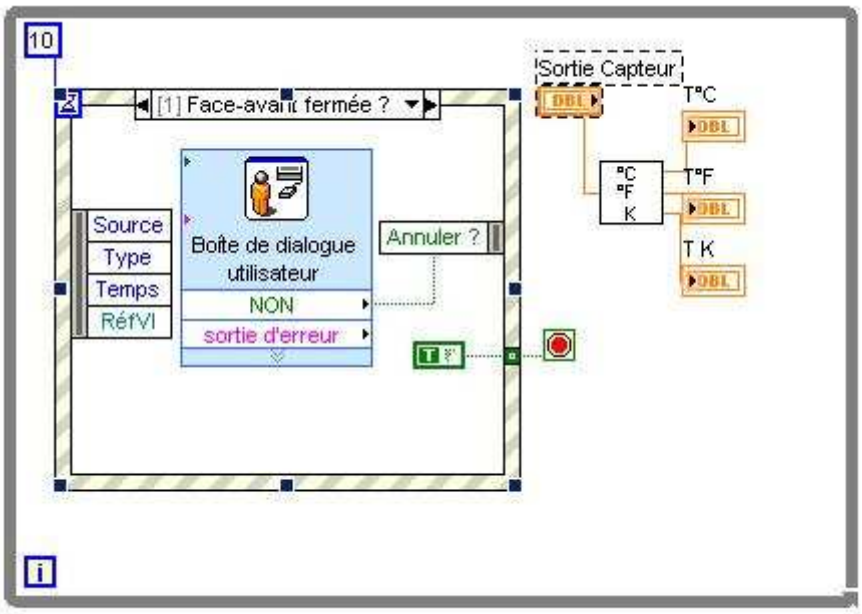


Figure 33 : Utilisation structure événement

Dans l'exercice si l'utilisateur répond NON alors la fenêtre ne se ferme pas.

Il existe plusieurs méthodes d'utilisation des structures événement qui ne seront pas développées ici, mais qui peuvent être vues grâce aux exemples. Il faudra absolument éviter d'utiliser 2 structures évènement dans la même boucle.

Attention :

Si vous verrouillez la Face Avant vous ne pourrez plus utiliser vos commandes etc… tant que le code dans l'événement ne sera pas fini, seules quelques fonctions de Face Avant subsisteront.

Si vous ne verrouillez pas Face Avant alors des événements pourront être mis en queue et s'exécuteront un par un une fois le code de chaque événement exécuté.

Cette structure événement peut gérer des événements dynamiques, les événements dynamiques vous laissent maître de décider quels événements seront générés, vous pouvez aussi modifier la génération d'événements ou supprimer des enregistrement d'événement. (Se référer à l'exemple «Enregistrement d'événement dynamique»).

Il est possible aussi de générer des événements utilisateur par programmation. (Se référer à l'exemple «Programmatically Fire Events»).

5. LES GRAPHES

Il existe plusieurs types de graphes : déroulant, XY, simple, déroulant d'intensité, d'intensité, numérique et 3D. Tous ces graphes ont des particularités différentes mais sont tous paramétrables par un click droit (notamment la modification des axes, l'ajout d'échelle, modification de couleur ….).

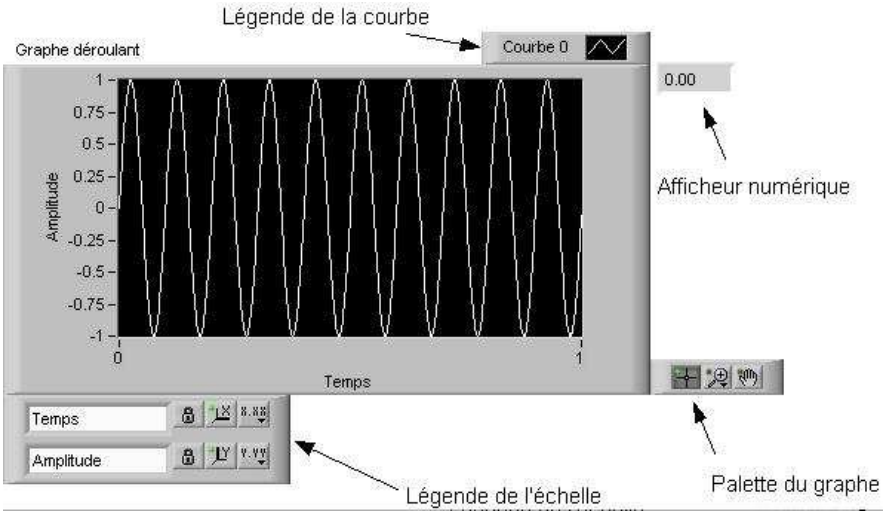


Figure 34 : Graphe et légende

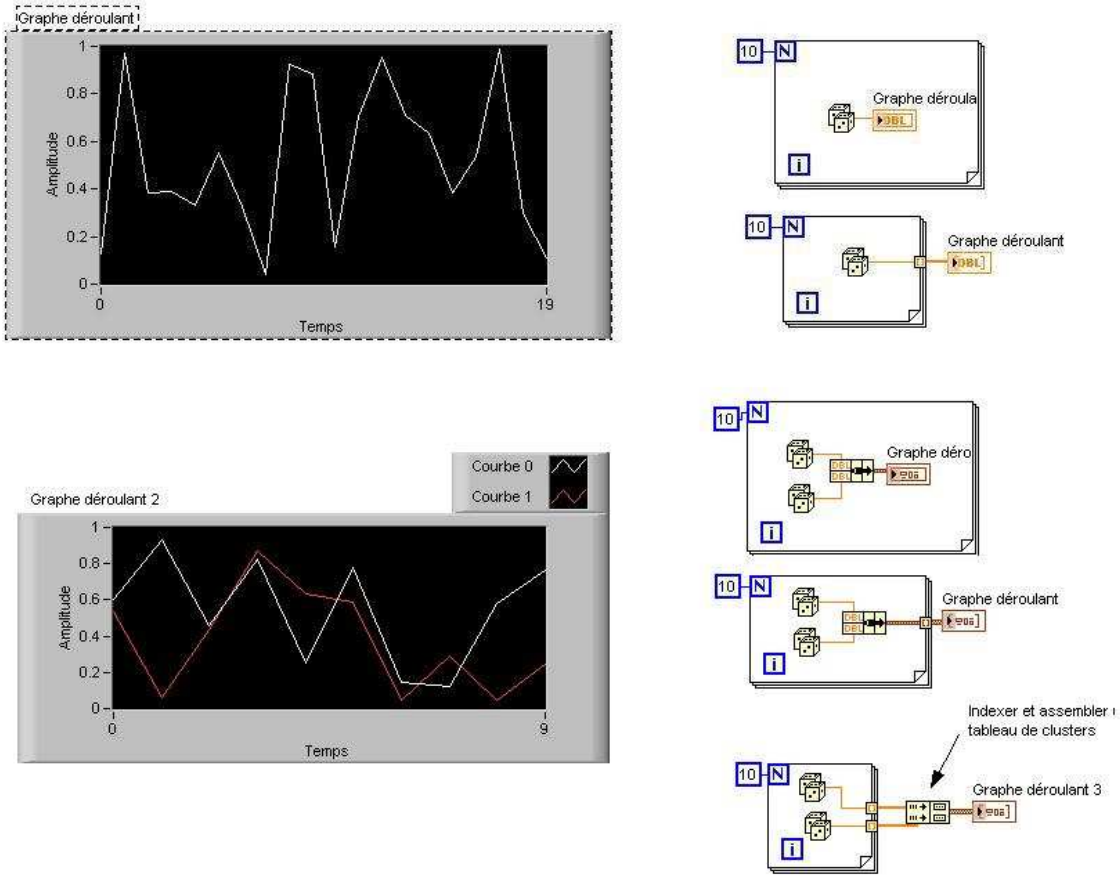


Figure 35 : Graphe déroulant

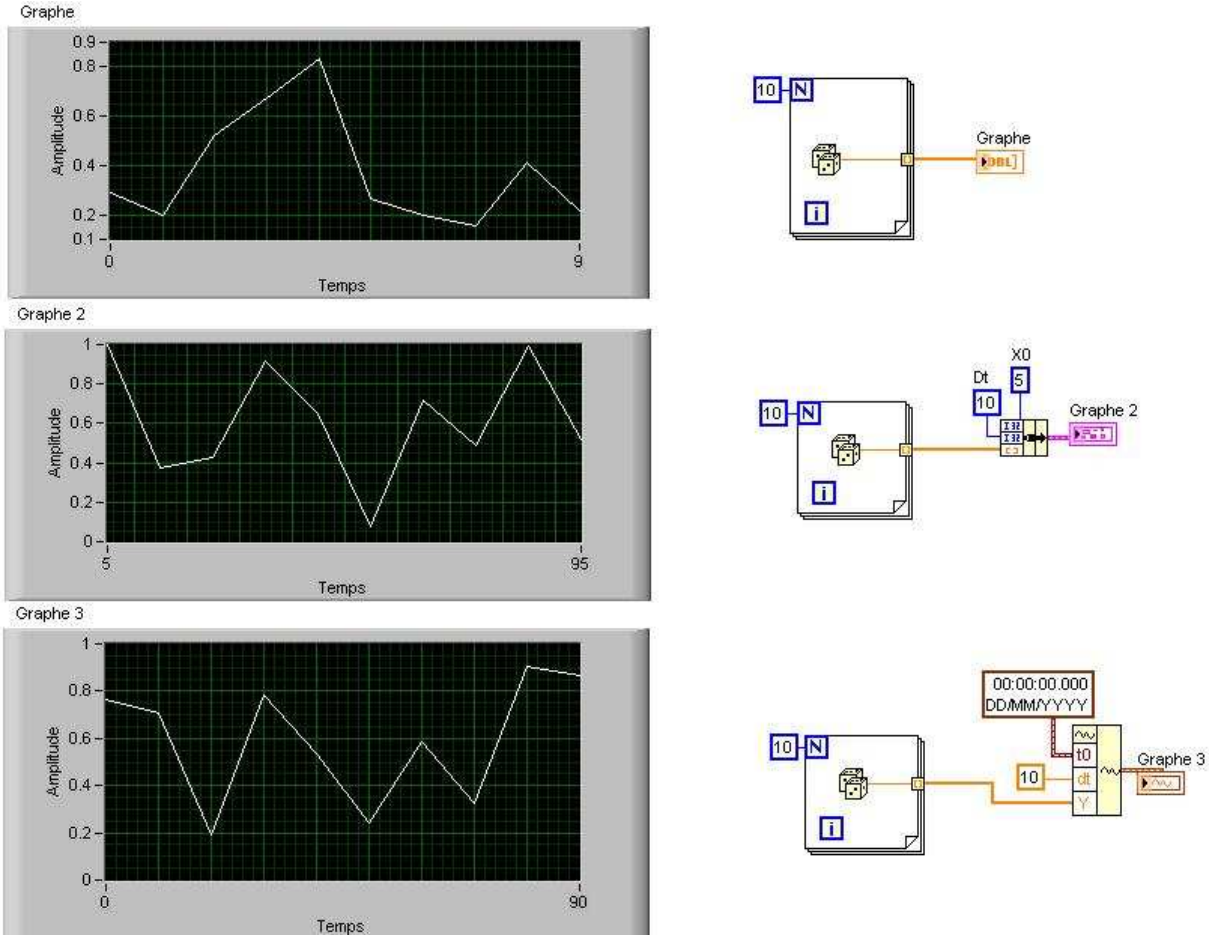


Figure 36 : Graphe simple 1 courbe

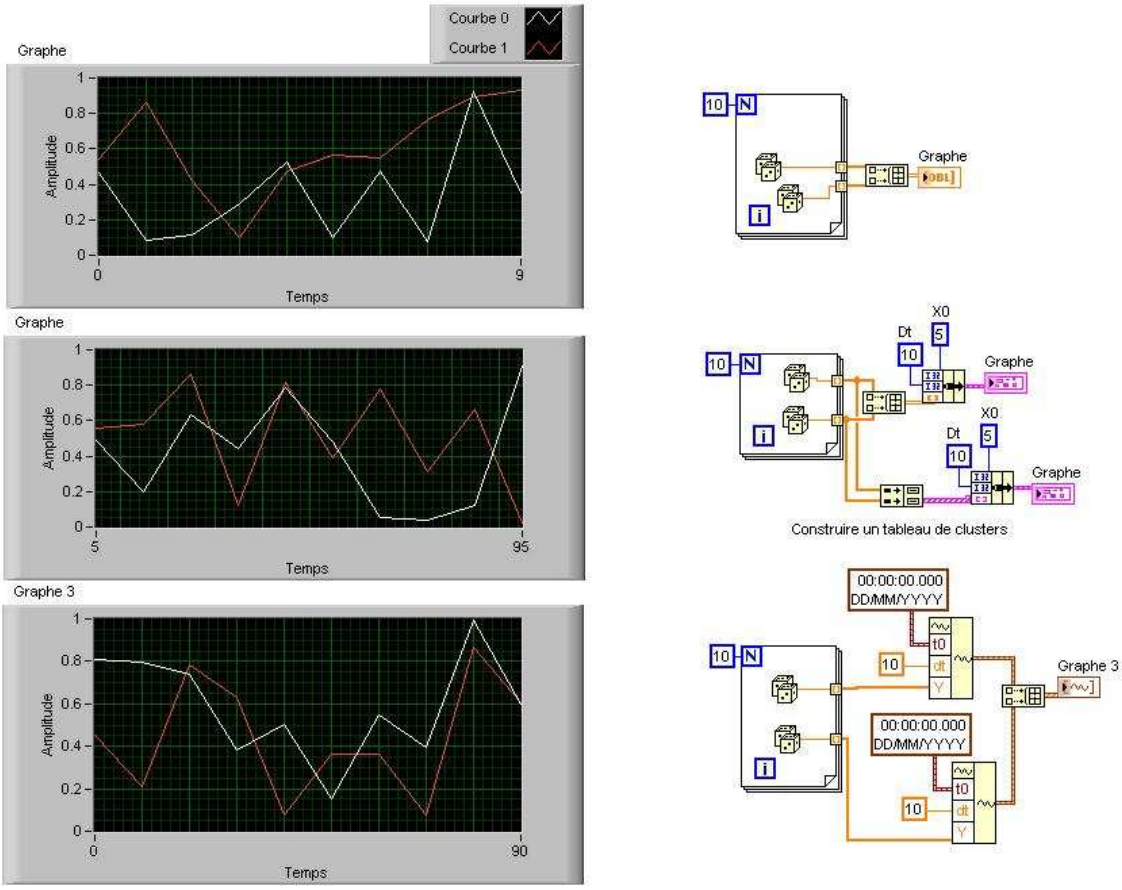


Figure 37 : Graphe simple 2 courbes

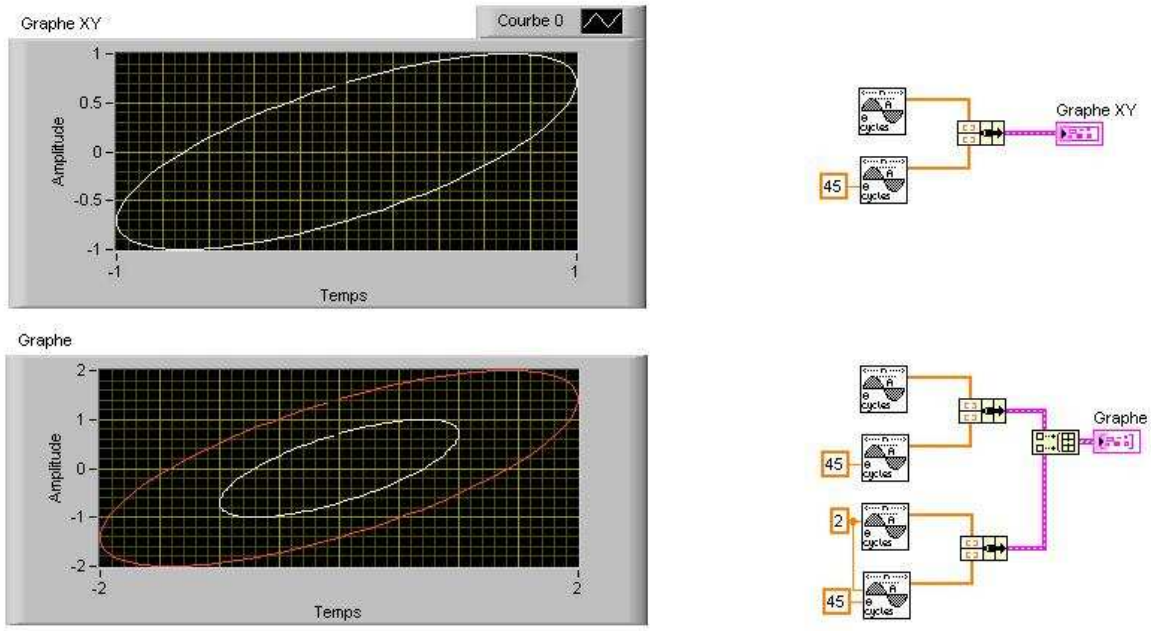


Figure 38 : Graphe XY

Exercice : Ajoutez un graphe déroulant à l'intérieur de la boucle afin de visualiser les 3 températures. Puis ajoutez des échelles pour les 3 courbes; après la sortie de boucle affichez un graphe XY de l'évolution de la température en fonction du temps

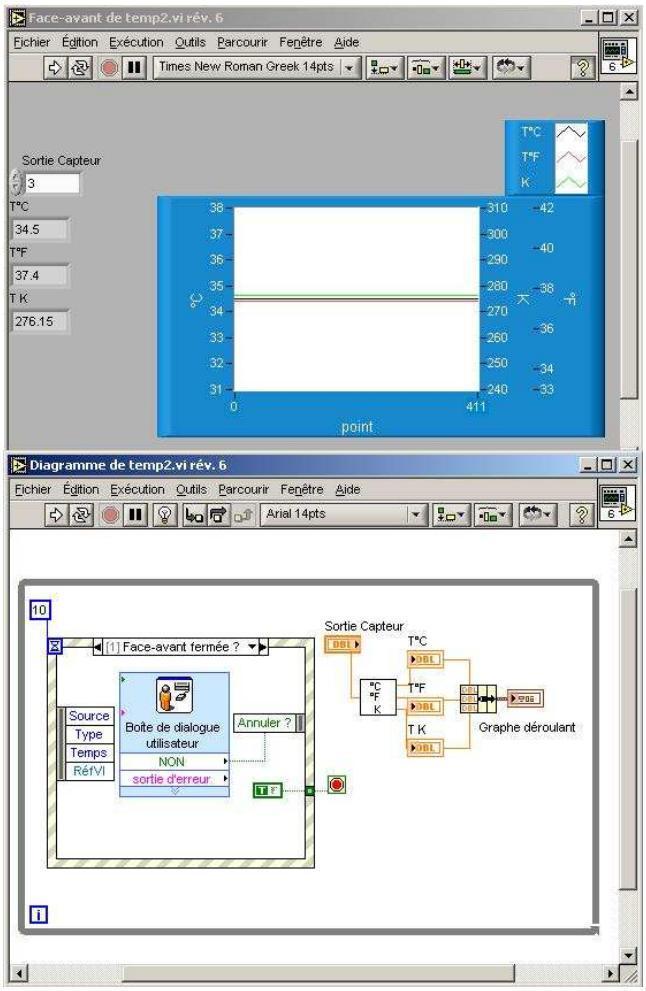


Figure 39 : Insertion graphe déroulant

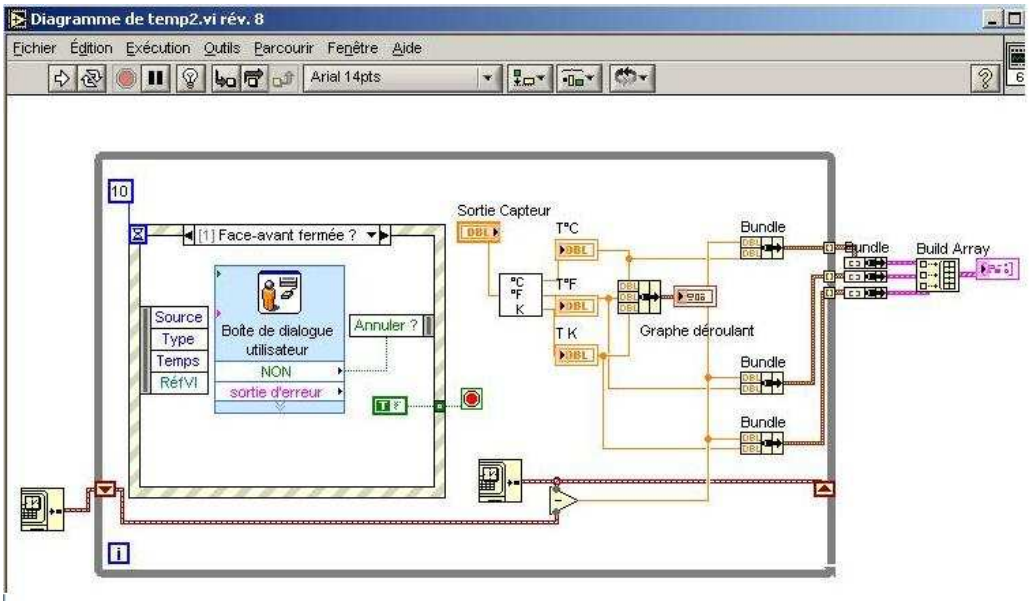


Figure 40 : Graphe XY en sortie de boucle

6. LES COMMANDES TABLEAUX, CHAINE DE CARACTERES, CLUSTERS.

6.1. Tableaux

Les tableaux sont un groupement ordonné d'éléments de même type, le type peut-être booléen, chaîne, waveform, cluster … Ils sont de une à plusieurs dimensions et jusqu'à 2^31-1 éléments par dimension. (L'indexation commence à 0 et se termine à N-1, N étant le nombre d'élément.)

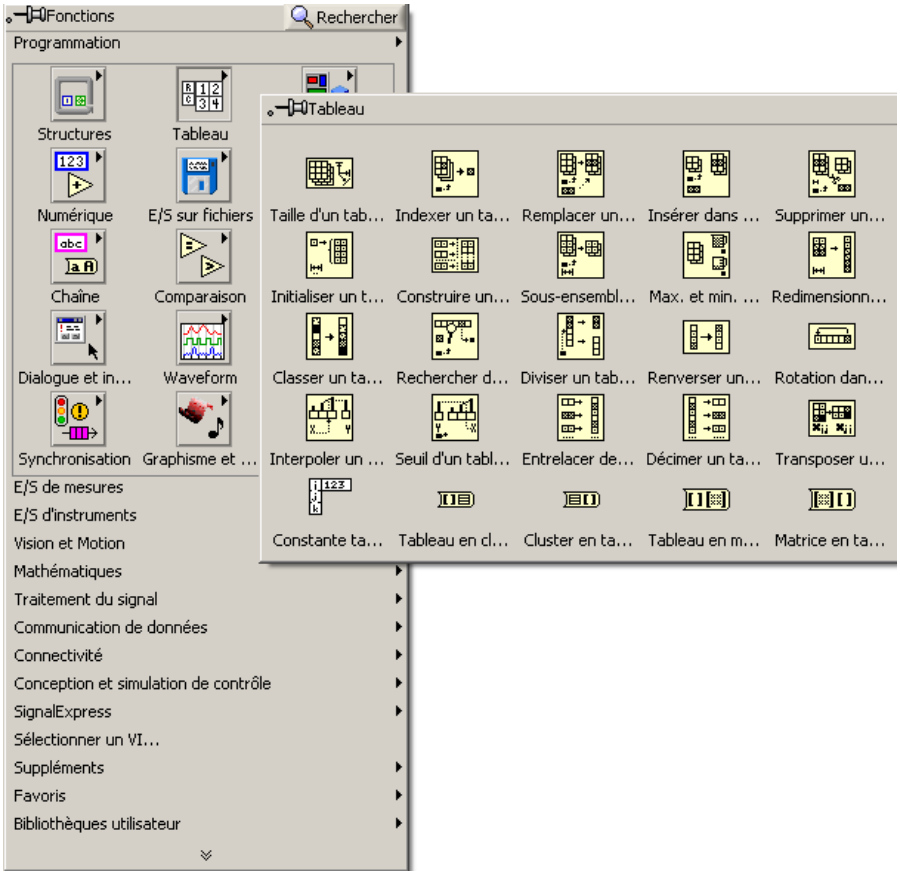


Figure 41 : Palette Tableau

Une panoplie de fonctions sur les tableaux existe, de plus des calculs simples sont possibles sur les tableaux car les fonctions sont polymorphiques. Si par exemple vous voulez sommer 2 tableaux vous pouvez sommer tous les éléments par un scalaire, ou sommer tous les éléments d'un tableau par tous les éléments d'un autre tableau.

Vous avez pu remarquer que nous avons déjà utilisé une fonction importante des tableaux : construire un tableau.

6.2. Chaîne de caractères

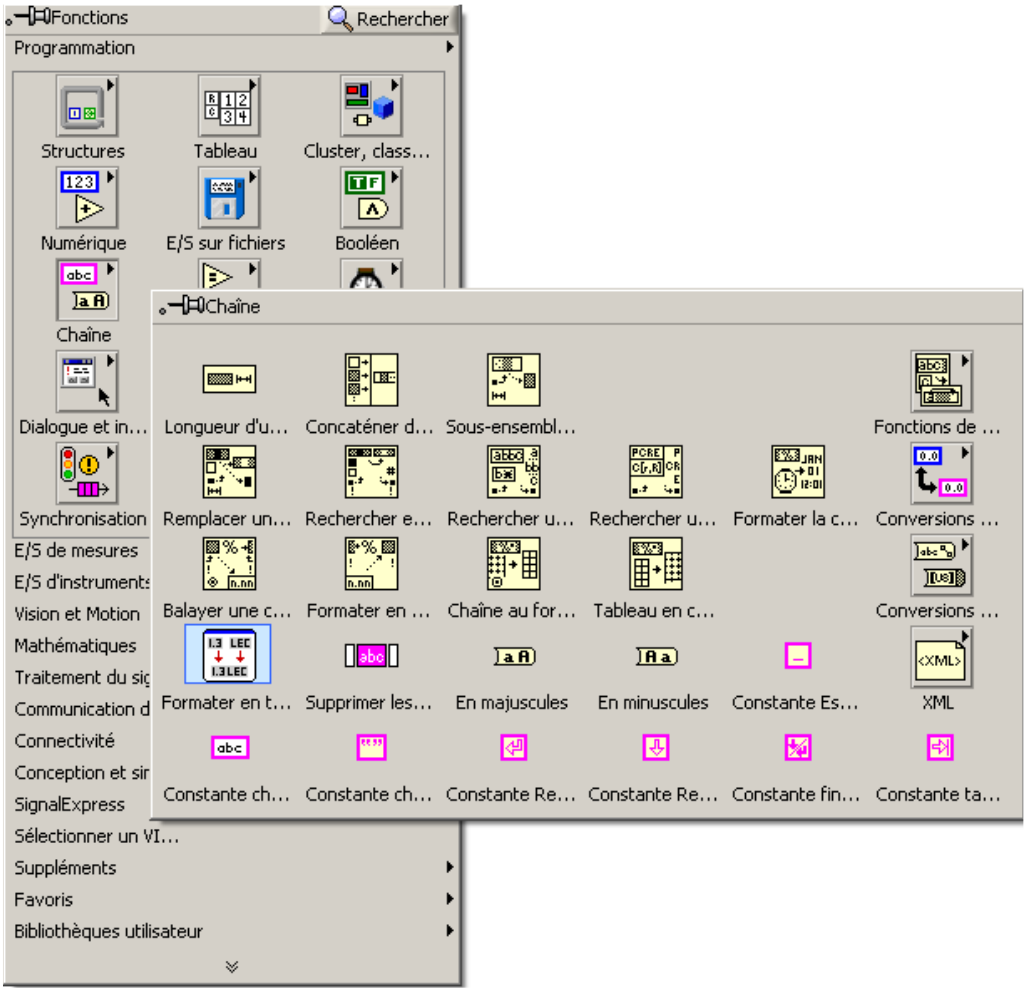


Figure 42 : Palette de Chaîne

Comme le nom l'indique ce sont des commandes ou des indicateurs où vous retrouvez une chaîne de caractères. Une des particularités est que nous pouvons afficher la chaîne d'une manière normale, ou bien avec ses codes (\s espace, \n retour à la ligne…), style mot de passe, et en hexadécimal, tout ceci trouve son utilité dans la gestion de commande d'instruments, des «login» avec mot de passe etc…

De plus encore une fois il existe une panoplie de fonctions sur les chaînes de caractères que vous retrouverez souvent dans les Vis de commandes d'instruments.

Certains exemples sur les chaînes vous montrent comment utiliser les fonctions principales.

6.3. Clusters

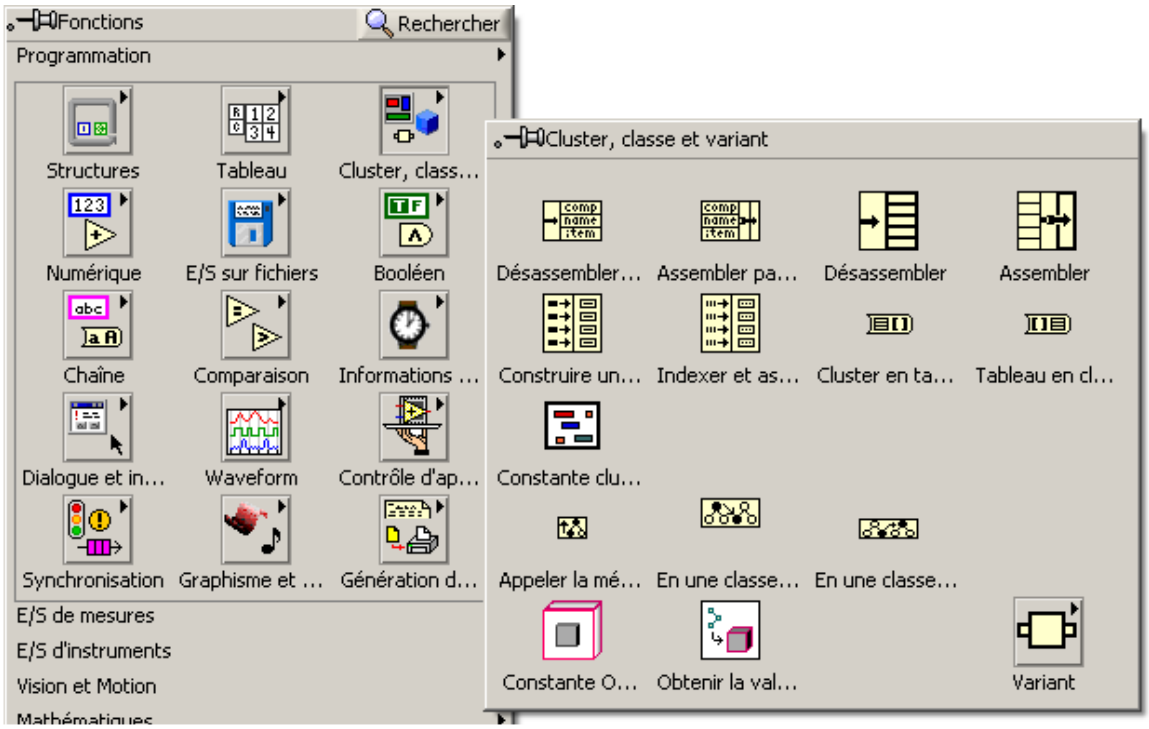


Figure 43 : Palette Clusters

Les clusters peuvent être comparés aux ensembles en mathématiques (record en Pascal, et struct en C), ce sont des ensembles pouvant englober tout type de donnée, c'est-à-dire des valeurs numériques, booléennes, chaîne de caractères mais aussi des tableaux, des graphes …

Ainsi donc les clusters seront très utiles pour transférer de nombreuses données d'une partie du code à une autre ou d'un VI à un autre.

Des fonctions existent sur les clusters notamment pour les créer et pour désassembler les données du cluster.

7. ENTREE – SORTIE SUR FICHIER.

Une panoplie de fonctions plus ou moins évoluées est à votre disposition pour les entrées sorties sur fichiers, aussi bien des fichiers de types ASCII que binaire.

Attention aux fonctions de bas niveau qui regroupe à l'intérieur des fonctions avancées.

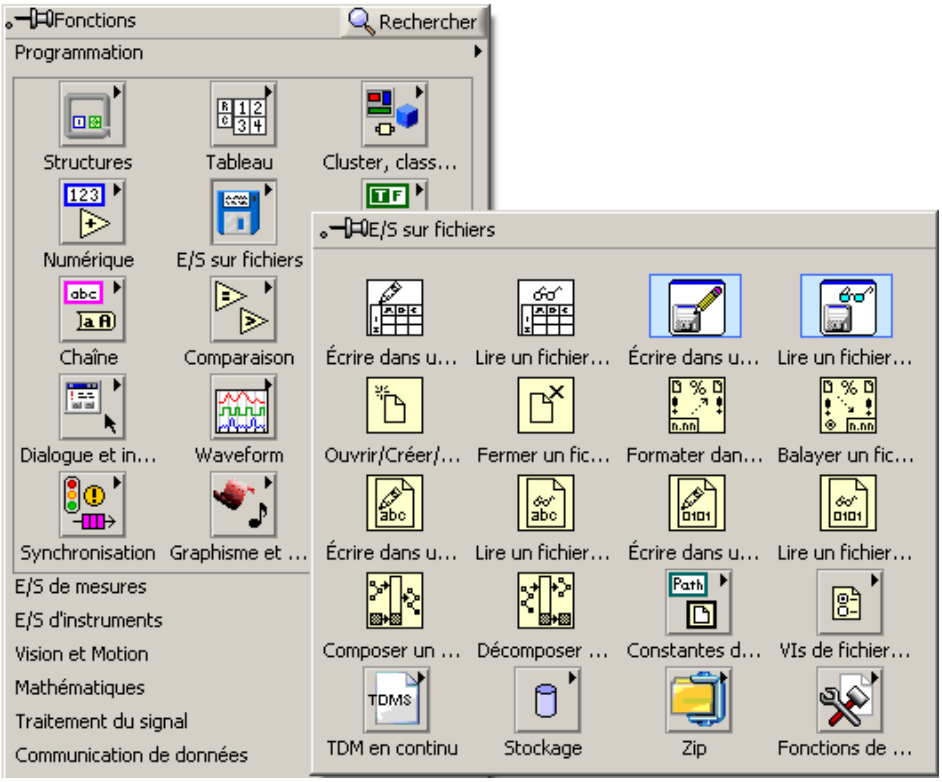
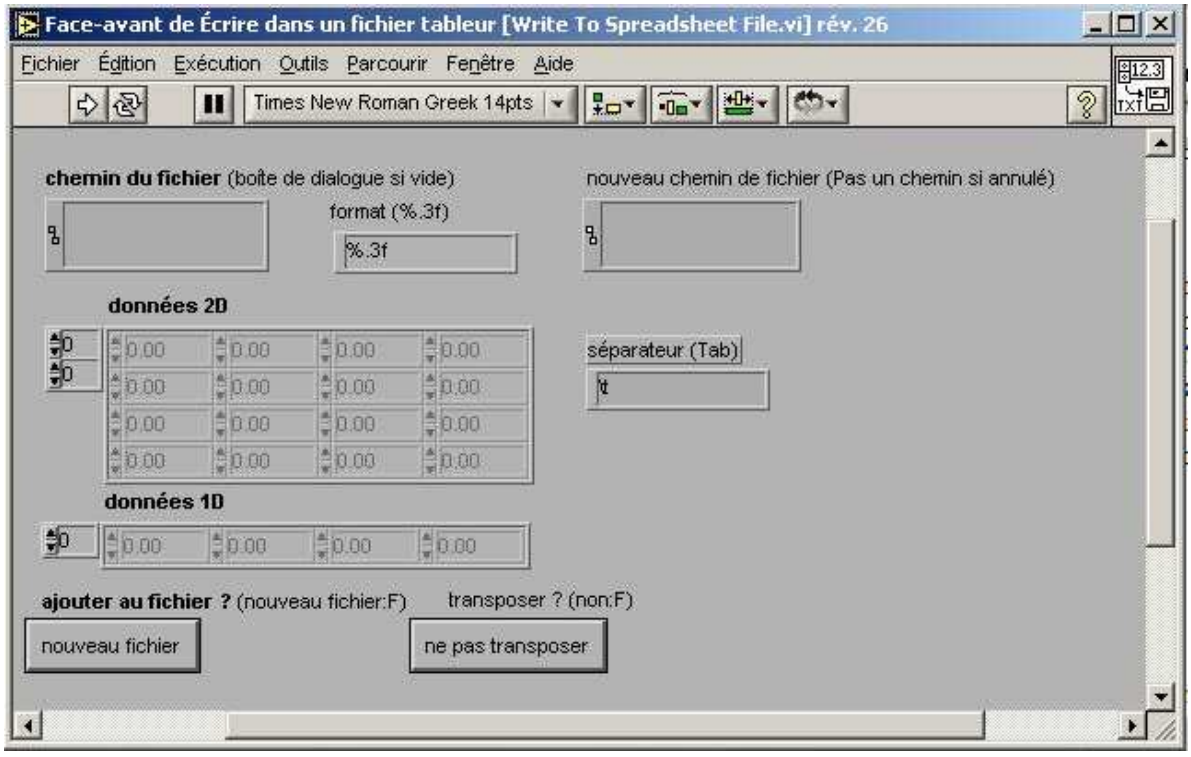
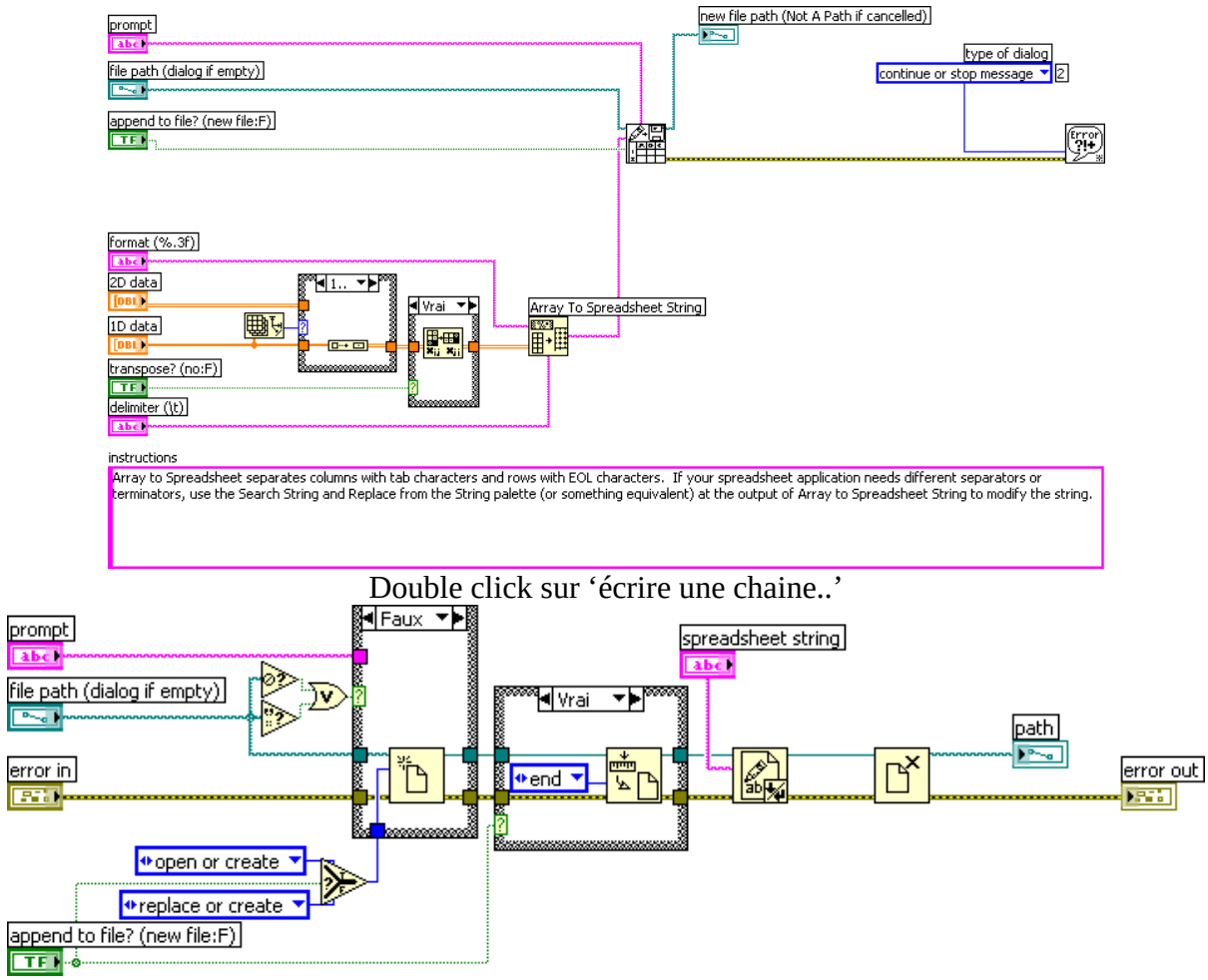


Figure 44 : Palette Entrée Sortie sur Fichier

Ouvrir la fonction «Ecrire dans un fichier tableur» :





Double click sur ‘écrire une chaine..’

Figure 45 : Ecrire dans un Fichier Tableur

Vous observez que cette fonction dans un premier temps ouvre le fichier puis écrit et enfin le ferme. Elle est très pratique pour écrire des données en une seule fois mais s’avère contraignante si elle est insérée par exemple dans une boucle.

Pourquoi ? Le fait d’ouvrir et de fermer un fichier de chaînes de caractères peut être plus ou moins long suivant la taille de ce fichier, aussi on préfèrera dans une boucle utiliser les fonctions avancées, on ouvrira le fichier avant la boucle et le fermera à la fin de la boucle.
Si la vitesse ainsi que la taille du fichier sont primordiales, nous utiliserons les fichiers binaires.

8. VARIABLES LOCALES, GLOBALES, NŒUDS DE PRORIETES, METHODE.

8.2. Variables locales et globales

Un objet de face avant possède qu’un seul terminal au niveau du code mais vous auriez besoin de lire ou de mettre à jour cet objet en différents endroits du code. En utilisant les **variables locales** vous pouvez avoir accès à cet objet en **différents endroits du code et transmettre des données entre différentes structures qui ne peuvent pas être connectées par un câble**.

Placer sur le diagramme 2 boucles «While» qui stopperont à l’appui de la même commande Stop.

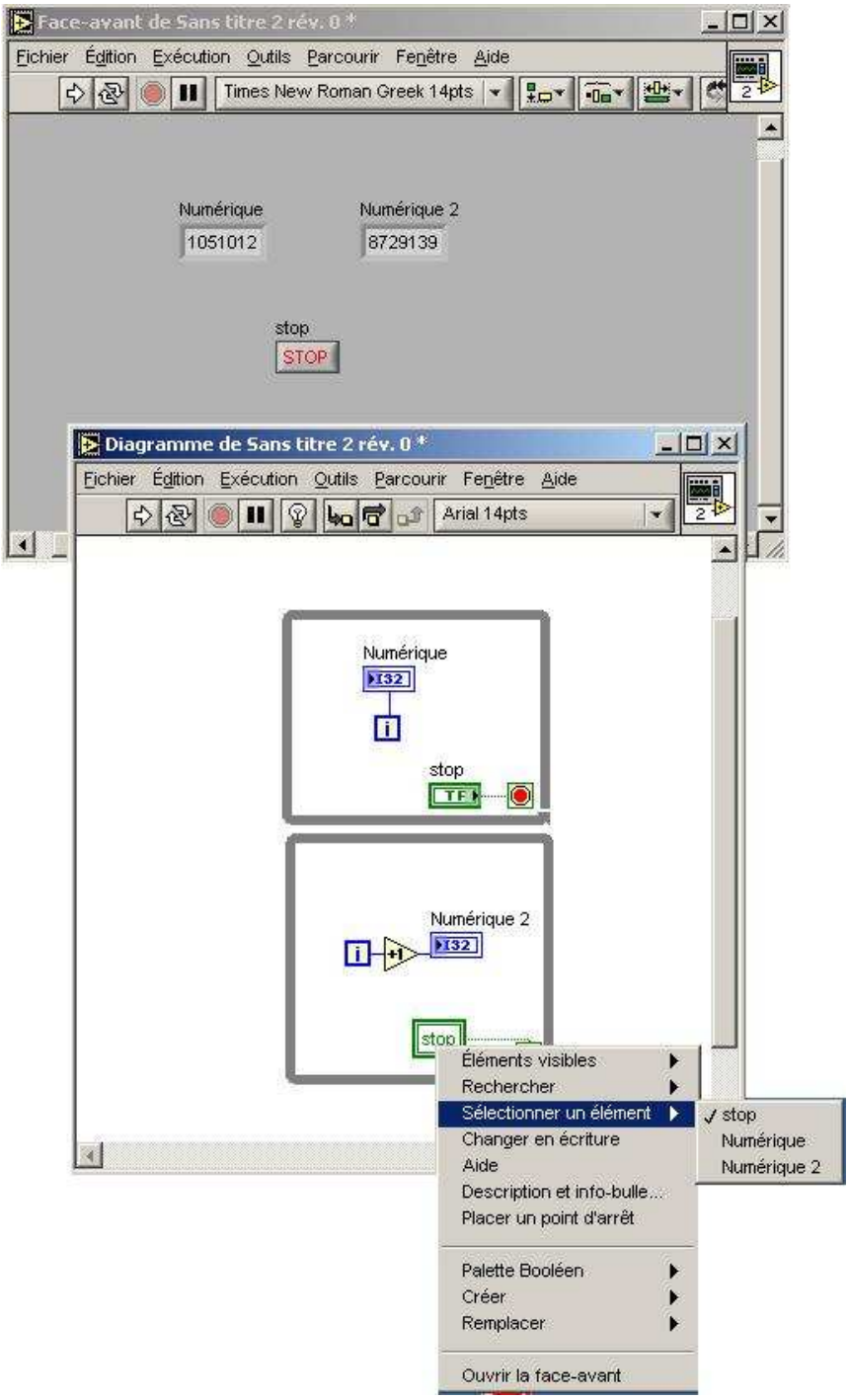


Figure 46 : Variable Locale

Variable locale : fonction structure - variable locale ou click droit sur la commande puis créer variable locale (diagramme).
Attention à la commande STOP, **en variable locale certaines actions mécaniques ne sont pas possibles**. Ici dans ce VI un seul bouton nous permet de gérer 2 séquences. Toute action qui retourne à la position initiale ne peut être utilisée en utilisant les variables locales.

Une variable globale est une variable pouvant être utilisée simultanément par plusieurs Vis. Je ne parlerai pas des variables globales sous labview car celles-ci peuvent amener des bugs difficiles à discerner. Il faut donc préférer utiliser des variables globales VI.
Pour réaliser une variable globale par programmation, nous allons utiliser le registre à décalage non initialisé, ce qui permet de récupérer la valeur mise en mémoire dans le registre (ce sera la lecture de la variable), pour l'écriture il faudra mettre dans le registre la nouvelle valeur.

Réaliser le code suivant :

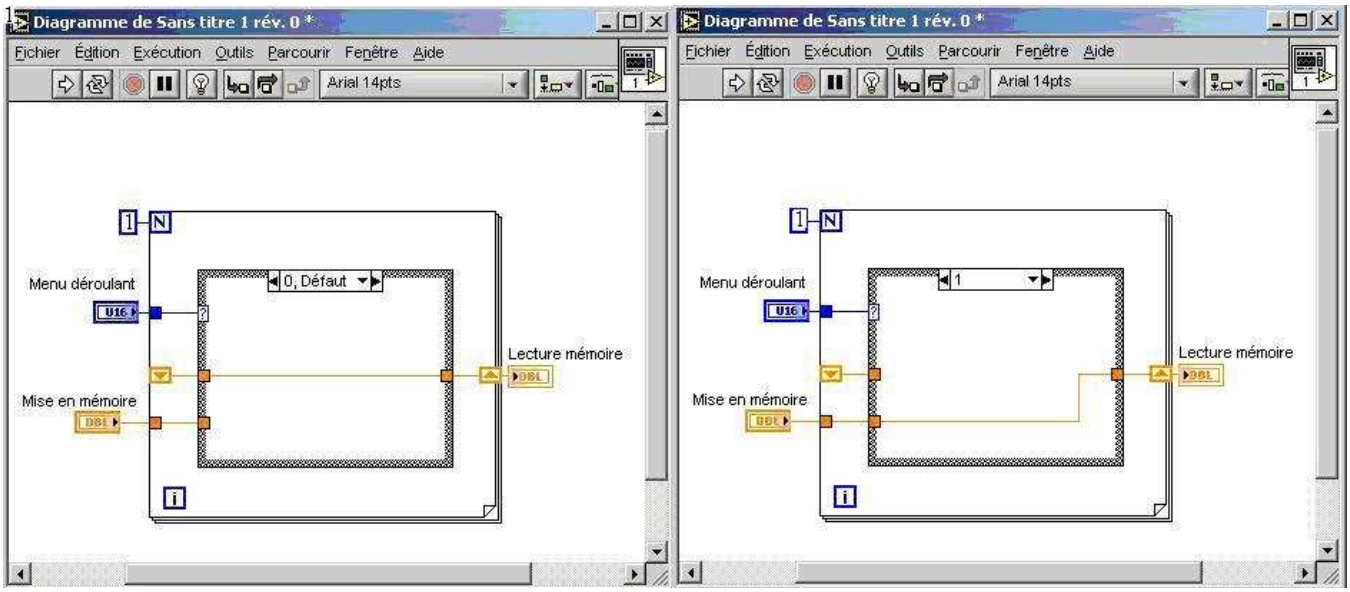
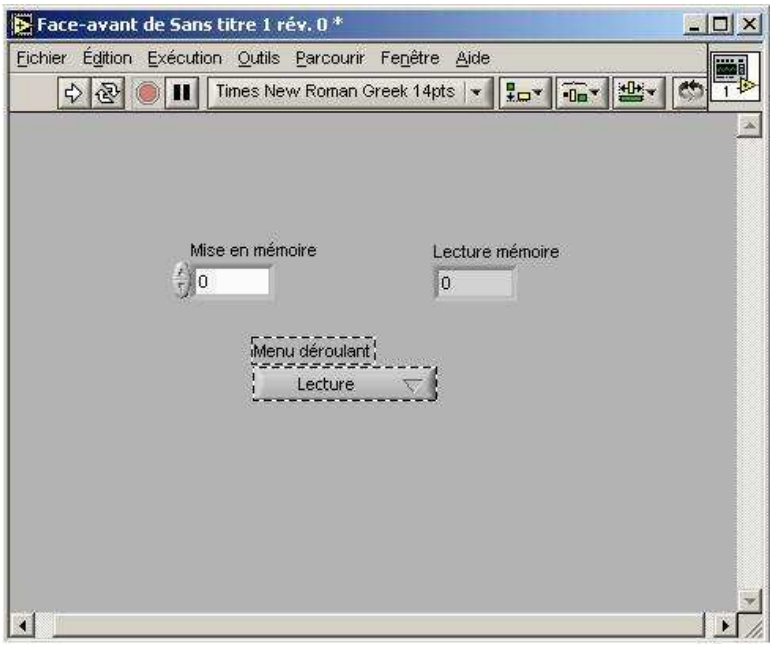


Figure 47 : Variable Globale



Transformez ce VI en sous - VI, vous venez d'écrire une variable globale par programmation, vous pouvez soit écrire une valeur soit la lire. Essayez d'utiliser cette variable globale dans un VI. Par cette méthode la mémoire utilisée lors de l'écriture ou la lecture de cette variable est immédiatement libérée après avoir été utilisée. Bien sûr cette méthode est plus longue à mettre en œuvre mais le gain de mémoire peut être très important.

8.3. Nœuds de propriété et de méthode

Le nœud de propriété permet de modifier l'apparence d'un objet de la FA. Par exemple changer la couleur d'un indicateur, faire clignoter un autre … On peut créer un nœud de propriété à partir du terminal de la commande en faisant click droit – créer nœud de propriété, sur ce nœud créé vous pouvez choisir la ou les propriétés à lire ou écrire toujours à l'aide du click droit.

- Exercice : prendre une commande réservoir et placer la en FA, une commande LED (appelez-la visible - invisible) et une commande Boite de couleur (palette numérique).

Faites un code qui permette de rendre invisible ou visible le réservoir et de changer la couleur de remplissage.

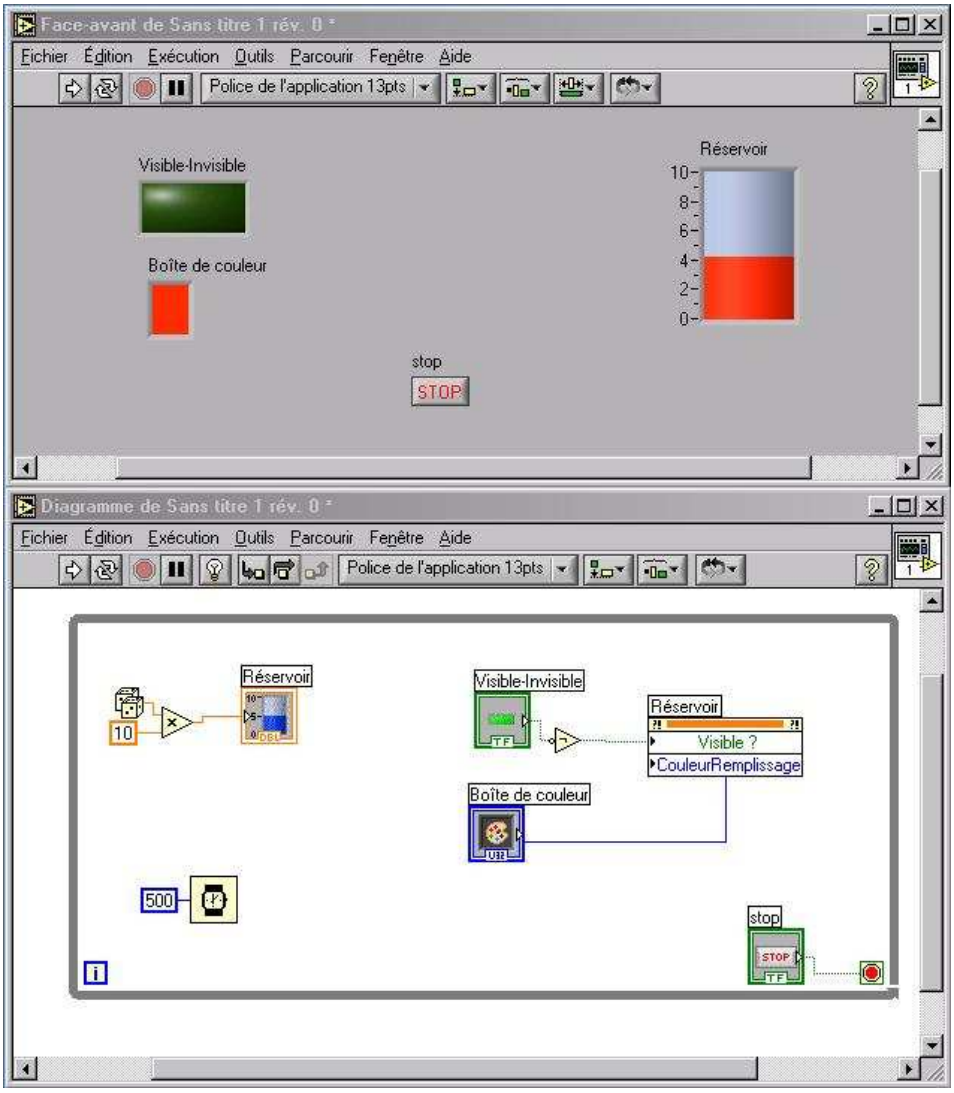


Figure 48 : nœuds de propriétés

Suivant les commandes les propriétés sont différentes.

Le nœud de méthode permet de la même manière d'avoir accès à des «méthodes», fort utile pour par exemple, appeler à exécuter un VI par le code d'un autre VI etc …

Dans ce programme intégrez un nœud de méthode pour visualiser l'image du réservoir dans une boîte image.

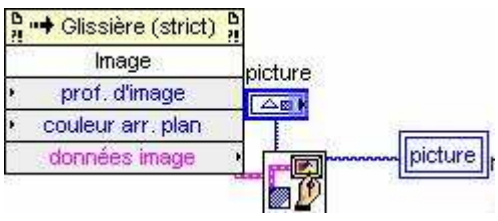


Figure 49 : Nœud de méthodes

9. COMMUNICATION INSTRUMENTS – VISA série,GPIB ...

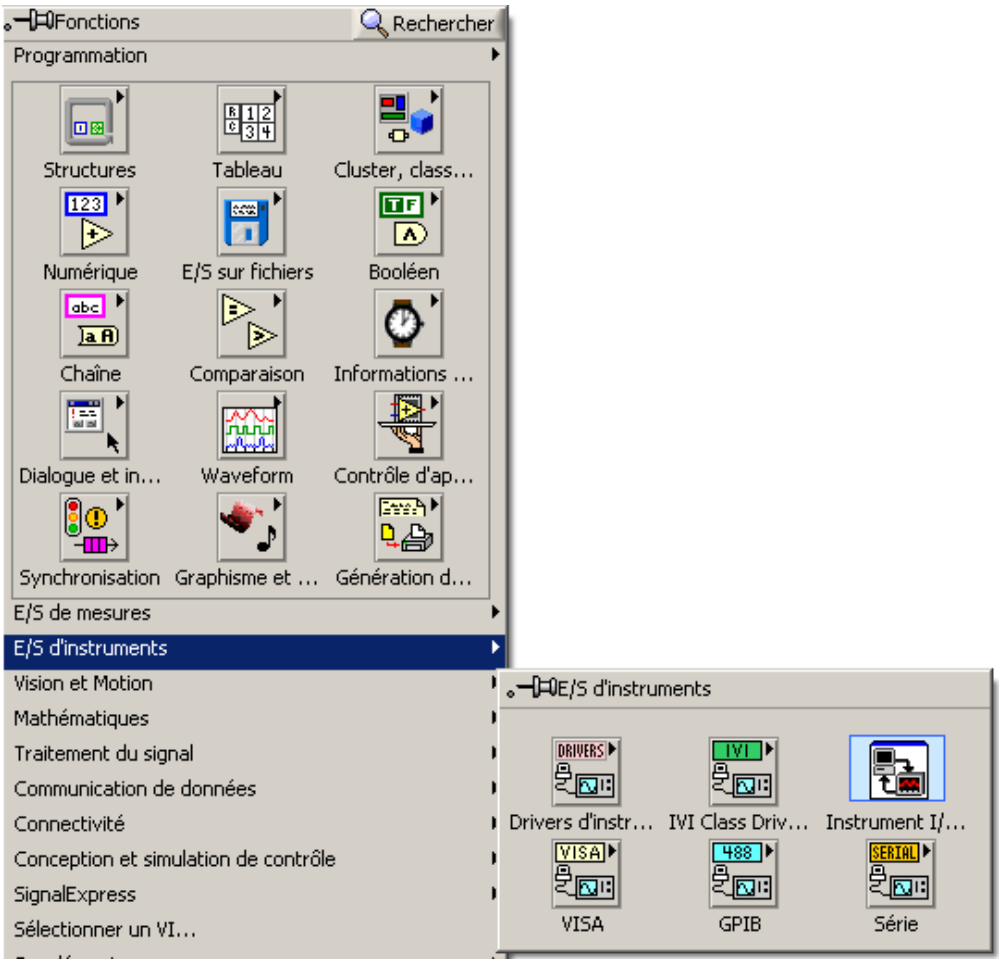


Figure 50 : Palette E/S Instruments

Nous utiliserons pour les communications avec les instruments principalement les fonctions VISA (Visual Instrument Software Architecture) utilisées sous labview pour les drivers d'instruments. Les fonctions VISA vont vous permettre de communiquer par des liaisons série, GPIB, VXI ou PXI dont les drivers sont écrits pour différents types de communication.

Figure 51 : VISA open

Pour bien comprendre les fonctions VISA, 3 termes sont à connaître : la **ressource** qui est n'importe quel instrument de votre système (incluant les ports série et parallèle), la **session** à une ressource qui permet la communication avec celle-ci ou encore le canal de communication (vous êtes toujours obligés d'ouvrir une session qui retournera un numéro de session correspondant à l'appareil avec lequel vous communiquez), la **description** de l'instrument qui est le nom de votre ressource et qui spécifie le type d'interface, l'adresse et le type de session VISA.
Le nom de ressource peut être : COM1,COM2…, ou ASRL0 ::INSTR…, GPIB0 ::adresse…

Il est assez aisé de programmer rapidement des VIs pour communiquer avec vos instruments car il existe de nombreux drivers déjà réalisés, vous pouvez créer vos propres drivers, par contre configurez toujours correctement vos interfaces, fermer vos sessions VISA quand elles ne sont plus utilisées.

Pour exemple ce VI qui demande l'identité de votre instrument. (Attention ici sur le port série 1 du PC) L'ordinateur possède une carte GPIB (IEEE). Le menu déroulant permet de choisir rapidement l'interface si MAX (Measurement & Automation Explorer a été rafraîchi).

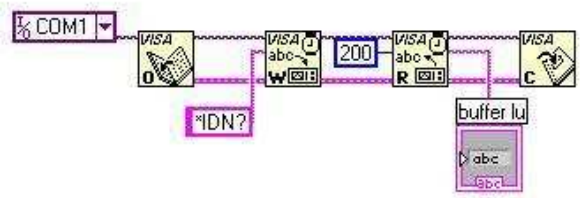


Figure 52 : VISA Ecriture / Lecture

Exercice : ouvrir l'exemple «HP34401 Getting started» ou «HP33120A Getting started» suivant votre matériel

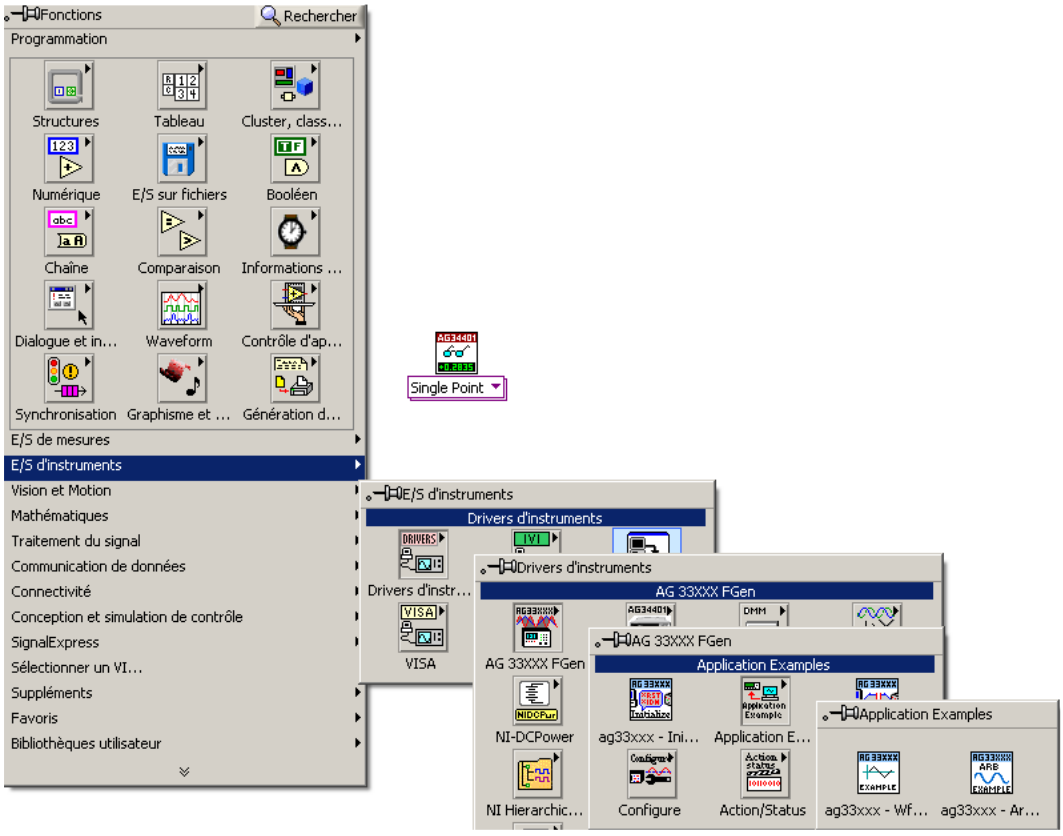


Figure 53 : Drivers d'Instrument

Observer comment se fait l'initialisation et la récupération des données. Vous retrouvez ici les fonctions des chaînes de caractères. Utiliser les VIs adéquats pour visualiser une mesure simple DC sur un graphe déroulant, marche/arrêt à l'appui d'un bouton.

Vous observez que pour un driver d'instrument comme celui-ci de nombreuses fonctions existent qui ne sont pas forcément utiles pour des mesures simples, à vous d'utiliser les fonctions adéquates ou d'écrire directement votre VI avec les commandes ASCII.

10. ACQUISITION ANALOGIQUE DE DONNEES DAQmx

10.1. Introduction

L'acquisition de données qu'elle soit de type analogique, numérique, comptage nécessite de connaître quelques points importants qui permettent de faire le bon choix de carte car de nombreuses cartes existent ayant des particularités différentes.

Vous aurez besoin de connaître : la gamme de mesure et d'amplification analogique, vitesse d'acquisition, de génération, horloge des compteurs, le nombre de voies, la résolution.

Par exemple un des points importants est le choix de la résolution et l'amplification de votre carte qui dépend du plus petit changement de signal que vous voulez mesurer. (appelé Code Width CW)

$$CW = \frac{\text{voltage range}}{\text{amplification} \times 2^{\text{resolution in bits}}}$$

- Exercice : nous voulons mesurer la température d'un bassin d'eau avec un thermocouple, celui-ci mesure une température comprise entre -270°C et 1372°C pour une différence de potentiel de -6.548 à 54.874mV. Nous désirons mesurer un changement de 2.1µV, quelle est la meilleure configuration parmi les suivantes ? On estime que la température de l'eau ne sera jamais inférieure à 0°C.

Config 1 : résolution 12bits, gamme 0 - 10V
Config 2 : résolution 12 bits, gamme -10 – 10V
Config 3 : résolution 16bits, gamme -10 – 10V
Config 4 : résolution 16 bits, gamme 0 – 10V

Amplifications disponibles 10 – 20 – 50 – 100

Quand la carte est choisie, il sera nécessaire de connaître certaines règles de connexion qui sont les suivantes :

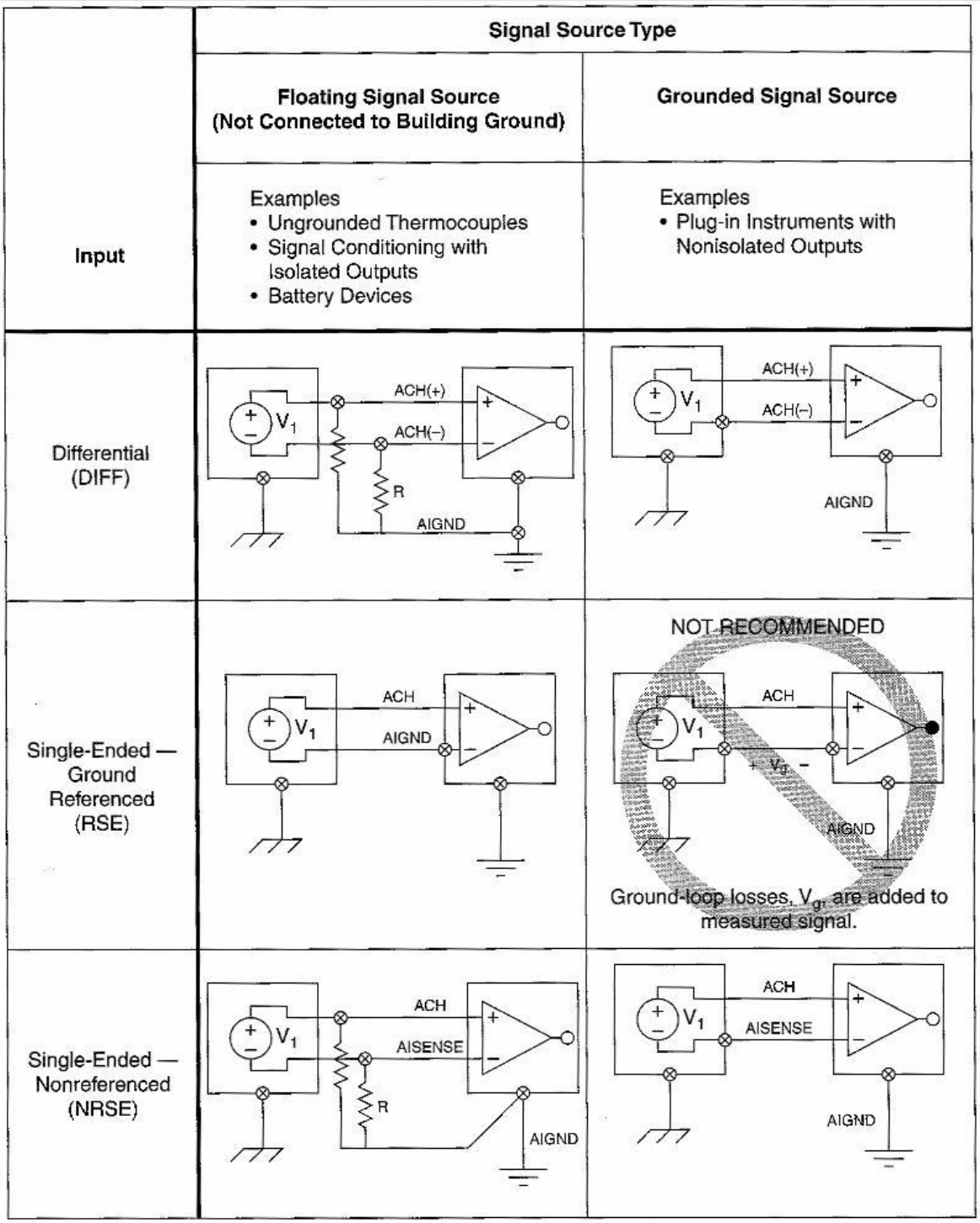


Figure 54 : Connexion carte

10.2. Driver d'instruments: software Measurement and Automation eXplorer

L'acquisition par carte NI ne peut se faire que s'il existe des drivers, qui sont gérés par un software MAX. MAX contient 2 types de drivers DAQ : les traditionnels (anciennes versions, NI-DAQ) et les nouvelles mx (NI-DAQmx). Attention encore aux choix des cartes car certaines ne fonctionnent qu'avec les nouveaux drivers alors que d'anciennes cartes ne fonctionnent qu'avec les anciens drivers.

Le principal changement est que les fonctions sont les mêmes pour tout type d'acquisition (analogique, numérique..) et nécessite un paramétrage, pour l'ancienne version des fonctions pour chaque type sont disponibles.

MAX inclut les catégories de fonctions suivantes :

- Voisinage de données
- Interfaces et périphériques
- IVI instruments
- Echelles
- Logiciels

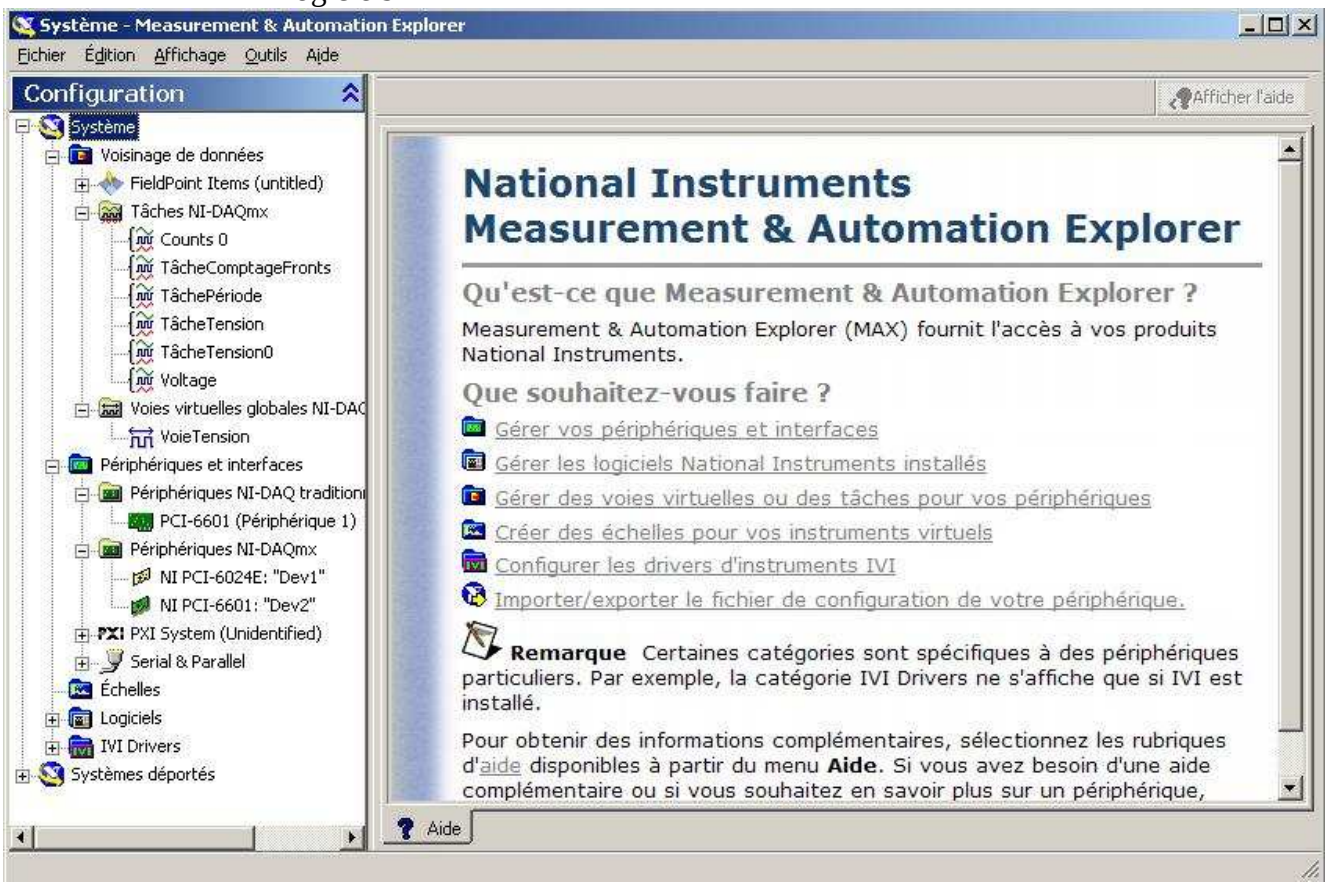


Figure 55 : Measurment & Automation Explorer

Sous MAX vous pouvez créer :

- une voie virtuelle contenant la configuration de voie que vous aurez choisie comme la gamme de mesures, donner un nom spécifique à la voie créée pour pouvoir y accéder sous labview ou avec une tâche.
- une tâche qui est une collection de voie avec le même type de trigger et de base de temps. Elle représente la mesure ou la génération que vous voulez réaliser.

▪ Exercice : Ouvrir MAX, cliquer pour développer les différents menus, observer les différents périphériques à votre disposition. Créer une voie virtuelle globale en cliquant droit sur Voisinage de donnée – créer un nouvel objet - …

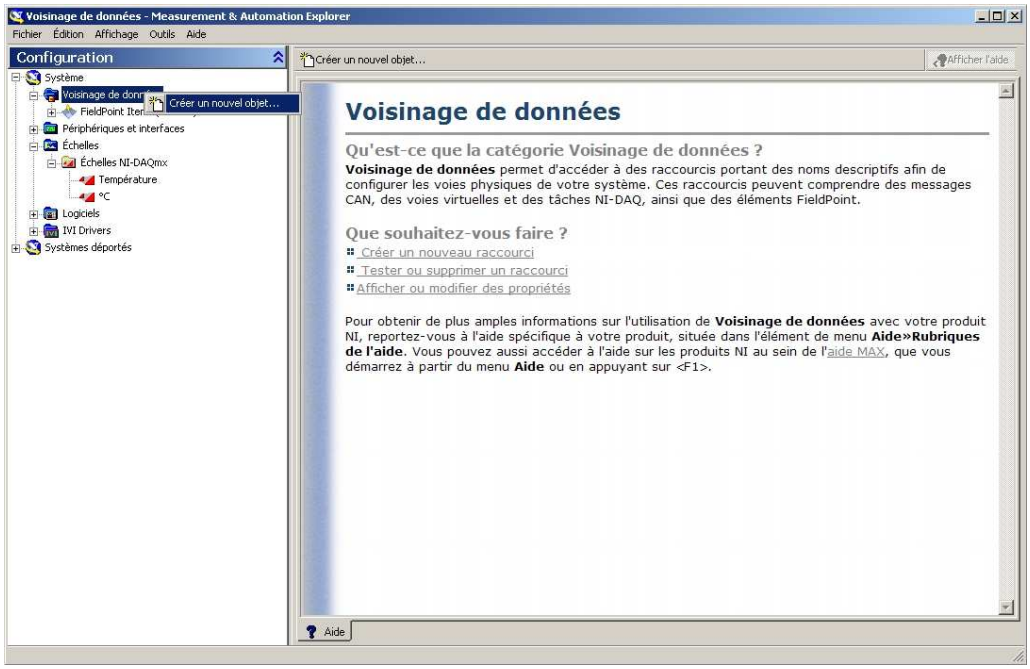


Figure 56 : Création Voie Globale 1

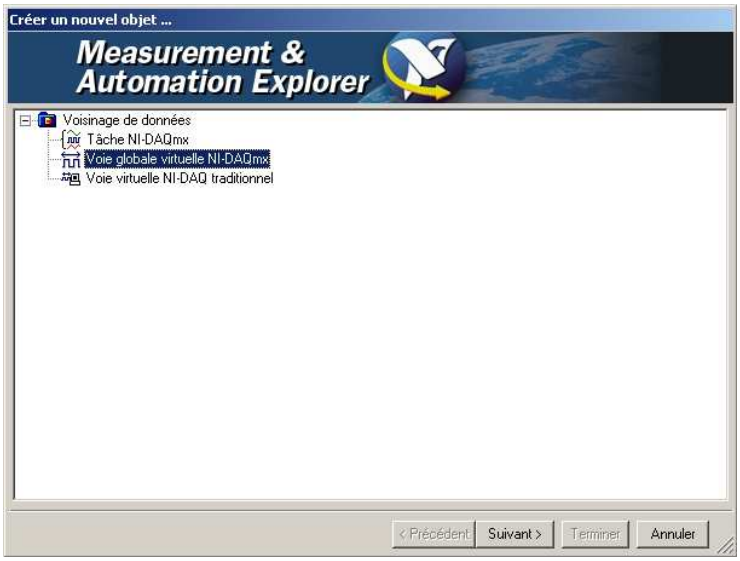


Figure 57 : Création Voie Globale 2

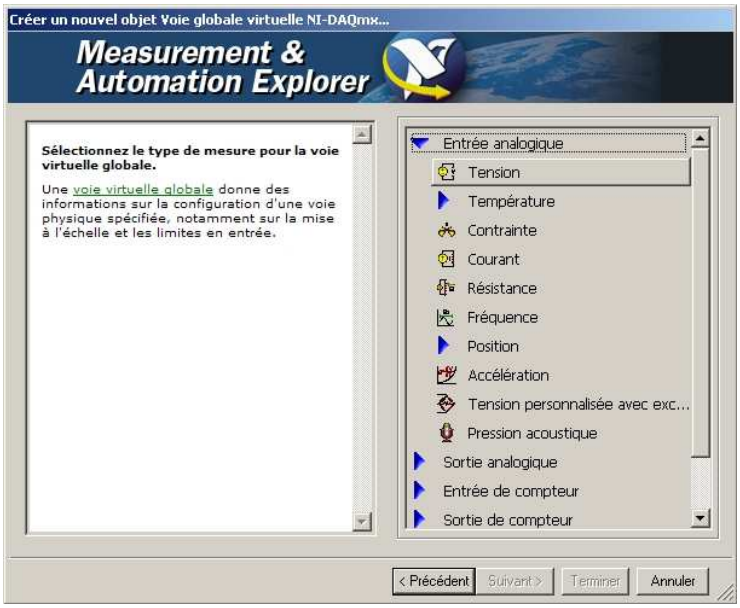


Figure 58 : Création Voie Globale 3 : choix entrée

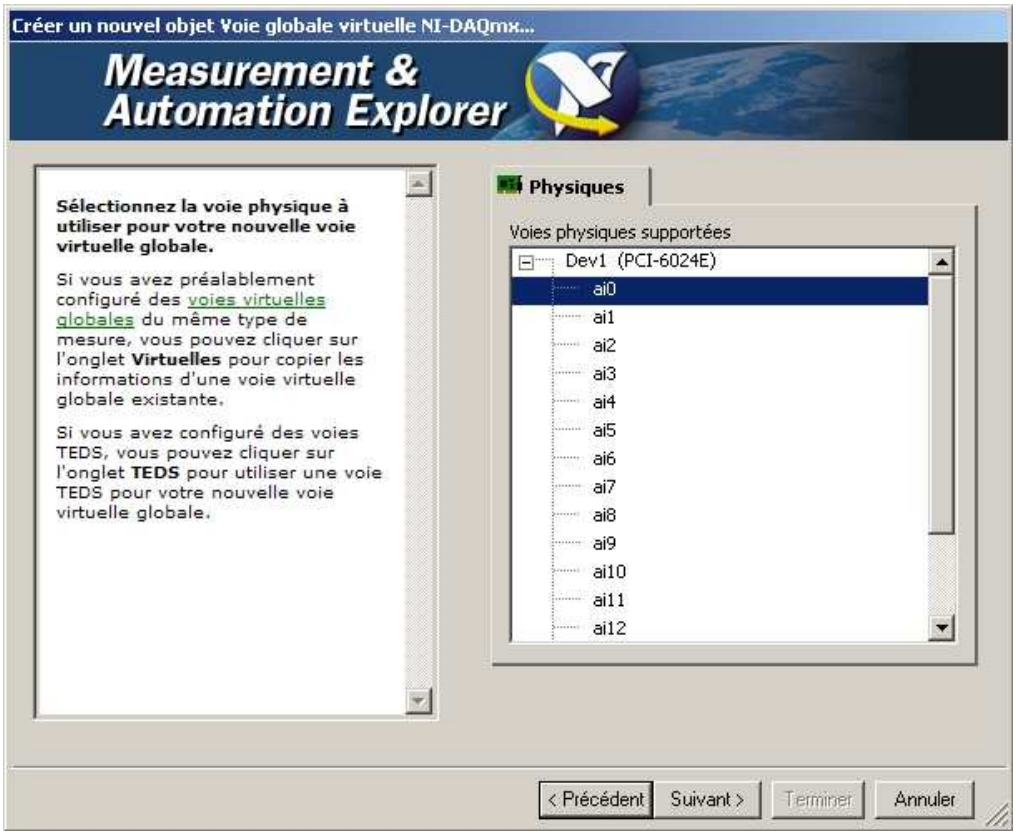


Figure 59 : Création Voie Globale 4 : choix numéro de voie

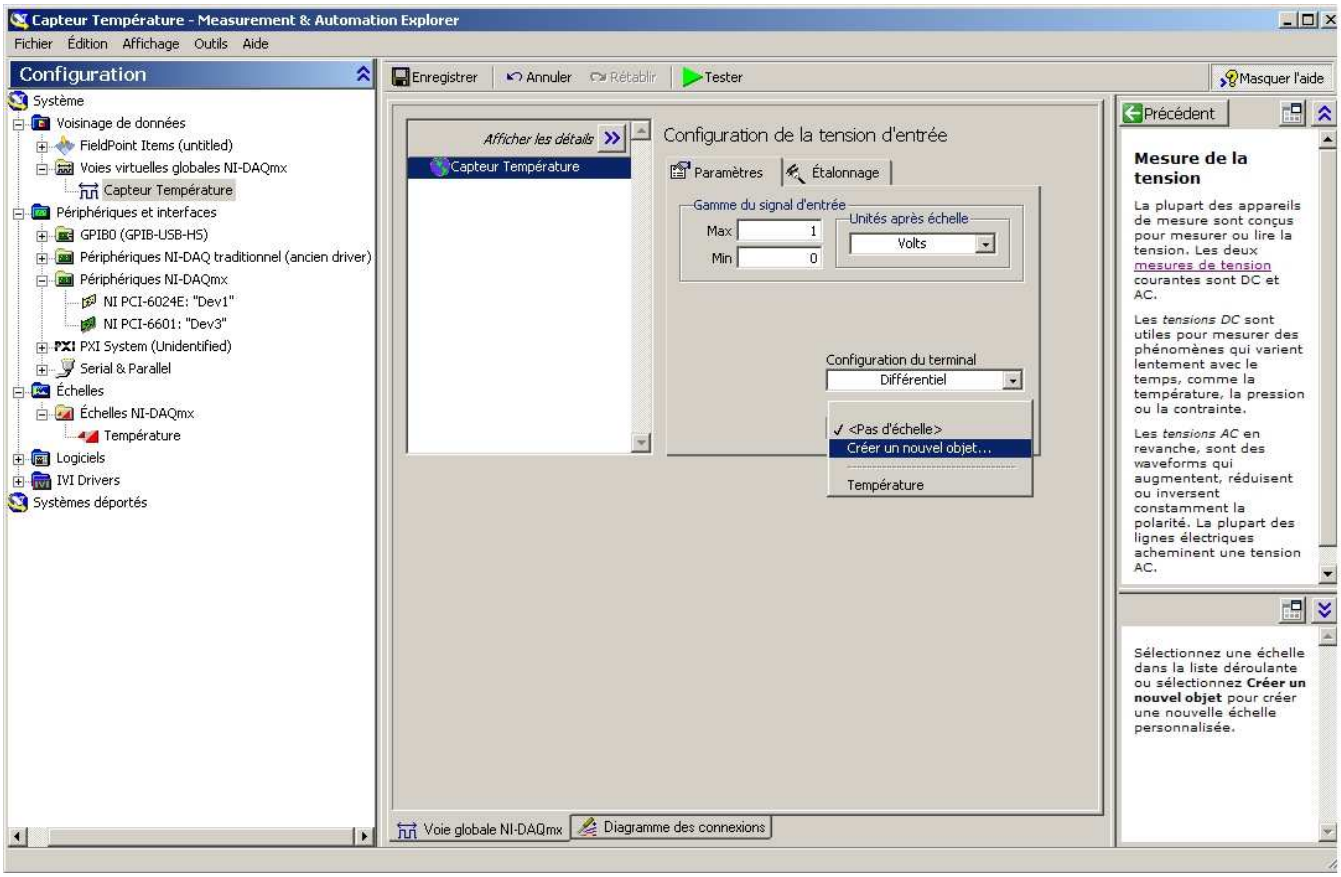


Figure 60 : Création Voie Globale 5 : Paramètres et échelle

Attention : en créant une échelle n'oublier pas de changer la gamme du signal, si par exemple la gamme 0-10V devient après l'utilisation d'une échelle linéaire de pente 1000 une gamme 0-10000.

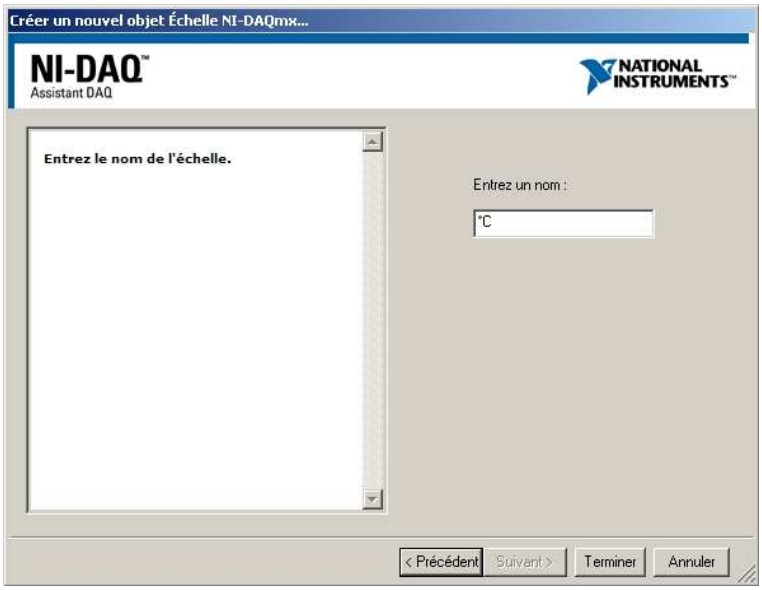


Figure 61 : Création Voie Globale 6 : nom échelle

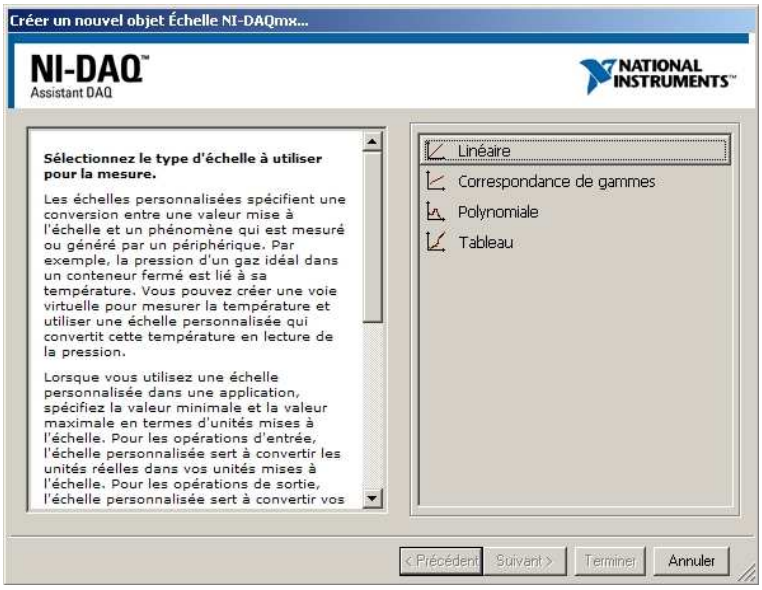


Figure 62 : Création Voie Globale 7 : création échelle

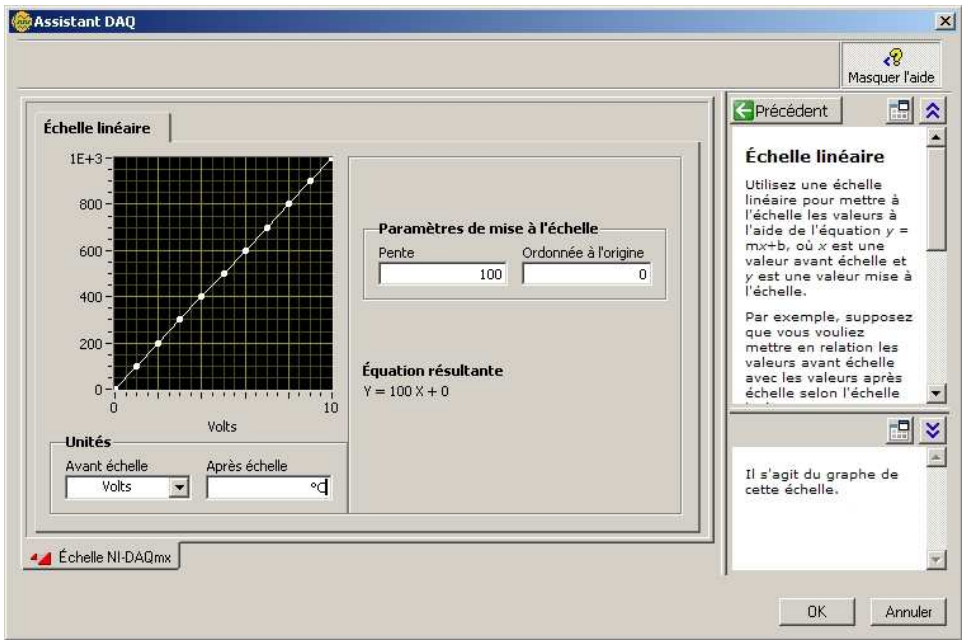


Figure 63 : Création Voie Globale 8 : paramètres échelle

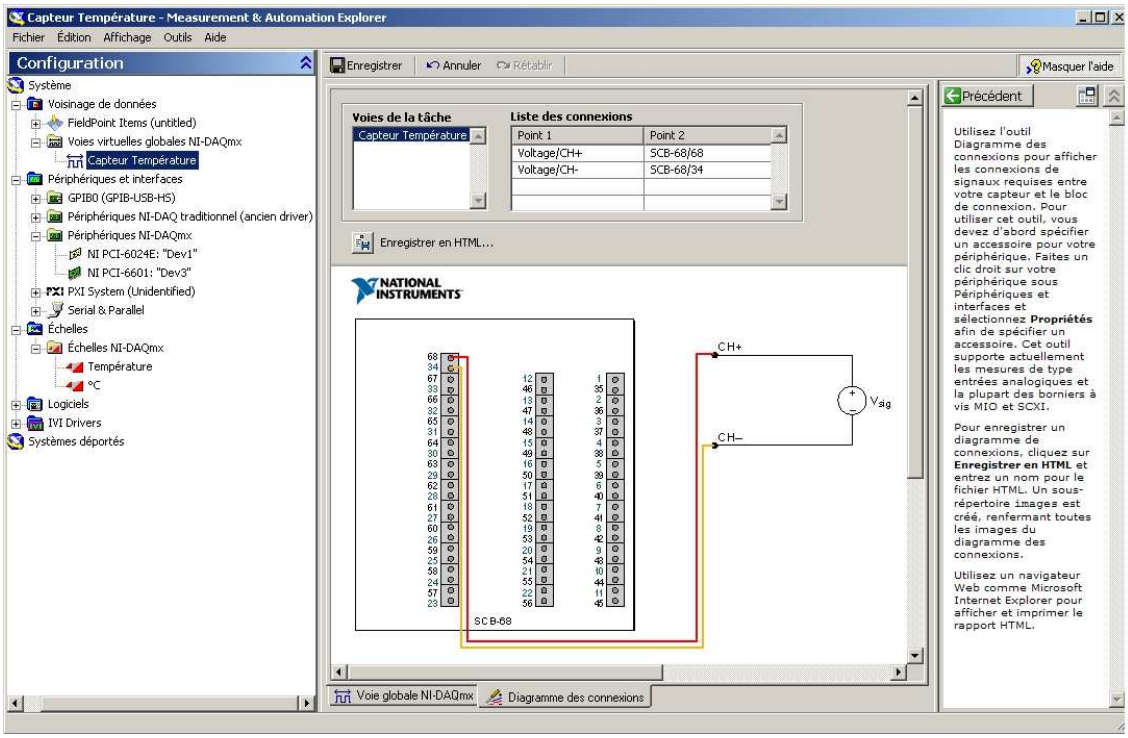


Figure 64 : Création Voie Globale 9 : Connexions

10.3. Acquisition analogique sous labview

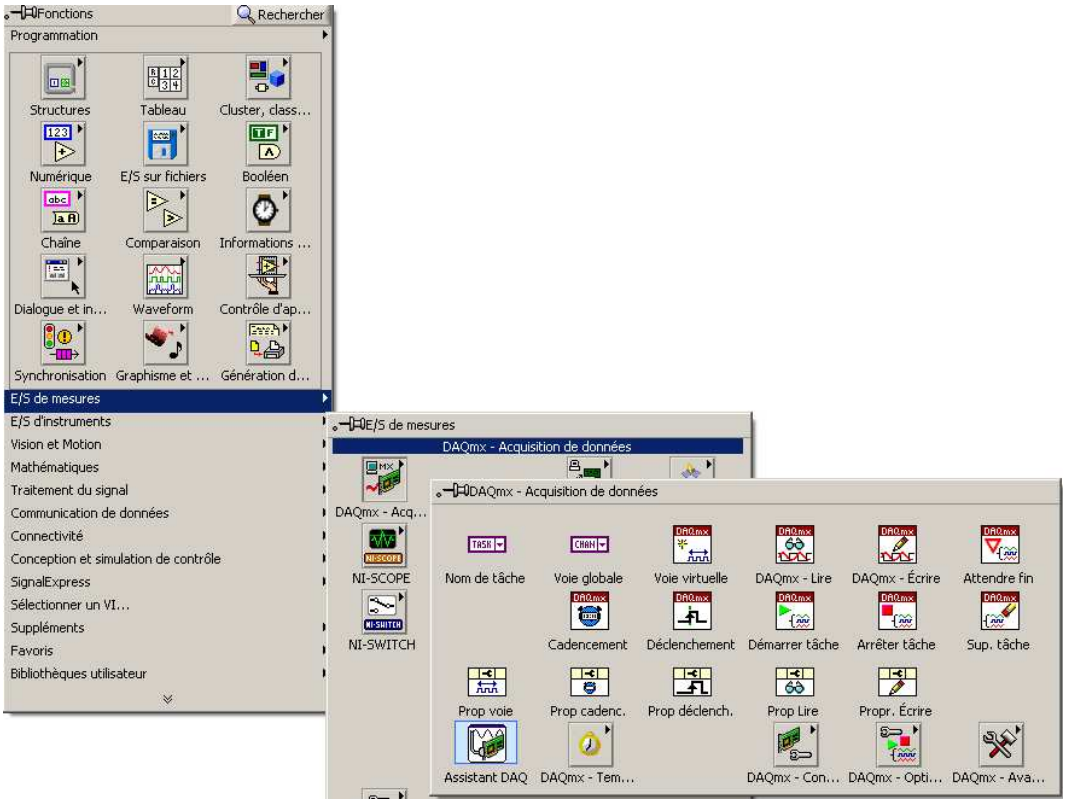


Figure 65 : Palette DAQmx

La palette acquisition de données comporte une seule catégorie de fonctions que vous devrez configurer suivant le type d'acquisition, génération, comptage… que vous désirez.

Nous allons nous intéresser principalement à l'acquisition analogique sous labview en sachant que pour les différents types d'acquisition la différence réside dans le paramétrage des fonctions.

2 types d'acquisition sont possibles :

- Acquisition simple non bufférisé : «time software». Elle sera employée pour des mesures ne nécessitant pas de rapidité (type DC) ou de «timing» précis ou une acquisition simple d'une ou plusieurs valeurs.
- Acquisition bufférisé : «time hardware». Elle sera employée lors d'une acquisition rapide ou des acquisition nécessitant la connaissance du temps, par exemple pour effectué des calculs FFT par la suite.

Pour utiliser les fonctions d'acquisition, un point important à connaître est la fréquence d'échantillonnage, c'est-à-dire la vitesse à laquelle vous allez acquérir vos données. Plus la vitesse est importante plus vous allez acquérir de données dans le même laps de temps et donc vous aurez une meilleure représentation de votre signal. Par exemple produire un signal de 1Hz en utilisant 1000 points pour un cycle à 1000 éch/s produit un signal plus fin qu'utiliser 10 points à une fréquence d'échantillonnage de 10 éch/s.

Pour acquérir correctement un signal et mesurer correctement des fréquences le théorème de Nyquist - Shannon énonce qu'il faut utiliser une fréquence d'échantillonnage supérieur à 2 fois la fréquence la plus haute contenue dans le signal.

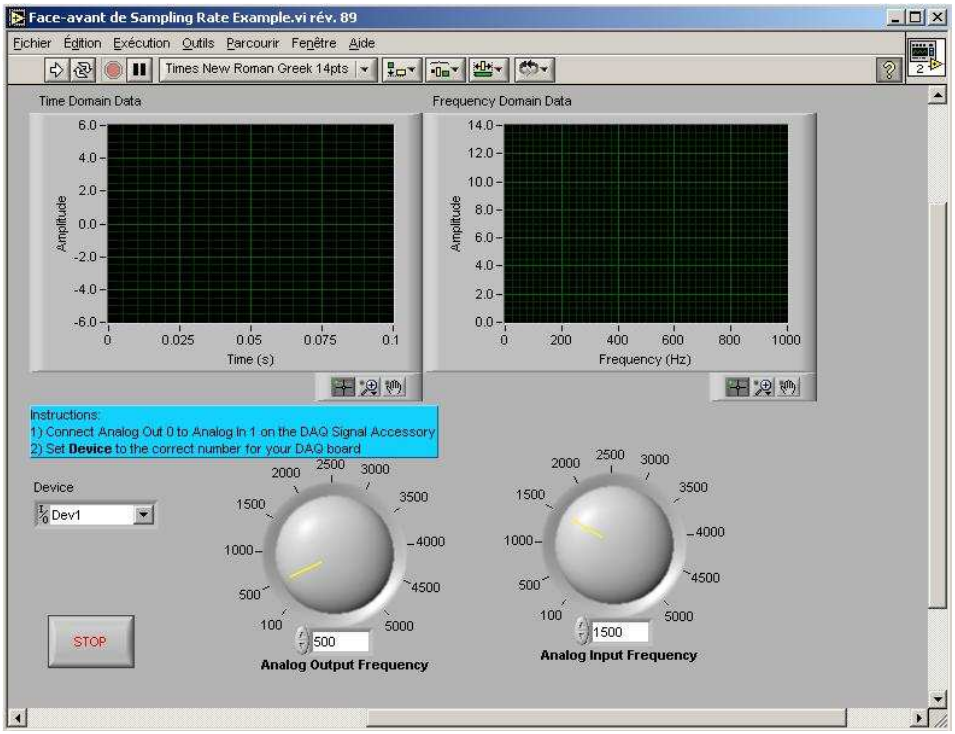


Figure 66 : Sampling Rate

Imaginons que nous avons une fréquence d'échantillonnage de Fs=100Hz et que le signal à mesurer comporte les fréquences suivantes : 25Hz, 70Hz, 160Hz et 510Hz, les fréquence en dessous de Fs sont bien échantillonnée mais les fréquence supérieure apparaissent comme des alias, 70Hz apparaît comme 30Hz, 160Hz comme du 40Hz et 510Hz comme du 10Hz. (alias F = ABS (plus proche multiple de Fs – Fréquence du signal).

10.3.1. Fréquence d'échantillonnage, aliasing et multiplexage

Pour observer les problèmes d'aliasing ouvrir le programme «sampling rate example» et réalisez les connections demandées.

Mettez 500Hz pour la génération et 1500Hz pour l'acquisition, vous observez bien un pic à 500Hz dans la fenêtre de fréquence (le théorème de Shannon est bien vérifié). Effectuer un zoom sur le Graph de votre signal, votre sinusoïde apparaît comme un triangle car la fréquence d'échantillonnage n'est que de 3 fois la fréquence du signal, ainsi vous ne pouvez pas observer la forme réelle de votre signal mais vous avez la bonne fréquence dans le graphe de fréquence.
Réduisez maintenant votre vitesse d'acquisition à 750Hz, il apparaît maintenant un alias du signal de 500Hz, notre graphe nous signale un signal de 250Hz.

Fs-Fsignal=750-500=250 alias du signal de 500Hz

Un autre point important avant d'aller plus loin est le fait que certaines cartes ne fonctionnent qu'en multiplexage, c'est-à-dire qu'elle possède un seul convertisseur analogique - numérique multiplexé entre les voies d'acquisition. Il existera donc un offset pour le même signal acquis sur deux voies simultanément. Cet offset sera lié directement à l'horloge du convertisseur.

10.3.2. Acquisition «Time Software»

Ouvrir le «VI Voltmètre simple.vi».

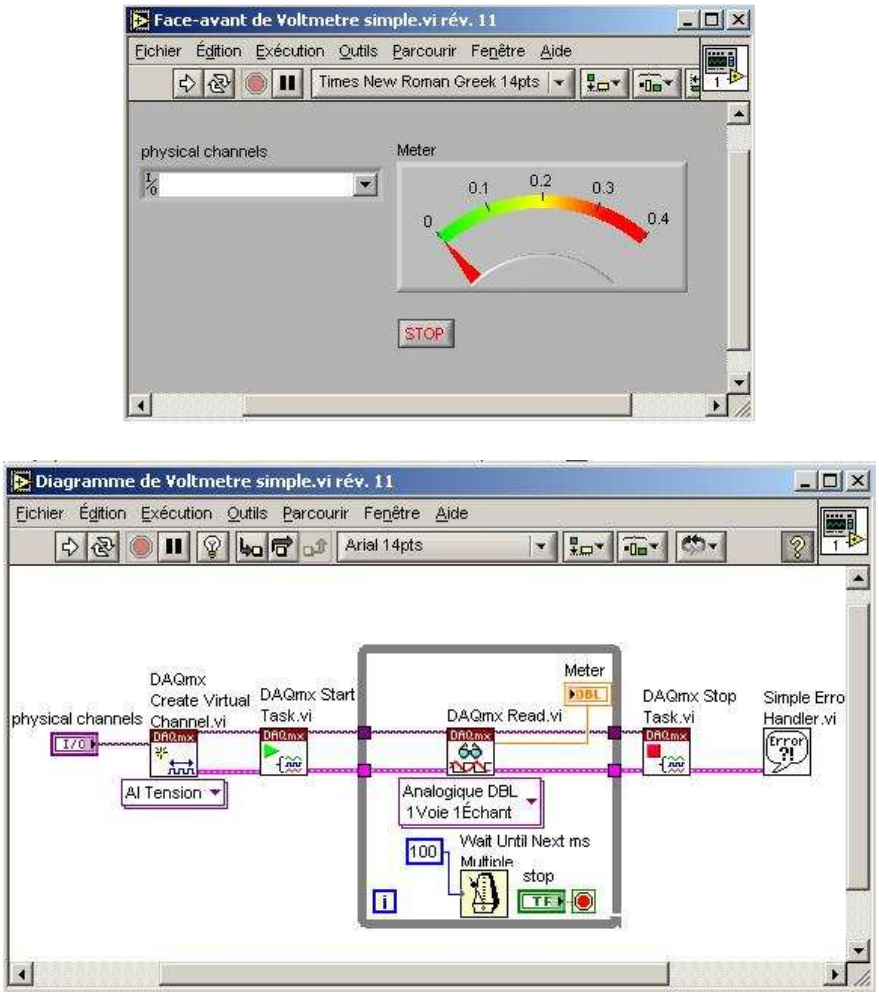


Figure 67 : Voltmètre simple

Observer les fonctions utilisées. Tout d'abord vous devez créez une voie virtuelle (si aucune voie n'a été créée sous MAX) que vous configurez ici comme étant une acquisition analogique, puis vous démarrez la tâche, une fois que cette fonction s'exécute la carte acquiert des mesures, puis vous lisez la mesure ici configurée de telle façon qu'une seule valeur est prise sur une seule voie, quand vous stoppez la prise de vos mesures vous arrêtez la tâche.
L'acquisition est dite «Software» car le temps d'échantillonnage est géré par le logiciel et l'ordinateur, nous prenons une valeur toutes les 100ms.

- Exercice : incorporez à ce VI la mesure de temps de boucle et visualisez ce temps à l'aide d'un indicateur.

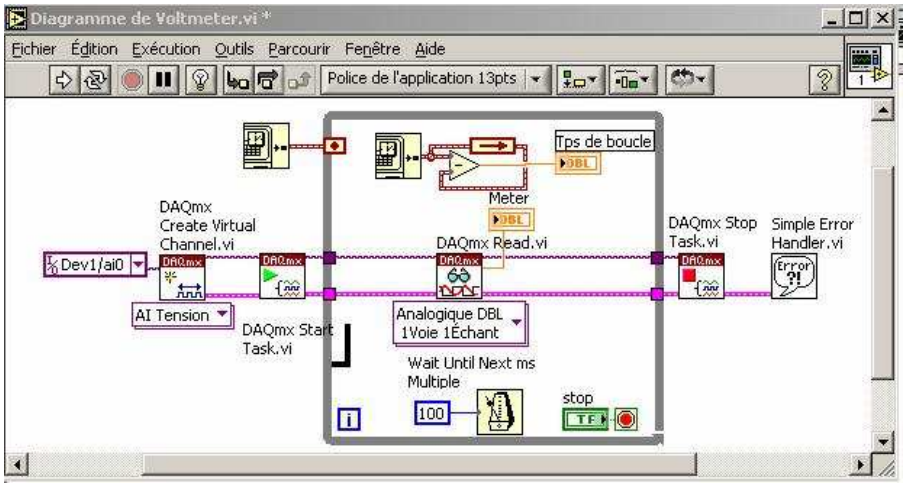
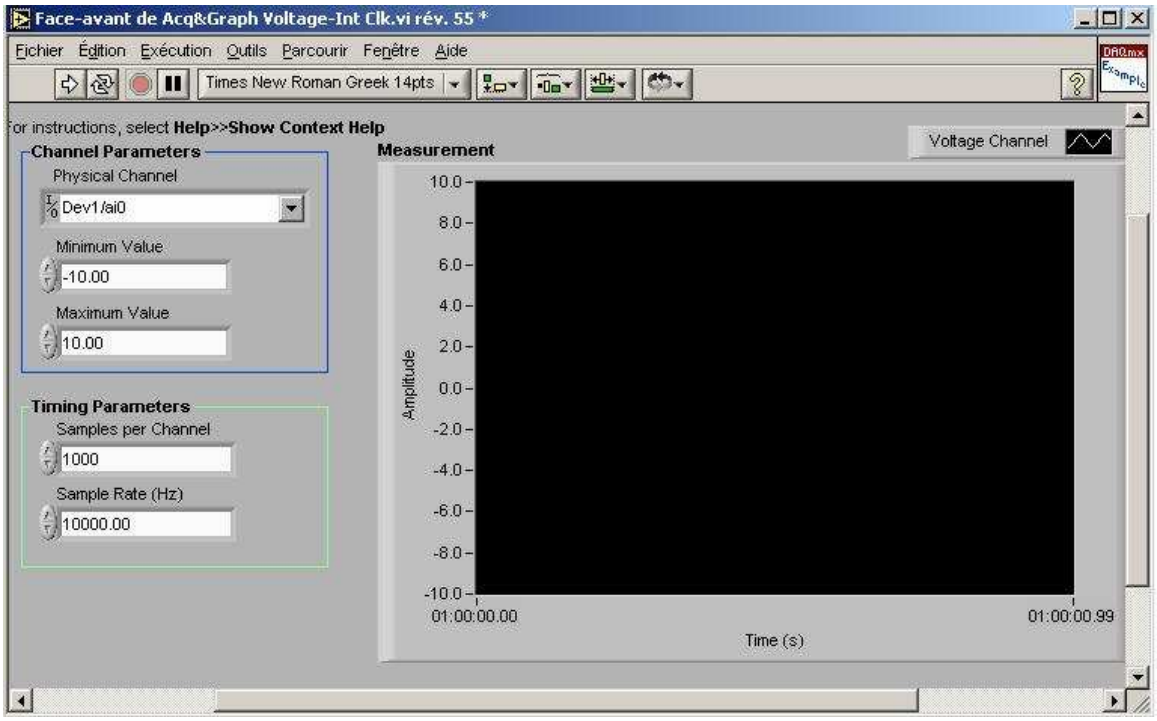


Figure 68 : Voltmètre simple et temps de boucle

10.3.3. Acquisition «Time Hardware»

Ouvrir l'exemple Acq&graph voltage-Int Clk.vi



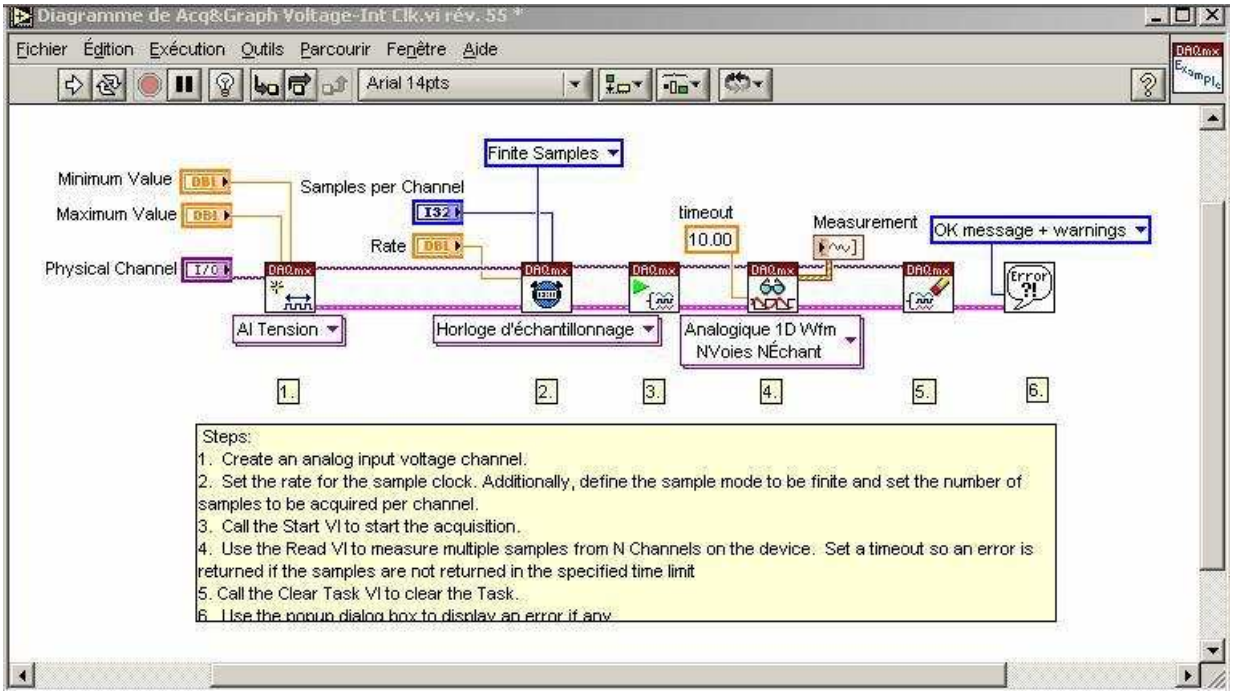


Figure 69 : Acq&Graph

Une fonction supplémentaire est utilisée, la fonction cadencement qui va permettre de régler la fréquence d'échantillonnage et le nombre d'échantillons acquis. Exécutez ce programme.

- Exercice : modifiez ce programme afin d'effectuer une acquisition continue.

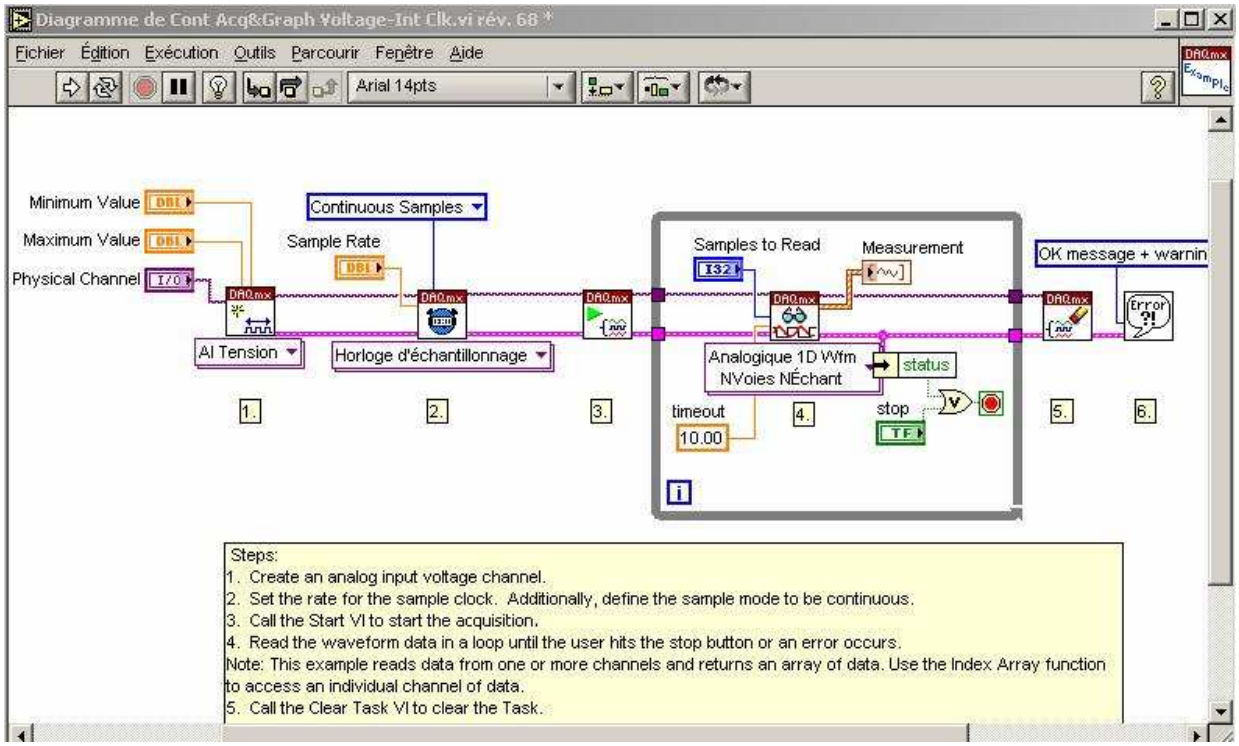


Figure 70 : Acq&Graph continu

(Vous pouvez ouvrir directement l'exemple correspondant). Modifier les valeurs de la fréquence d'échantillonnage et le nombre d'échantillons à acquérir.

Deux possibilités s'offrent à vous pour la lecture, mettre ou ne pas mettre le nombre d'échantillons à lire :

- Si vous ne mettez pas le nombre alors le driver lit tous les échantillons disponibles dans le buffer.
- Si vous imposez le nombre d'échantillons à lire alors attention de le placer au quart ou à la moitié du buffer pour éviter tout dépassement de celui-ci. Pour connaître ou imposer la longueur du buffer vous avez une propriété de buffer dans DAQ-mx constante et nœuds de propriété.

Fonctionnement du buffer : le driver rempli le buffer à la vitesse d'acquisition imposée et lit le nombre d'échantillons demandés et vide de ce nombre le buffer ainsi tant que la vitesse de lecture est suffisante le buffer n'est pas plein. Si le buffer est plein alors une erreur est renvoyée.

10.3.4. Acquisition déclenchée

Il est possible de déclencher des acquisitions, que ce soit sur des signaux numériques ou analogiques.

Reprendre l'exemple précèdent et incorporez la fonction Daq-mx démarrer un déclenchement, click droit sur source créer une commande, click droit sur front créer une commande. Cette fonction est placée avant la fonction DAQ-mx démarrer une tache.
Vous devriez avoir ce diagramme :

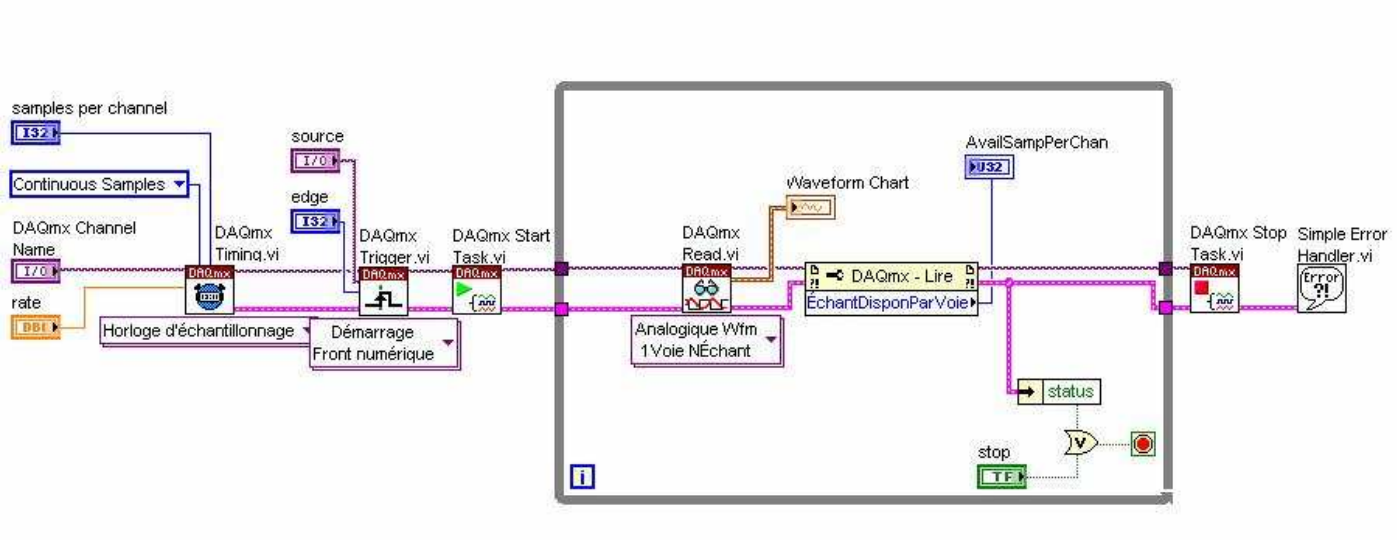


Figure 71 : Acquisition déclenchée

La source sera PFI0 correspondant au bouton du boîtier de démonstration. Lancer votre programme, tant que vous n'appuyez pas sur le bouton rien ne se passe.

10.3.5. Génération de code

Nous pouvons directement générer du code sous labview pour une acquisition (ou autre) simple comme nous avons vu précédemment pour cela placez dans votre diagramme une Constante de nom de tâche, click droit et Nouvelle tâche.

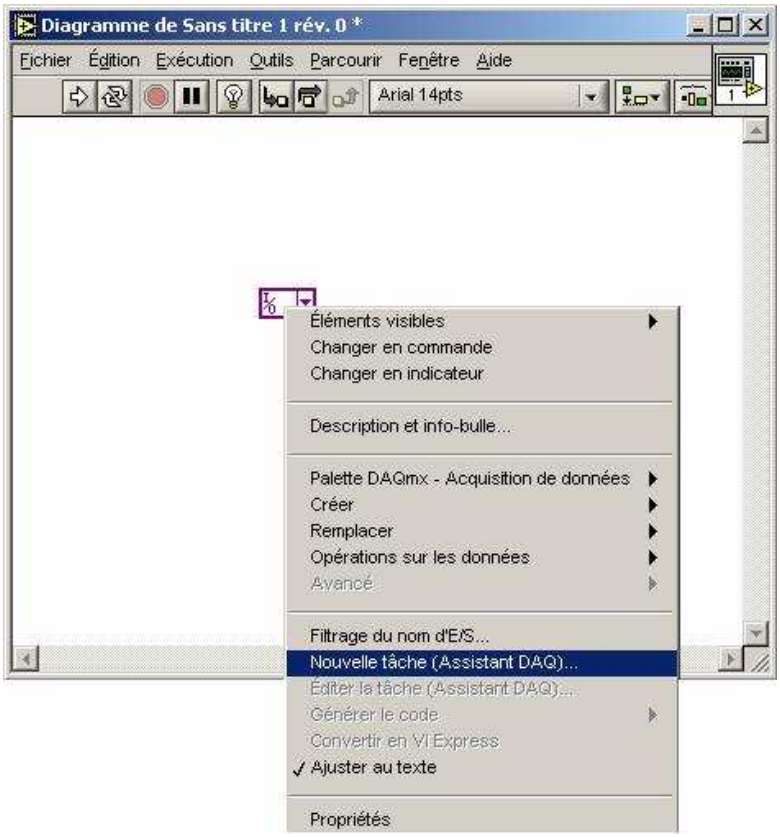


Figure 72 : Génération de code Constante Tâche

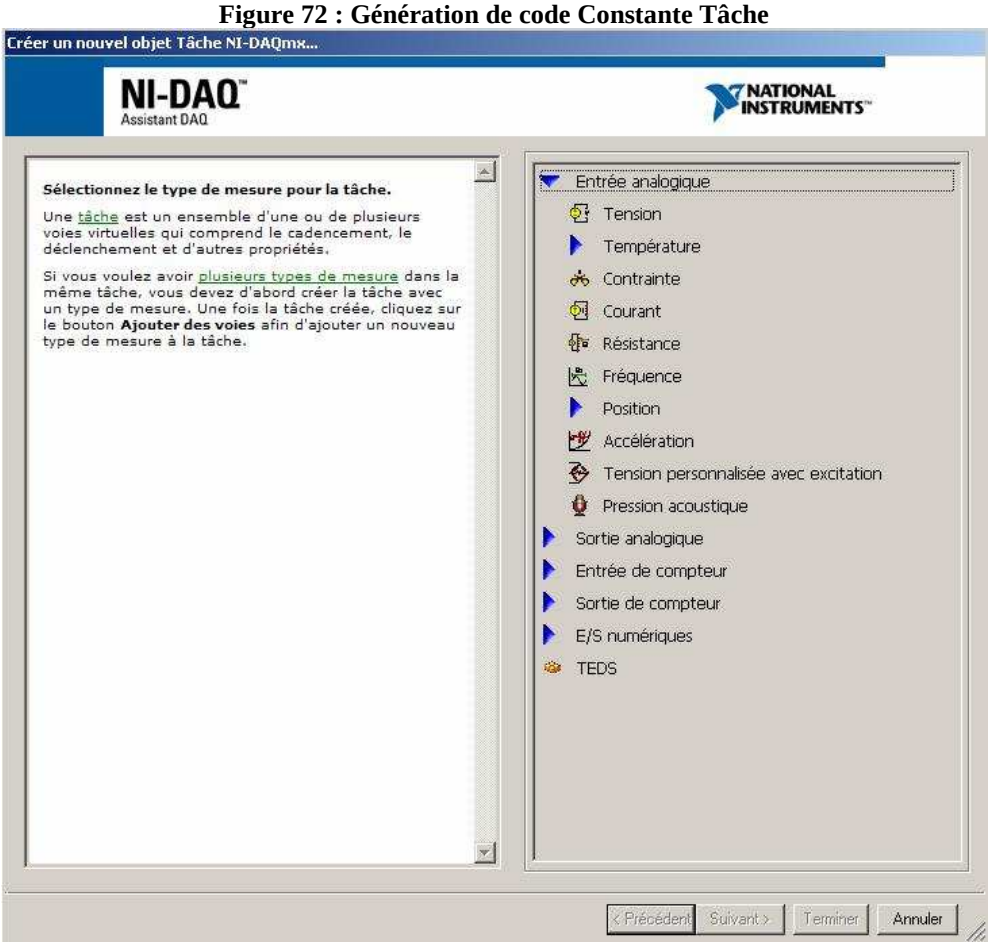


Figure 73 : Génération de code Choix du Types de mesures

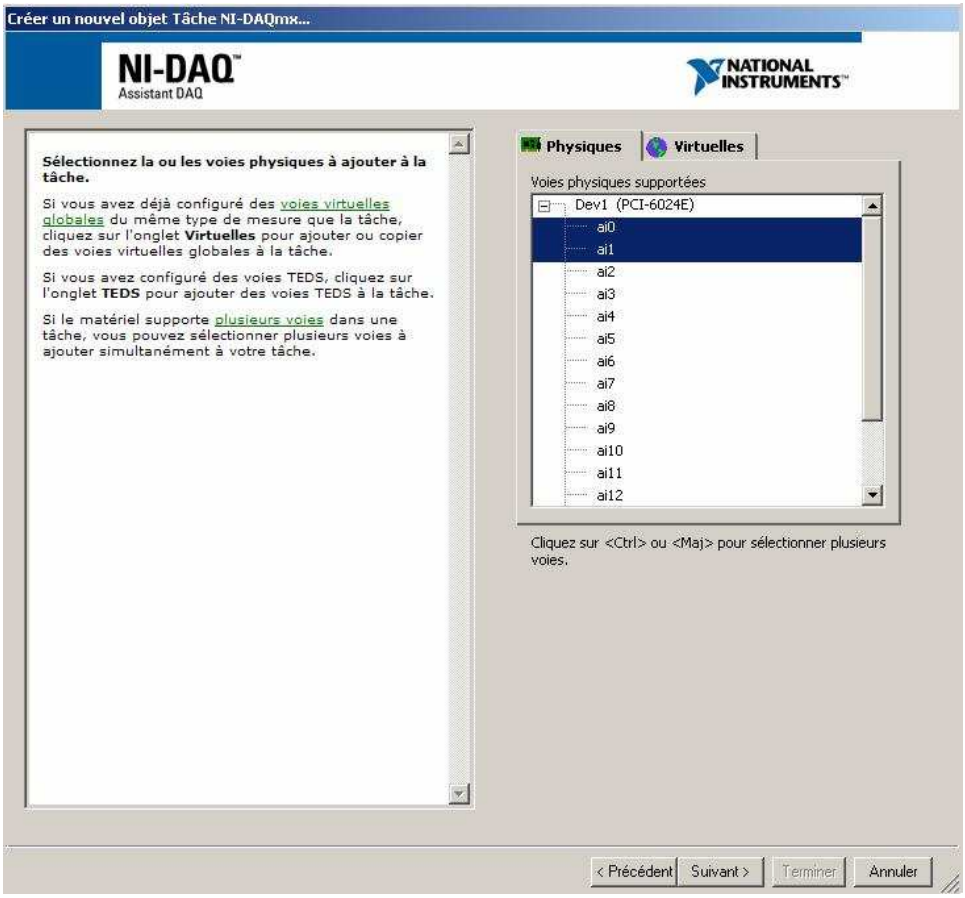


Figure 74 : Génération de Code Choix des voies

Figure 75 : Génération de code Choix du Nom

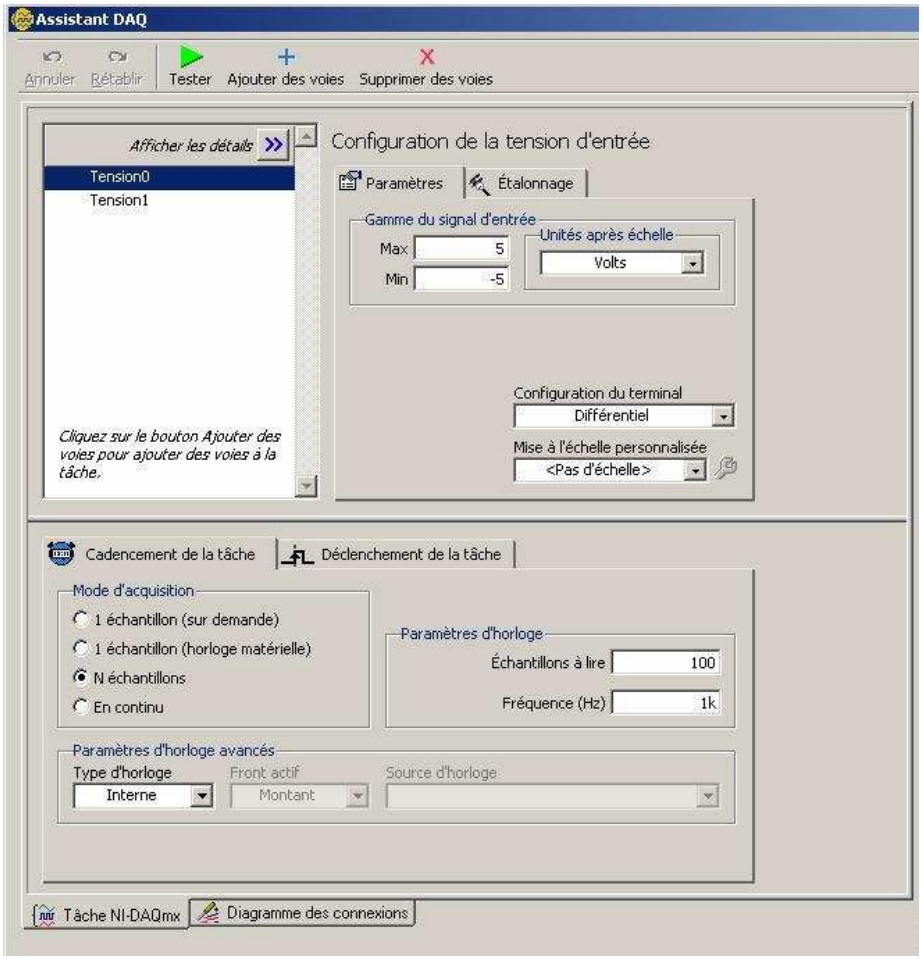


Figure 76 : Génération de code Propriétés de la Tâche et Visualisation connexions à réaliser

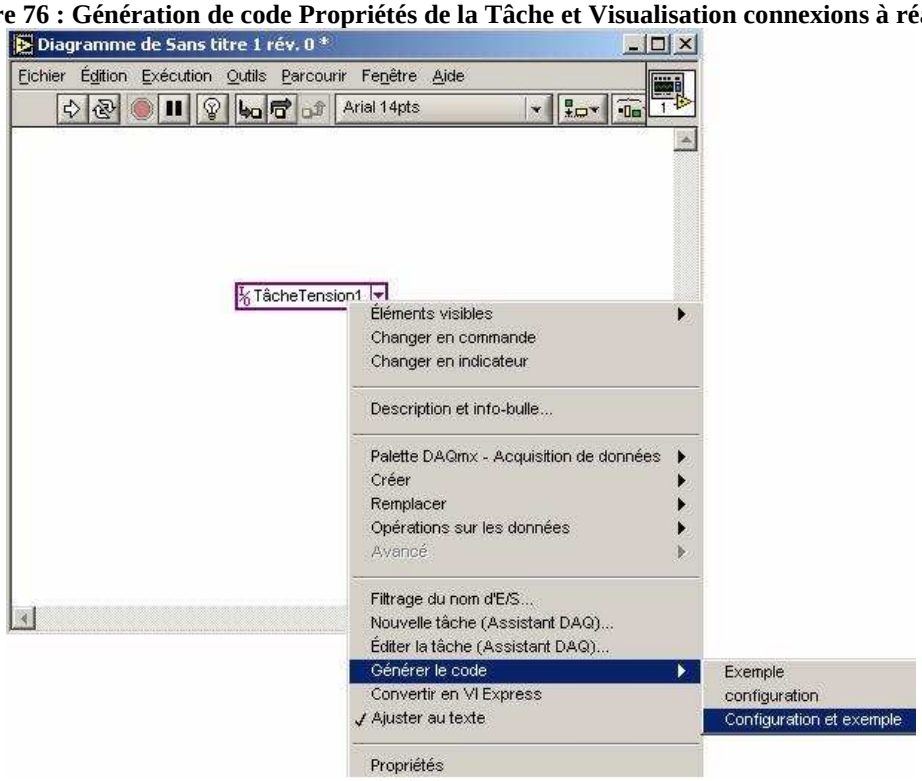


Figure 77 : Génération de code Générer du code sur le diagramme

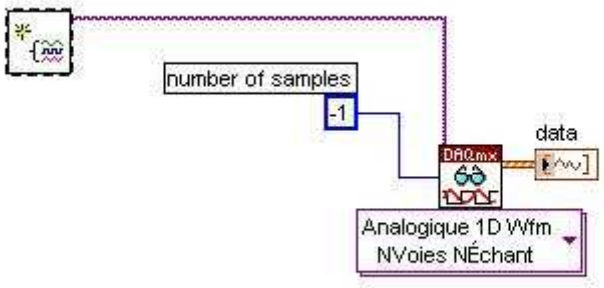


Figure 78 : Code généré

Votre code ainsi généré avec un sous VI de configuration.

11. GENERATION ANALOGIQUE DE DONNEES

La plupart du temps les voies de génération possèdent leurs propres Convertisseur Digital Analogique (DAC) mais ils sont mis à jour avec la même horloge, les voies sont donc synchronisées.

11.1. Génération d'un Voltage

Générer le code suivant :

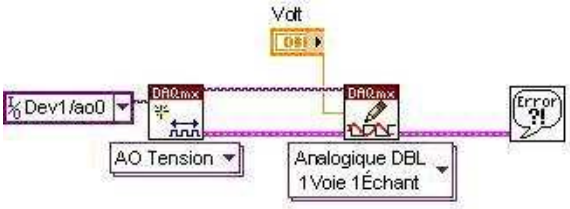


Figure 79 : Génération simple de tension

Il suffit de définir la voie utilisée et d'écrire une valeur numérique. La carte génère ainsi le voltage désiré.

11.2. Génération continue d'un Voltage

Modifier le code précédent afin de pouvoir modifier la valeur du voltage en continue. Pour cela faire comme l'acquisition «time software» vue précédemment.

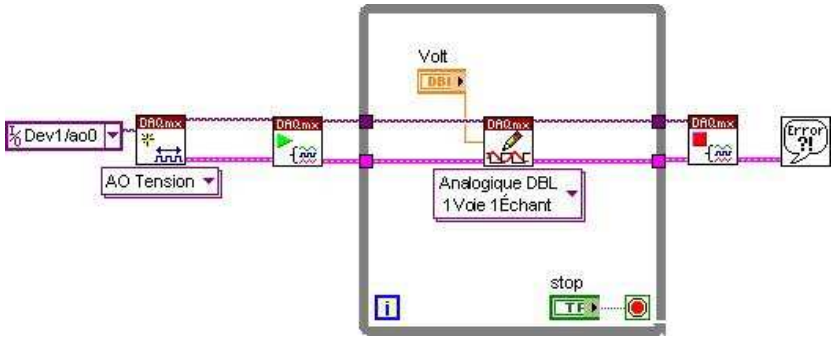


Figure 80 : Génération continue de tension

Vous pouvez maintenant modifier en continu votre tension.

11.3. Génération finie bufférisée

Nous utiliserons les mêmes fonctions que pour une acquisition bufférisée avec une fonction supplémentaire qui nous permettra d'attendre la fin de la génération avant de stopper la tâche. De plus 2 méthodes s'offrent à vous.

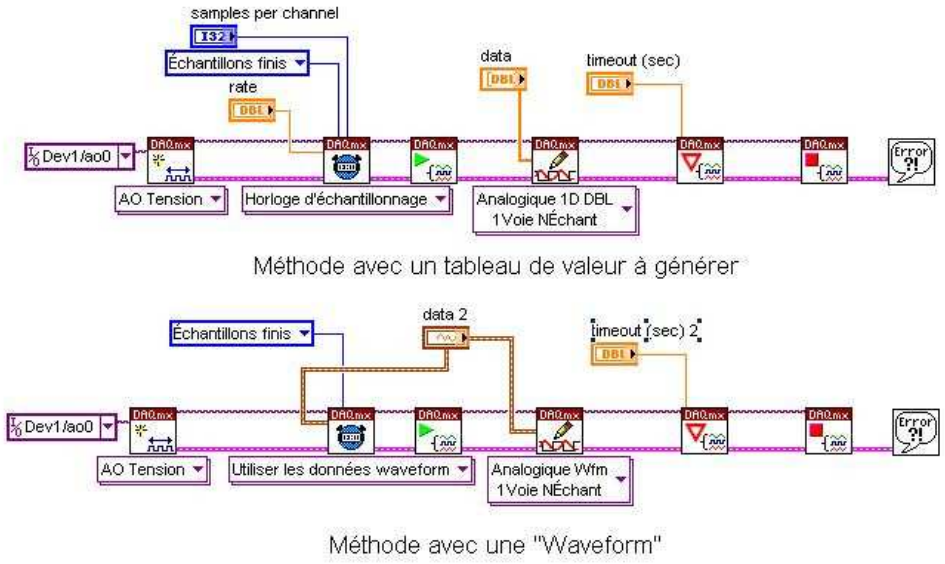


Figure 81 : Génération Finie Bufferisée

Si vous utilisez une «Waveform» alors l'information sur le temps et le nombre de points est compris dans celle-ci.

Fréquence signal = (cycles x fréquence de mise à jour) / nombre de points du buffer

Une fonction supplémentaire, «attendre jusqu'à la fin», a été intégrée permettant de vérifier si la tache est terminée avant d'arrêter.

11.4. Génération continue bufférisée

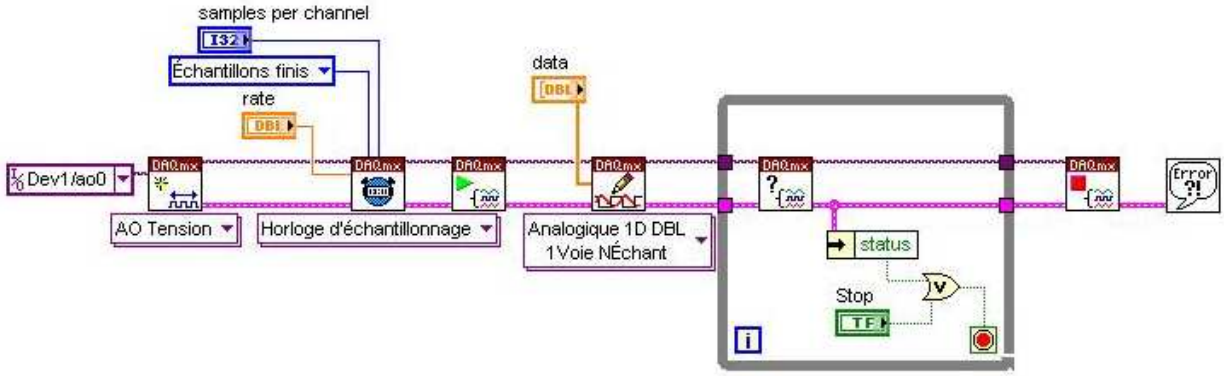


Figure 82 : Génération Continue Bufferisé

Comme pour l'acquisition il nous faut ajouter une boucle «while» mais dans celle-ci la fonction «Tâche finie ?» est mise à la place d' «attendre jusqu'à la fin».

Parcourez les différents exemples DAQmx afin d'observer les différentes possibilités d'acquisition ou de générations de signaux.

Note : vous avez vu que nous utilisons les mêmes fonctions quelques soient le type de sorties ou d'entrées, aussi paramétrez toujours en premier vos fonctions avant de créer vos commande ou indicateurs pour éviter tout problème de style commande de voie analogique pour une acquisition numérique par exemple.
Avec tout utilisation d'exemple, si vous voulez modifier un exemple copiez le d'abord afin d'éviter de modifier l'exemple sous Labview.

Bibliographie

- « Labview programmation et application » Francis Cottet
- « Manuel utilisateur » National Instruments
- « Labview Basic I » et « Labview Basic II » National Instruments
- http://cnx.org/content/col10241/latest
- http://www.ni.com/academic/
- http://www.ni.com/academic/lv_training/how_learn_lv.htm
- http://egweb.mines.edu/eggn350/labview/
- http://www.iit.edu/~labview/Dummies.html
- http://www.mech.uwa.edu.au/jpt/tutorial/ieindex.html
- http://zone.ni.com/devzone/cda/tut/p/id/5055
- http://sine.ni.com/idnetwork/