femto-st SCIENCES & TECHNOLOGIES

afaq ISO 9001 Qualité AFNOR CERTIFICATION

SERVICE COMMUN ÉLECTRONIQUE ET INSTRUMENTATION

LabVIEW : des fondamentaux aux structures avancées pour l'instrumentation et le contrôle-commande

UFC UNIVERSITÉ DE FRANCHE-COMTÉ — cnrs — ensmm — utbm université de technologie Belfort-Montbéliard — UBFC

Objectifs de ce cours

- ✓ **Les fondamentaux de la programmation graphique en langage G**
- ✓ **Les concepts de haut niveau structurants**
  - ➢ Machines à états
  - ➢ Modèles parallèles
  - ➢ Outils de synchronisation
  - ➢ Programmation évènementielle

=> indispensables au développement d'une application performante intégrant une interface Homme/Machine

- ✓ **Les trois thèmes suivants font l'objet de séances de travaux pratiques**
  - ➢ L'acquisition de données capteurs
  - ➢ La génération de signaux
  - ➢ Les différentes méthodes d'enregistrement de données

Sommaire

Sommaire (Suite)

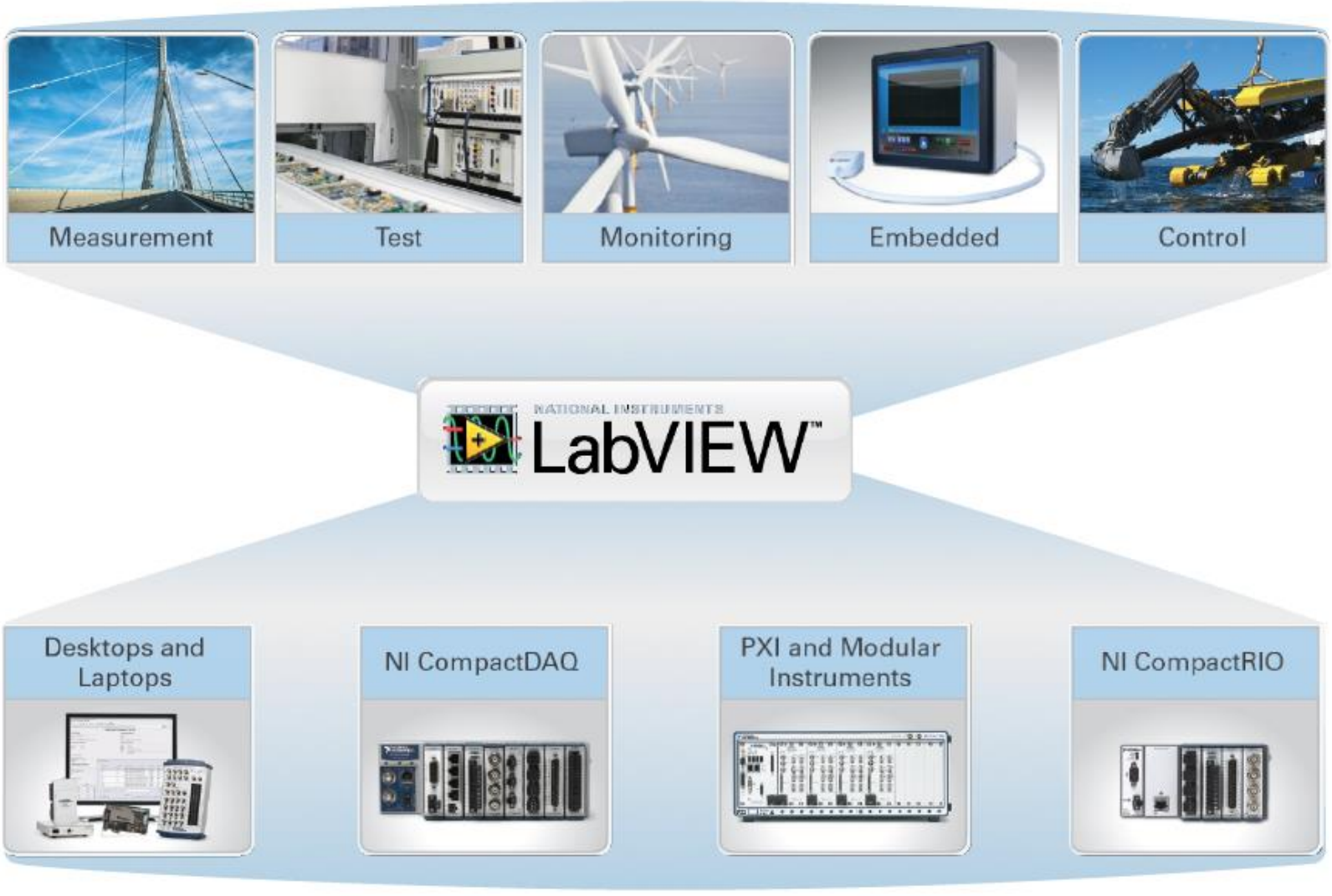
1. Qu’est-ce que LabVIEW

LabVIEW est un environnement de programmation graphique dédié au développement de systèmes de mesure, de test, d’instrumentation et de contrôle-commande sophistiqués.

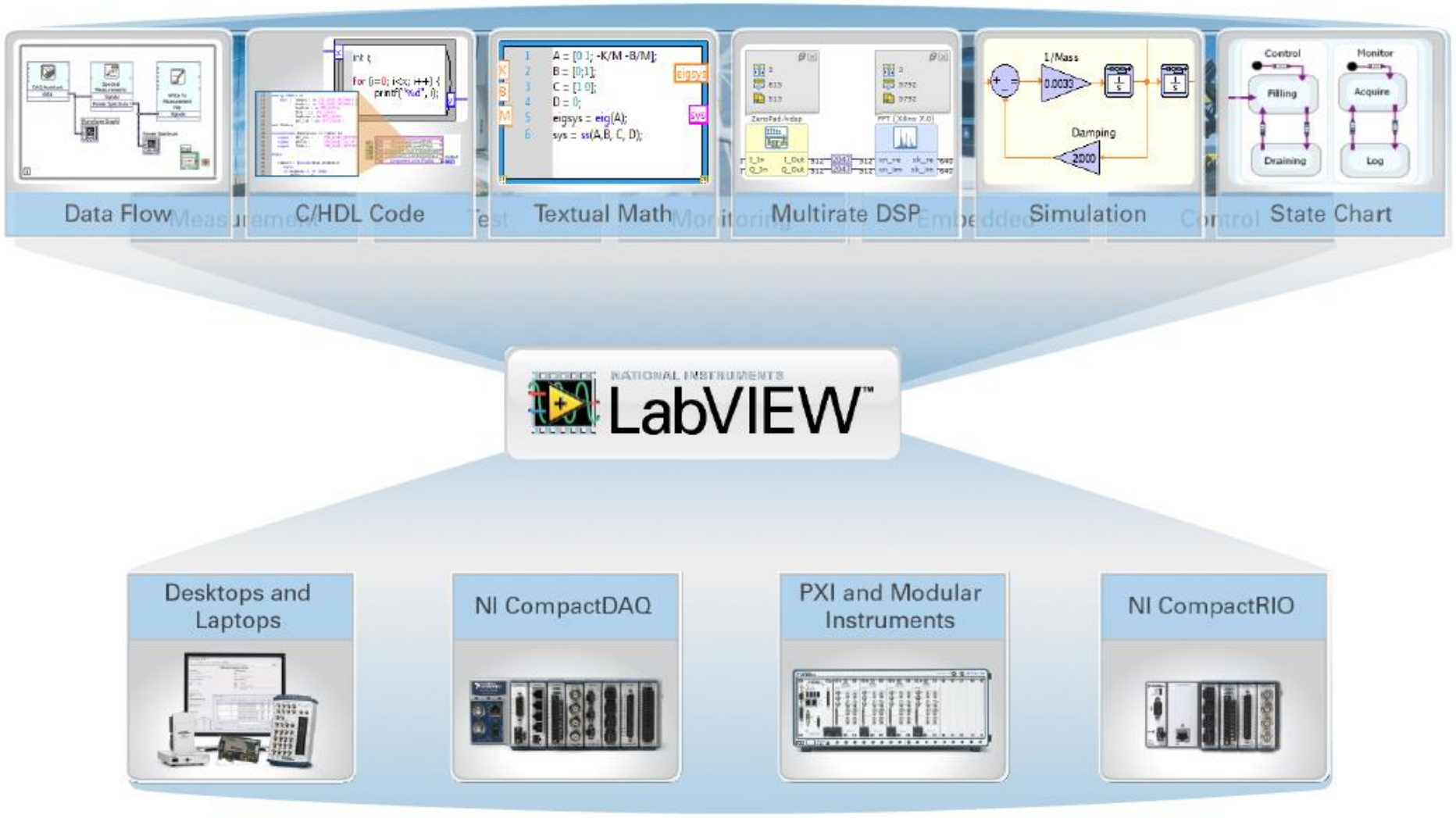
Utilisable avec une grande variété de matériels, il est multi-plate-forme, multi-cible et intègre un grand nombre de bibliothèques en natif.

De plus en plus de constructeurs fournissent une bibliothèque LabVIEW avec leurs matériels.

2. Conception graphique de systèmes



2. Conception graphique de systèmes



3. Caractéristiques fondamentales

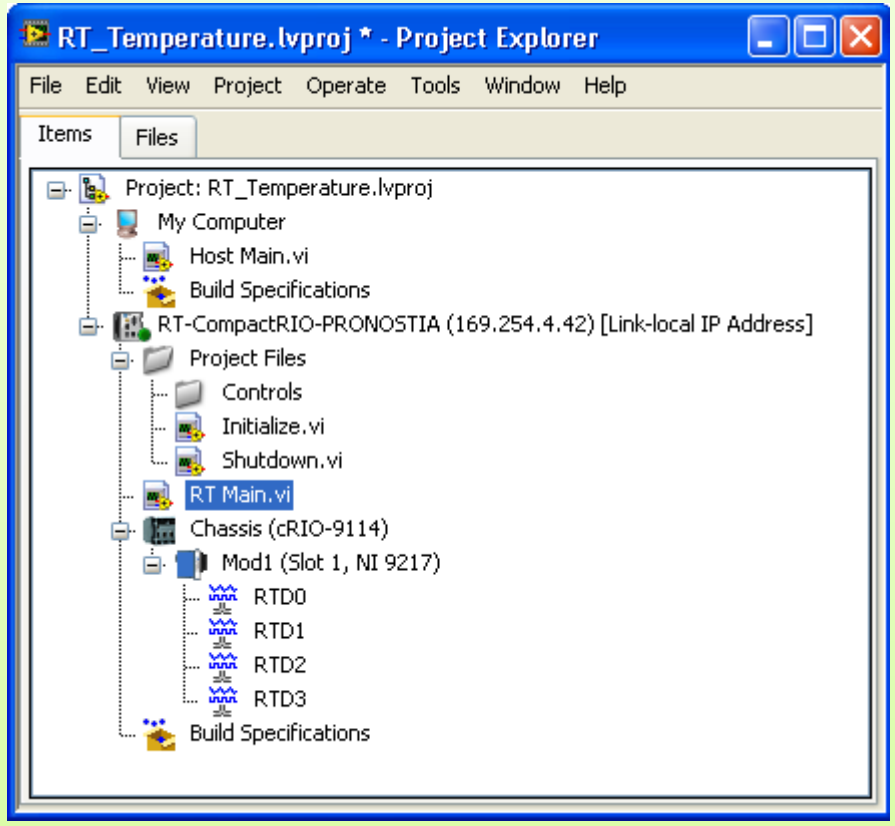
- ✓ **Langage graphique basé sur la notion de flux de données**
- ✓ **Programmation mixte séquentielle et parallèle**
- ✓ **Multi-plate-forme et multi-cible**
- ✓ **Compilé et non interprété**
- ✓ **Programmation évènementielle**
- ✓ **Synchronisation de processus parallèles**
- ✓ **Programmation orienté objet**
- ✓ **Multithreading et temps réel**

4. Environnement de développement

4.1. Explorateur de projet

Les dernières versions de LabVIEW intègrent la notion de projet dont l'intérêt premier est de faire apparaitre :

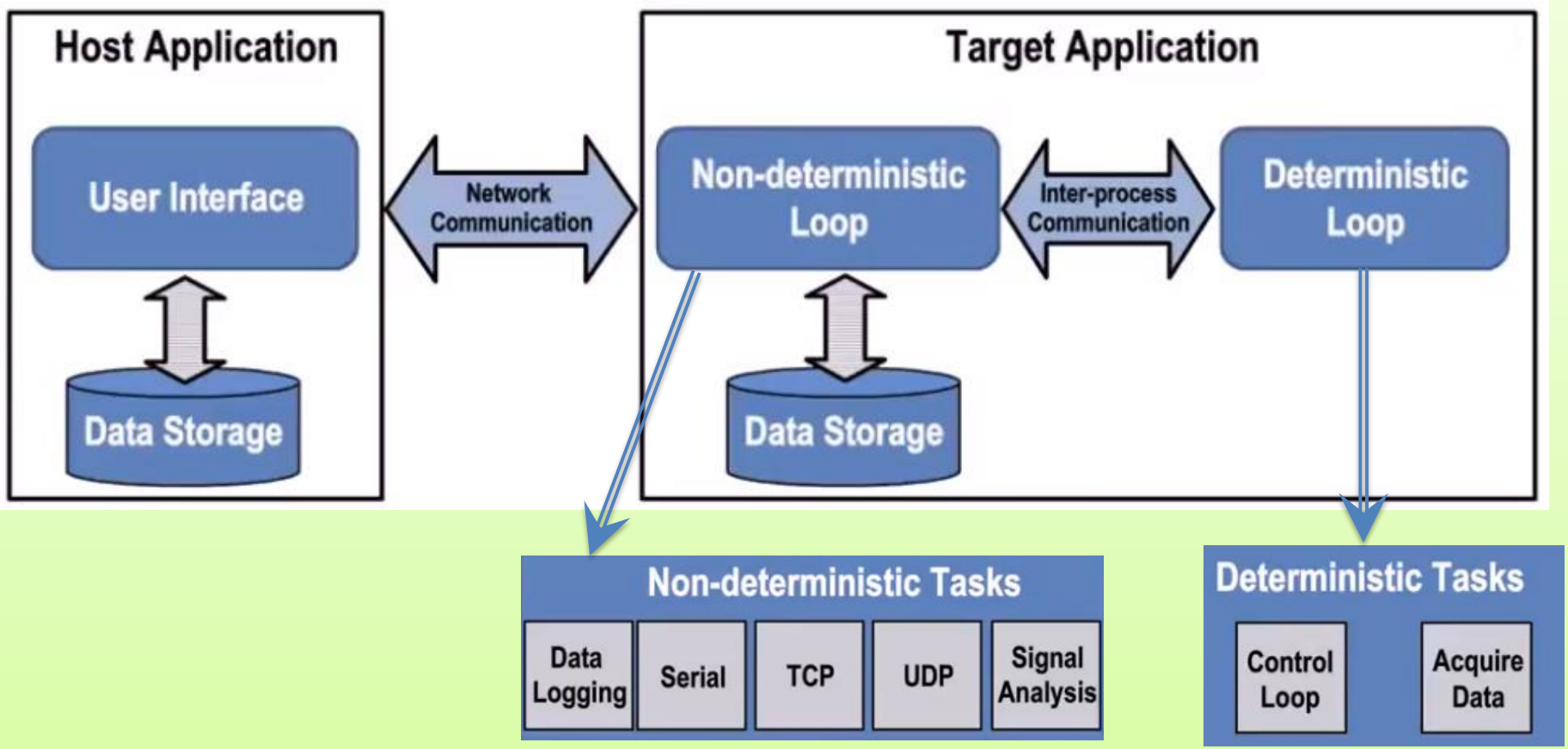
- **toutes les cibles ou matériels,**
- **les portions de logiciel liés à chacune d'entre elles,**
- **avec les bibliothèques,**
- **et toute la documentation que l'on souhaite y intégrer.**



4. Environnement de développement

4.1. Explorateur de projet (suite)

Liens logiques dans le cas d'une cible temps réel



4. Environnement de développement

4.1. Explorateur de projet (suite)

La fenêtre de l'explorateur de projet permet donc de :

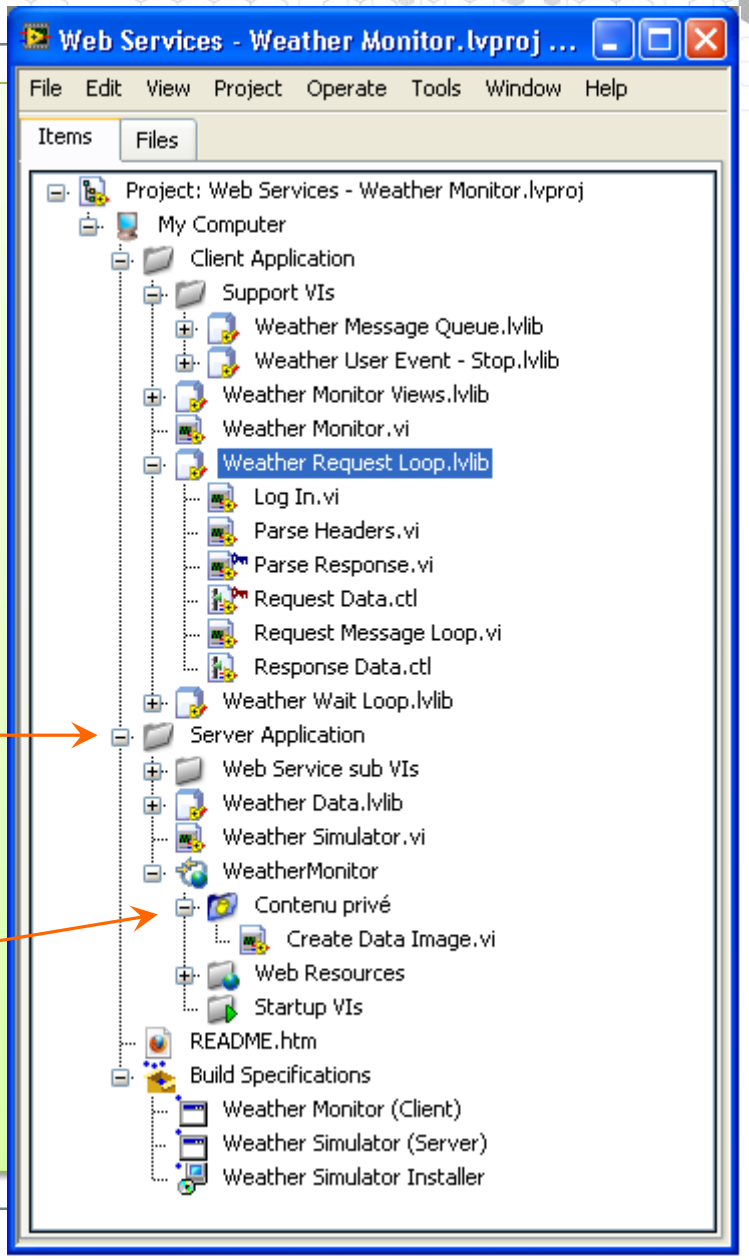
- ✓ **Rechercher, organiser et accéder aux fichiers d'un projet**
- ✓ **Éviter, détecter et résoudre les liens incorrects**
- ✓ **Déployer (ou télécharger) des fichiers sur les cibles**
- ✓ **Gérer le code des options de construction (exécutables, installeurs, zip)**
- ✓ **Intégrer du code source**
- ✓ **…**

4. L'environnement de développement

4.1. Explorateur de projet (suite)

Types de fichiers natifs dans LabVIEW

- Extensions courantes :
  - ✓ Projet LabVIEW : .lvproj
  - ✓ Librairies : .lvlib
  - ✓ VI (instrument virtuel) : .vi
  - ✓ Commande personnalisée : .ctl
- Ajout de dossiers à un projet
  - ✓ Dossier virtuel : dossier qui n'a pas d'existence sur le disque dur ou un support physique de mémoire
  - ✓ Auto-populating folder : mise à jour continuelle du contenu du répertoire



Micro-Travaux Pratiques dirigés

Prérequis

- Créer un répertoire LabVIEW Projects sous « Mes documents »

I. Création d'un projet vide

- Créer des dossiers virtuels
- Insérer des fichiers dans chaque dossier du projet
- Test la suppression d'un fichier sur le disque dur

4.2. Les composantes d'un instrument virtuel (VI)

La palette d'outils commune à la face-avant et au diagramme

Elle permet de sélectionner et manipuler tous les objets à disposer dans le diagramme.

Sélection d'outils automatique

Modifie la valeur d'un contrôle

Créé les fil de liaison

Raccourci palettes

Insère ou supprime un point d'arrêt

Insère une sonde

Sélectionne, modifie la position ou la taille d'un objet

Insère, édite ou modifie du texte

Déplace l'espace de travail dans une fenêtre

Pipette de sélection de couleur

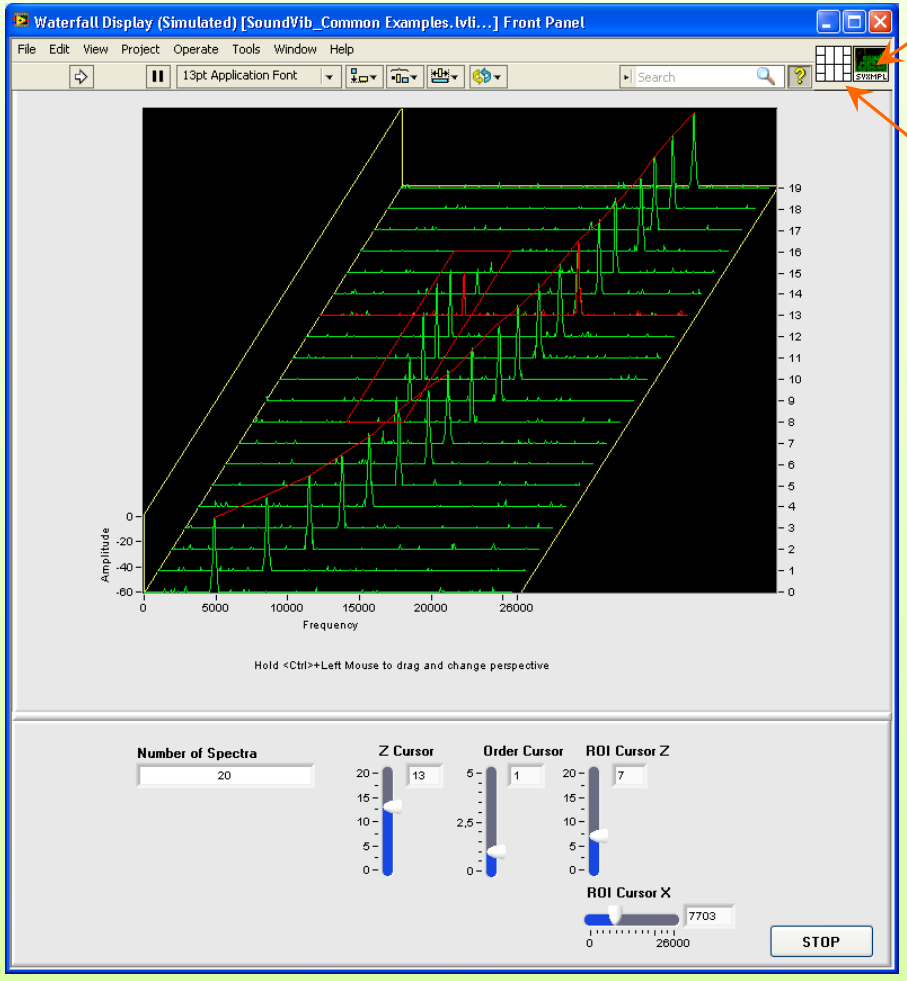
Sélectionne et attribue une couleur

4. L'environnement de développement

4.2. Les composantes d'un instrument virtuel (VI)

La face-avant

Elle représente l'interface Homme/Machine et intègre notamment l'icône du VI et ses connecteurs. Ces deux derniers éléments constituent la boite noire qui permettra d'utiliser le VI et tant que sous VI d'un autre VI.



Icône (représentation graphique du VI)

Connecteur (mappage des entrées et sorties du VI)

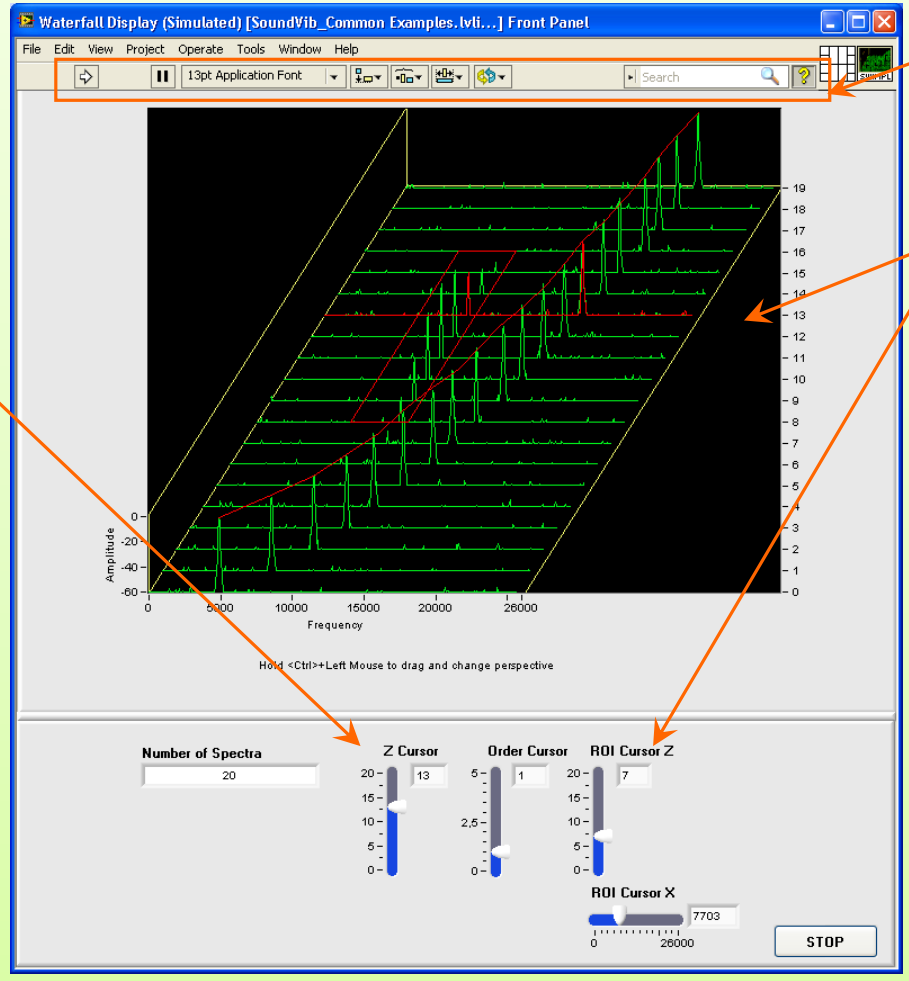
4. L'environnement de développement

4.2. Les composantes d'un instrument virtuel (VI)

La face-avant (suite)

Commandes

- Périphériques d'entrée
- Boutons rotatifs et glissières
- Fournissent les données au diagramme



Barre d'outils

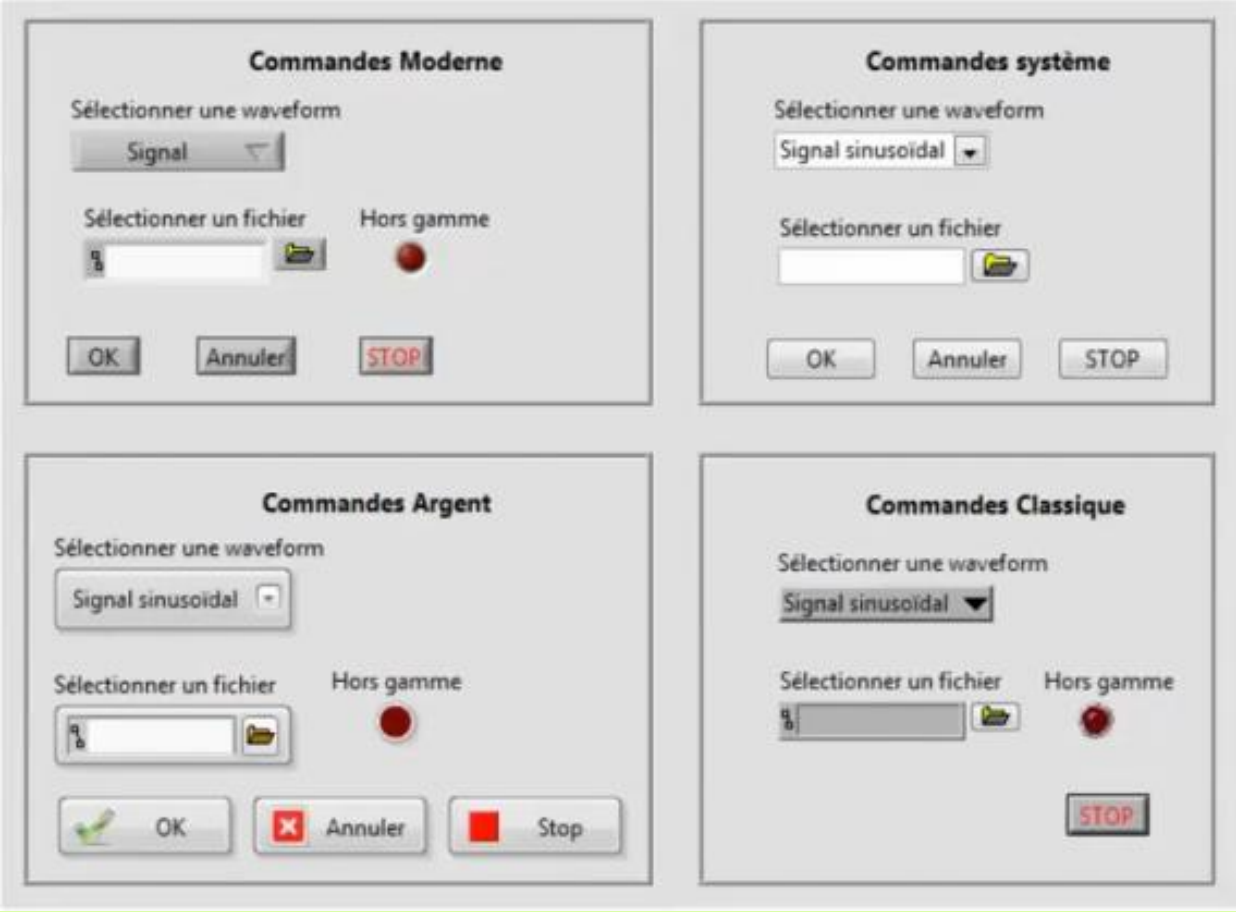
Indicateurs

- Périphériques de sortie
- Graphes, leds
- Affichent les données acquises ou générées par le diagramme

4.2. Les composantes d'un instrument virtuel (VI)

La face-avant (suite)

Les objets de la face avant sont disponibles dans 4 styles différents



4. L'environnement de développement

4.2. Les composantes d'un instrument virtuel (VI)

Le diagramme

Il représente le code du VI.

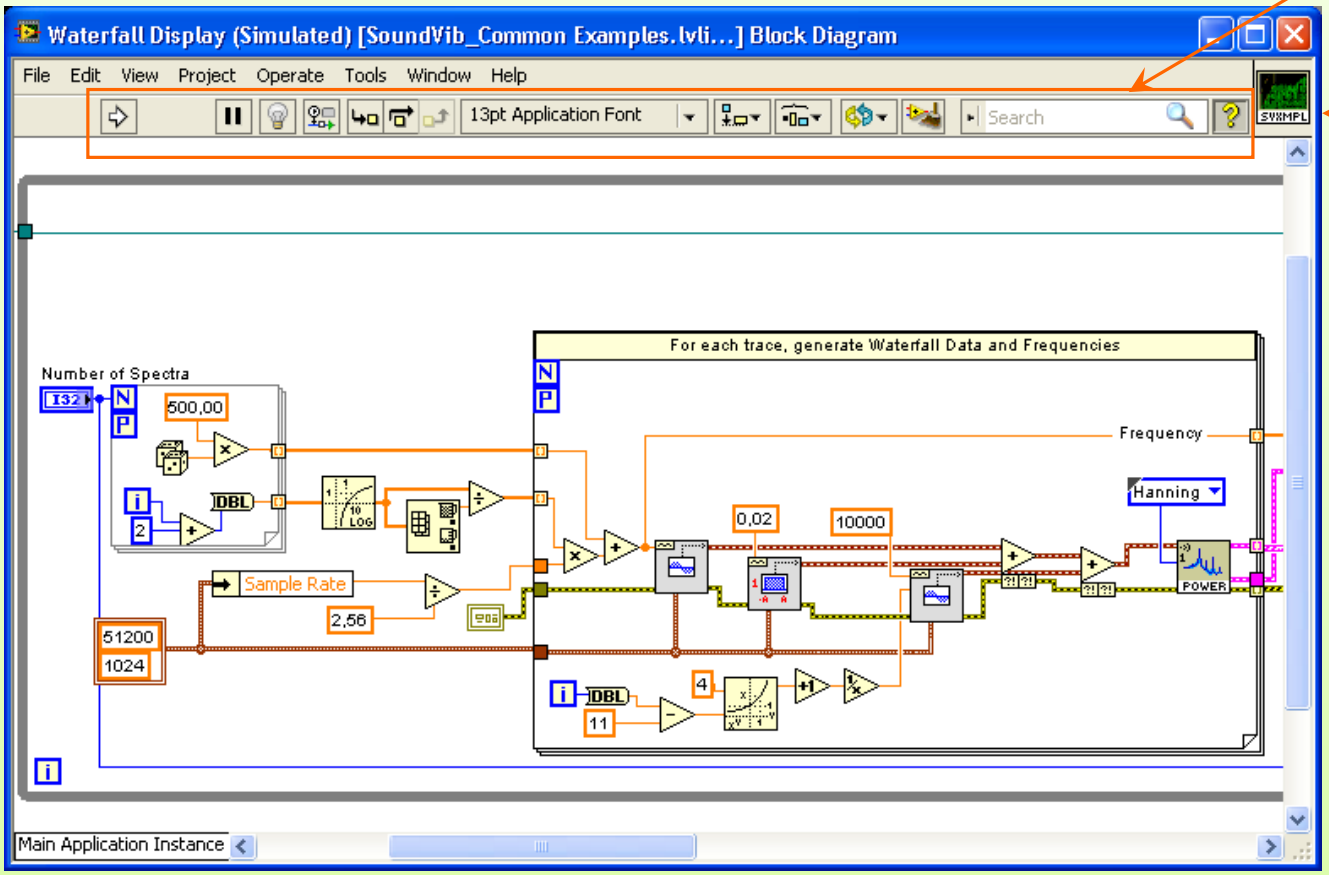


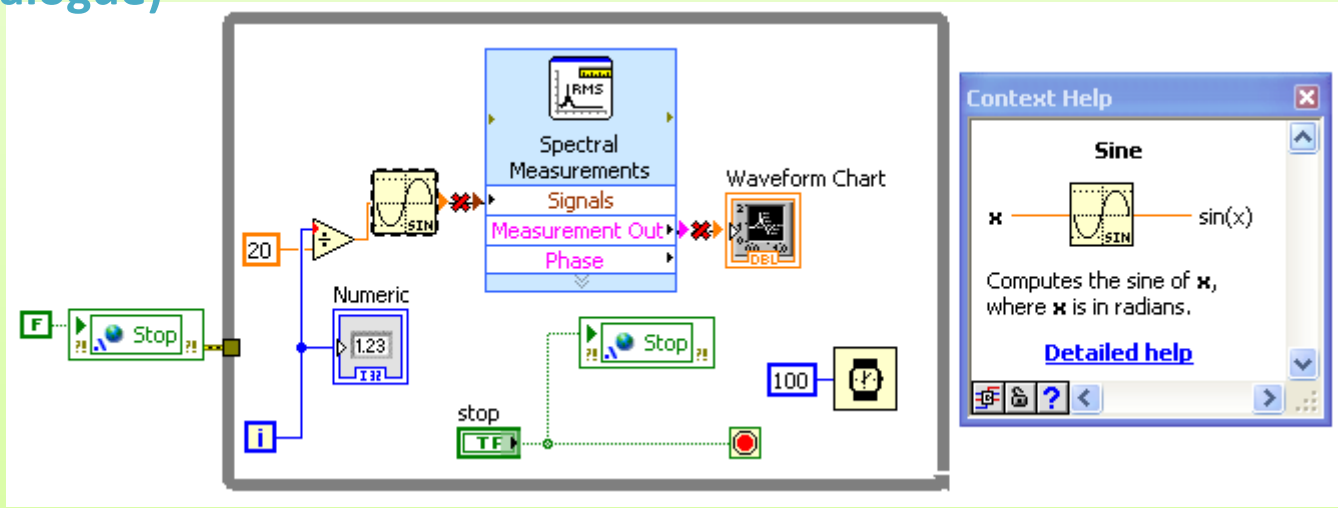
Diagramme ⟵ Ctrl+E ⟶ Face-Avant

4. L'environnement de développement

4.2. Les composantes d'un instrument virtuel (VI)

Les éléments du diagramme

- ✓ **Terminaux : ils représentent les entrées et les sorties dans le diagramme (représentation sur le diagramme d'un objet de la face-avant)**
- ✓ **Nœuds**
  - ➢ **Fonction (icône avec arrière plan jaune clair)**
  - ➢ **Sous-VI (possède une face avant et un diagramme)**
  - ➢ **VI Express (boite de dialogue)**
  - ➢ **Structure**
- ✓ **Fils de liaison (Flux de données)**
- ✓ **Etiquettes libres**
- ✓ **Aide contextuelle**



Micro-Travaux Pratiques dirigés

II. Création d'un VI vide

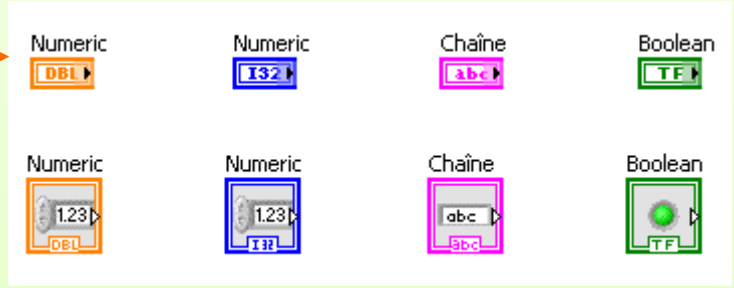
- Le nommer
- Insérer une commande sur la face avant
- Insérer un indicateur sur la face avant
- Insère une boucle while
- Créer le bouton stop
- Créer les liens
- Exécute le VI

4. L'environnement de développement

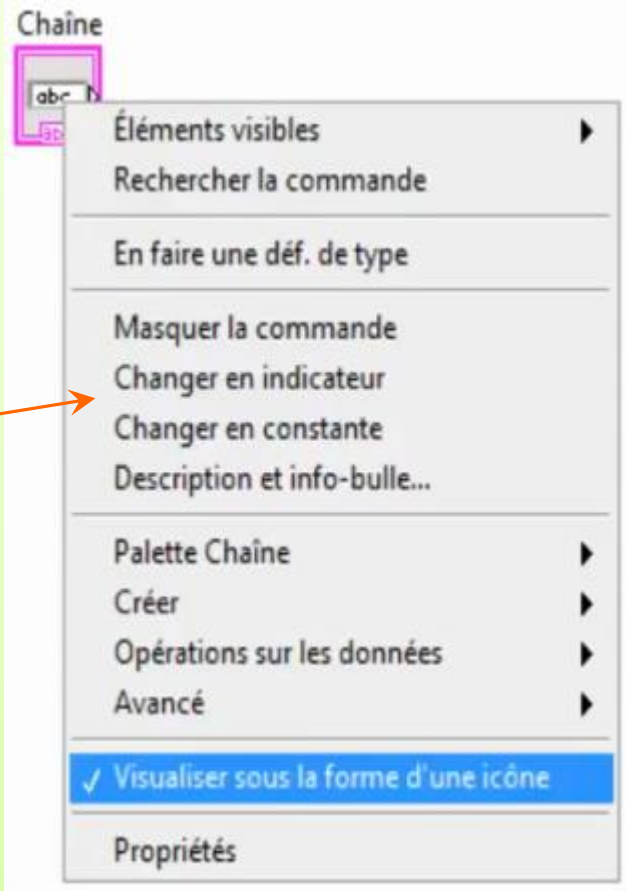
4.2. Les composantes d'un instrument virtuel (VI)

Les terminaux

Visualisation des terminaux sous-forme d'icône



Possibilité de changer une commande en un indicateur ou une constante avec le menu contextuel



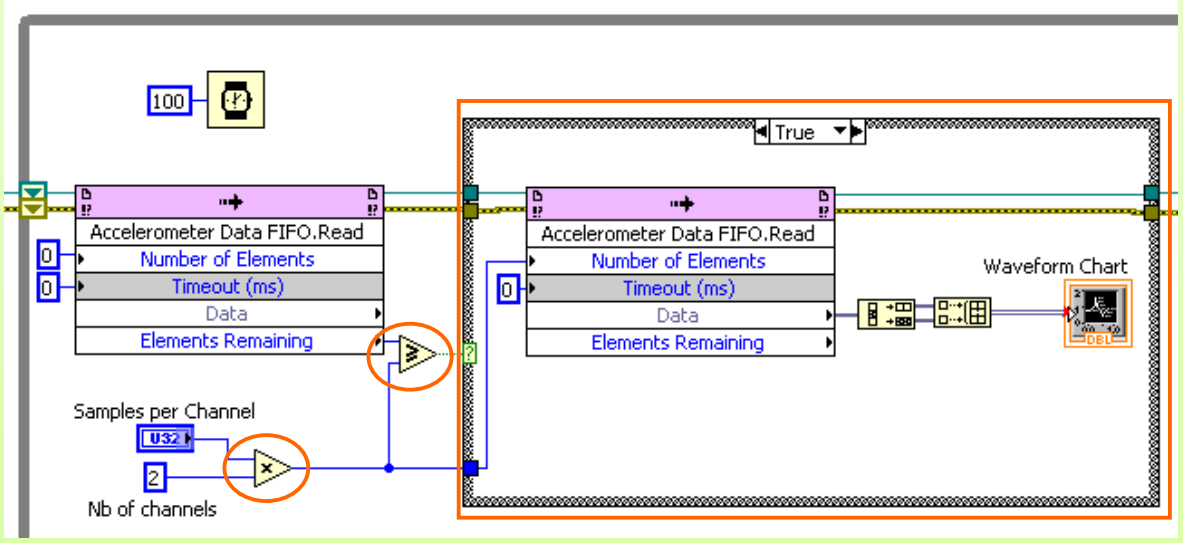
4.2. Les composantes d'un instrument virtuel (VI)

Les noeuds

Objets du diagramme, comportant des entrées et ou des sorties, et qui réalisent des « opérations » lorsque le VI s'exécute

Il existe des nœuds de fonction (icône avec arrière plan jaune clair), des nœuds de sous-VI ou de VI-Express (un double clic sur son icône a pour effet d'ouvrir un diagramme ou une fenêtre de paramétrage) et des nœuds de structure comme la structure conditionnelle de la portion de diagramme ci-dessous.

Le nœud de fonction « Supérieur ou égal » ne peut s'exécuter que lorsque ces deux entrées seront valides et donc aussi après l'exécution du nœud « multiplier » on met ainsi en évidente le concept de flux de données ...



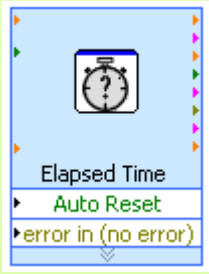
4. L'environnement de développement

4.2. Les composantes d'un instrument virtuel (VI)

Les nœuds de sous-VI

- ✓ VI utilisé dans le diagramme d'un autre VI
- ✓ Dispose d'une face-avant et d'un diagramme
- ✓ L'icône en haut et à droite de la face-avant est celle qui apparaît sur le diagramme
- ✓ Un double clic sur l'icône ouvre la face-avant et le diagramme du VI
- ✓ N'importe quel VI peut être utilisé comme sous-VI (vérifier toutefois les propriétés d'exécution du sous-VI…)

Les nœuds de VI-Express



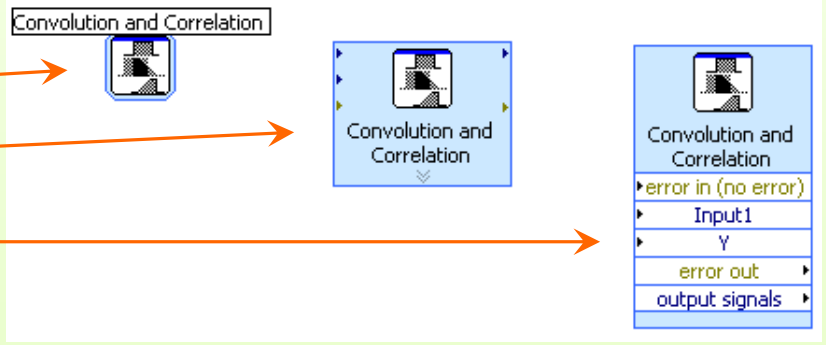
- ✓ Type spécial de VI (Fonction évoluée et paramétrage)
- ✓ Câblage minimaliste
- ✓ La configuration s'opère au travers d'une boite de dialogue
- ✓ L'icône dispose d'un arrière plan bleu

4. L'environnement de développement

4.2. Les composantes d'un instrument virtuel (VI)

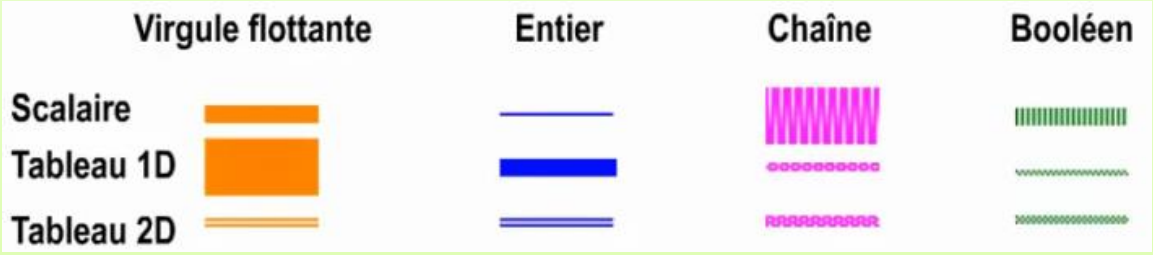
Les options d'affichage des nœuds

- ✓ Affichage simple
- ✓ Sous forme d'icône
- ✓ Icône avec visualisation des entrées et des sorties ainsi que leur mnémonique

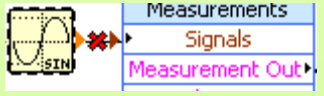


Les fils de liaison

- ✓ Transfert unidirectionnel des données entre les objets du diagramme
- ✓ Leur couleur, style et épaisseur est fonction du type de données transférées

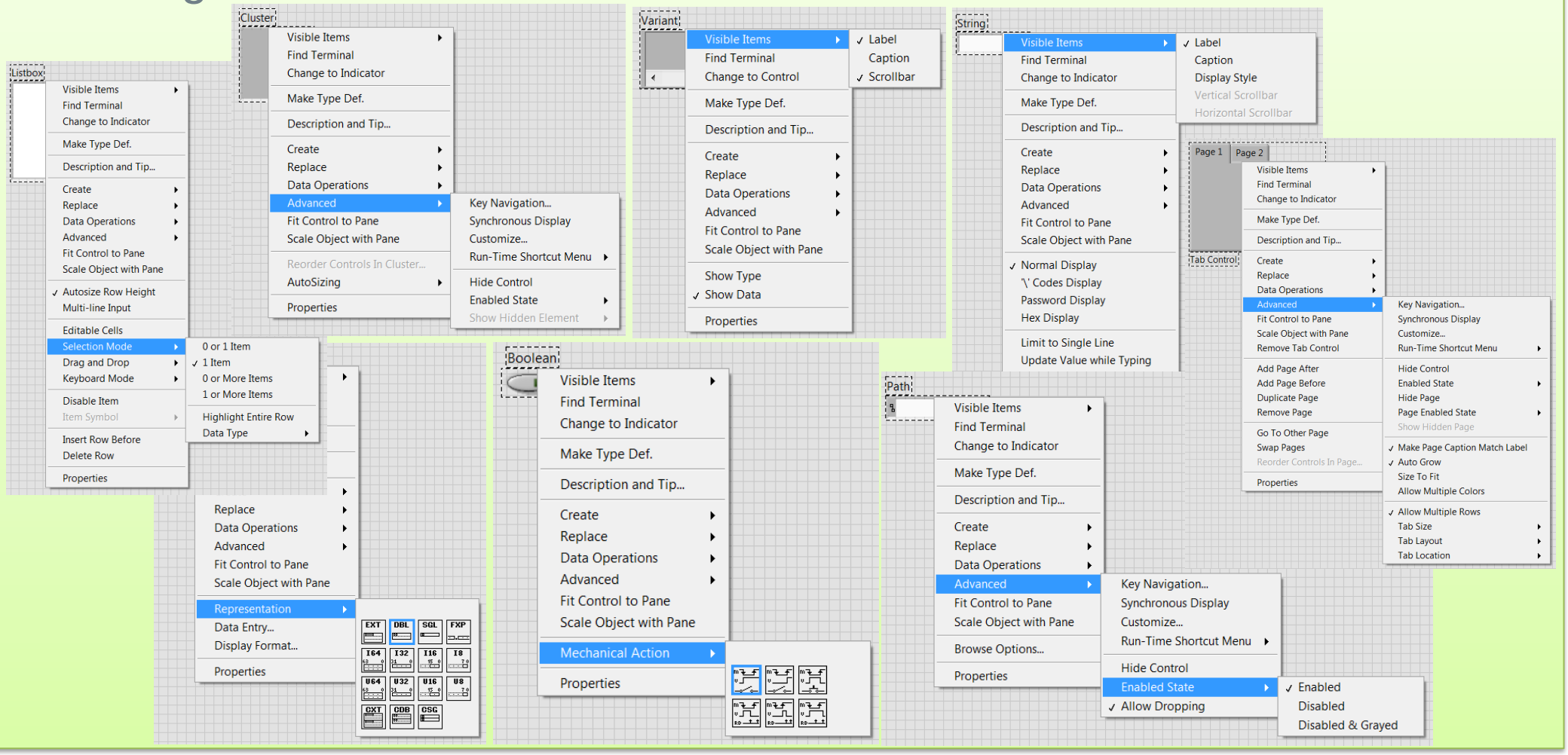


- ✓ Un fil de liaison brisé apparaît comme un trait en pointillés avec le symbole d'une croix rouge à l'intersection : ce fil indique une discordance de type de données entre les deux objets reliés



4. L'environnement de développement

4.3. La grande diversité des menus contextuels



4. L'environnement de développement

4.4. Les 3 niveaux de présentation de l'aide

L'aide contextuelle

- ✓ Apparaît lorsque que l'on place le curseur sur un fil de liaison ou un nœud et affiches des informations succinctes
- ✓ La fenêtre de l'aide contextuelle peut être affichée/masquée soit en utilisant l'icône de la barre d'outils, soit en appuyant sur <Ctrl+H>, soit en sélectionnant **Aide>>Afficher l'aide contextuelle** dans le menu LabVIEW.

L'aide LabVIEW

- ✓ Fichier à l'extension .chm qui contient des descriptions détaillées et des instructions pour la plupart des palettes, menus, outils, Vis et fonctions
- ✓ Cette aide est accessible :
  - ➢ en sélectionnant **Aide>>Aide LabVIEW** dans le menu de la fenêtre du diagramme ou de la face-avant ;
  - ➢ en utilisant le lien **Aide détaillée** de la fenêtre de l'aide contextuelle ;
  - ➢ en sélectionnant **Aide** du menu contextuel d'un objet.

4. L'environnement de développement

4.4. Les 3 niveaux de présentation de l'aide (suite)

L'aide à travers la recherche d'exemples

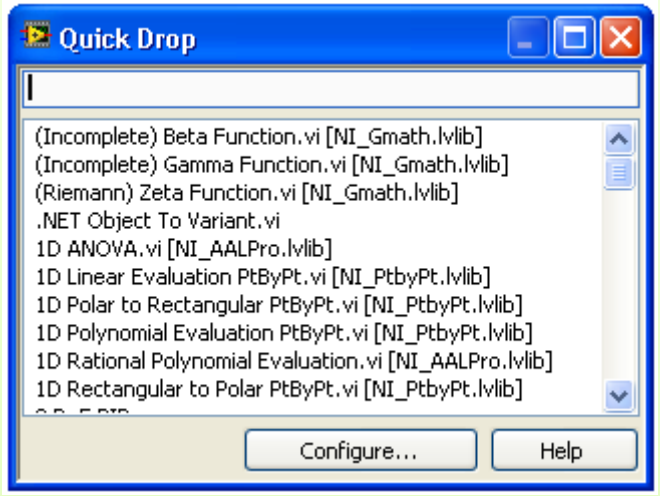
- ✓ Via **Aide>>Recherche d'exemples**
- ✓ Via l'aide détaillée grâce aux icônes

4. L'environnement de développement

4.5. Recherche de commandes, de VI et de fonctions

Recherche et navigation dans les palettes

- ✓ Palette des commandes dans la face avant
- ✓ Palette des fonctions dans le diagramme

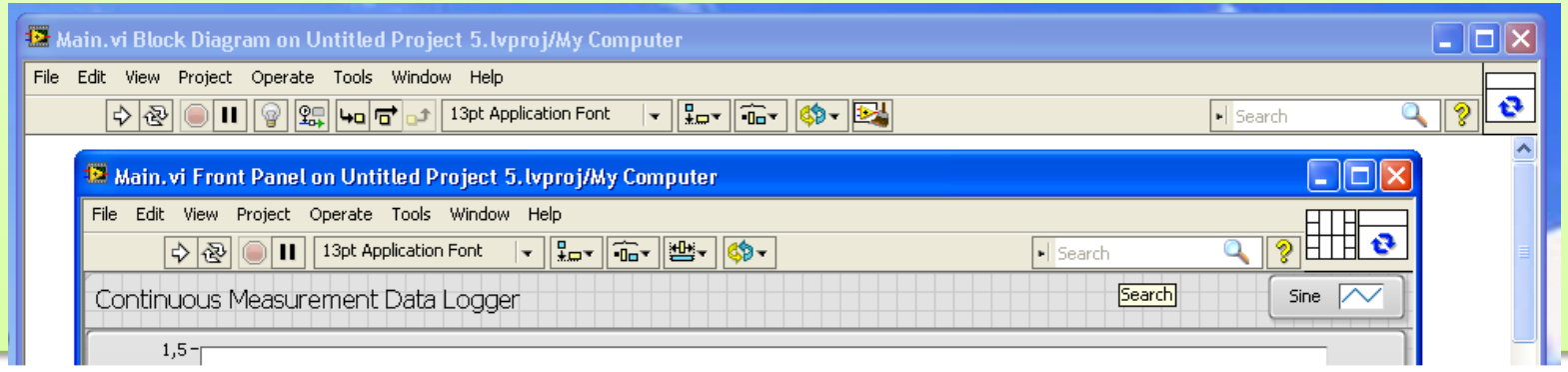


Placement rapide

- ✓ Ctrl + barre espace
- ✓ Recherche sur le nom de l'objet à intégrer

Recherche globale NI

- ✓ Permet d'effectuer des recherches dans les palettes, dans l'Aide LabVIEW et sur le site web ni.com

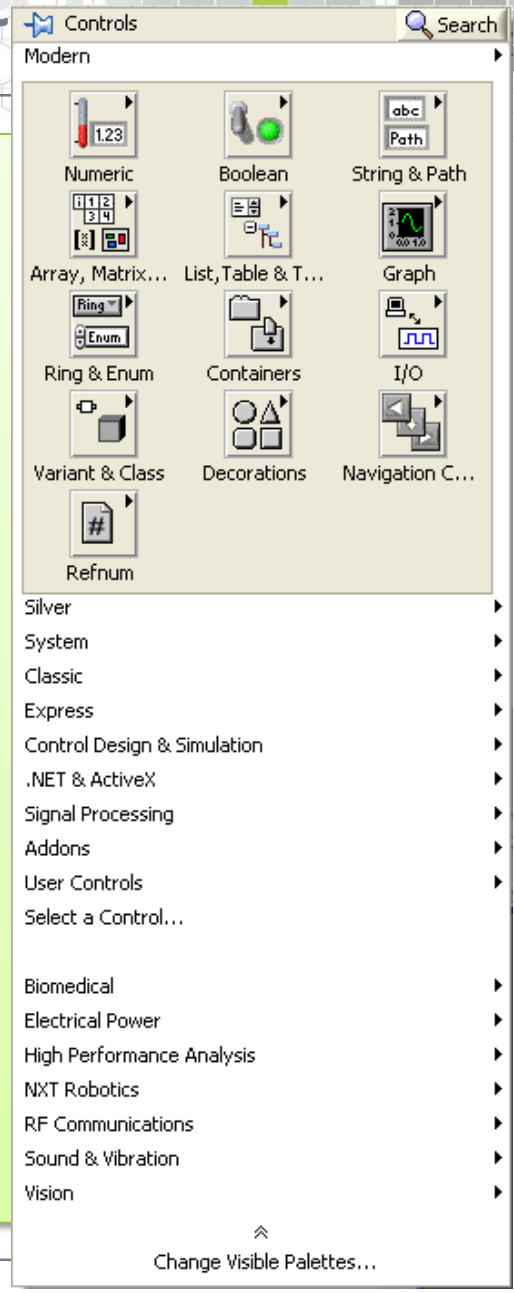


4. L'environnement de développement

4.5. Recherche de commandes, de VI et de fonctions

Palette de commandes (spécifique à la face-avant)

- ✓ Contient les commandes et les indicateurs utilisables pour la création de la face-avant
- ✓ Permet la navigation dans les sous-palettes
- ✓ Donne accès à la fonctionnalité rechercher à l'image de la recherche rapide

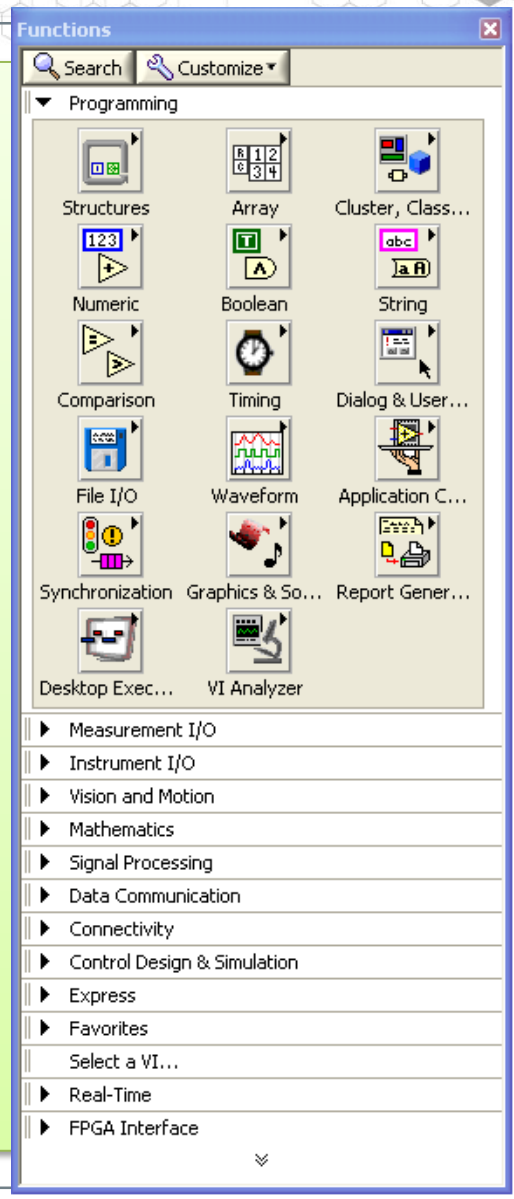


4. L'environnement de développement

4.5. Recherche de commandes, de VI et de fonctions

Palette de fonctions (spécifique au diagramme)

- ✓ Contient les Vis, fonctions et constantes utilisables pour créer le diagramme
- ✓ Permet de naviguer dans les sous-palettes
- ✓ Donne accès à la fonctionnalité **Rechercher**

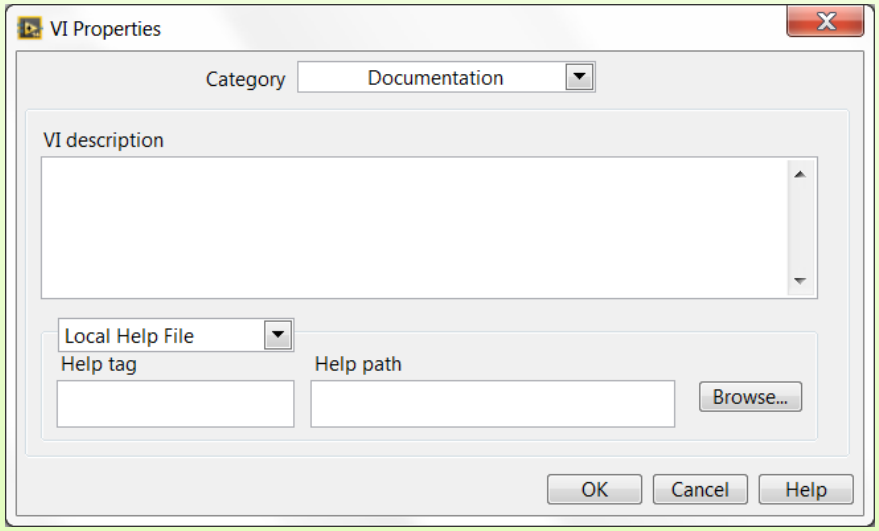


4. L'environnement de développement

4.6 La documentation : comment renseigner son code ?

Au niveau du VI : son nom et sa description

L'onglet « Fichier », puis Propriété du VI permet d'accéder à la description de ce dernier.



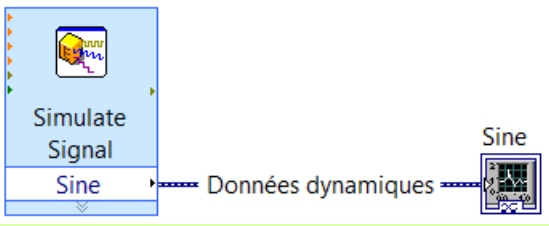
D'autre part, que ce soit au niveau de la face–avant comme du diagramme, des étiquettes libres peuvent être ajoutées en utilisant l'outil « édition de texte » de la palette d'outils.

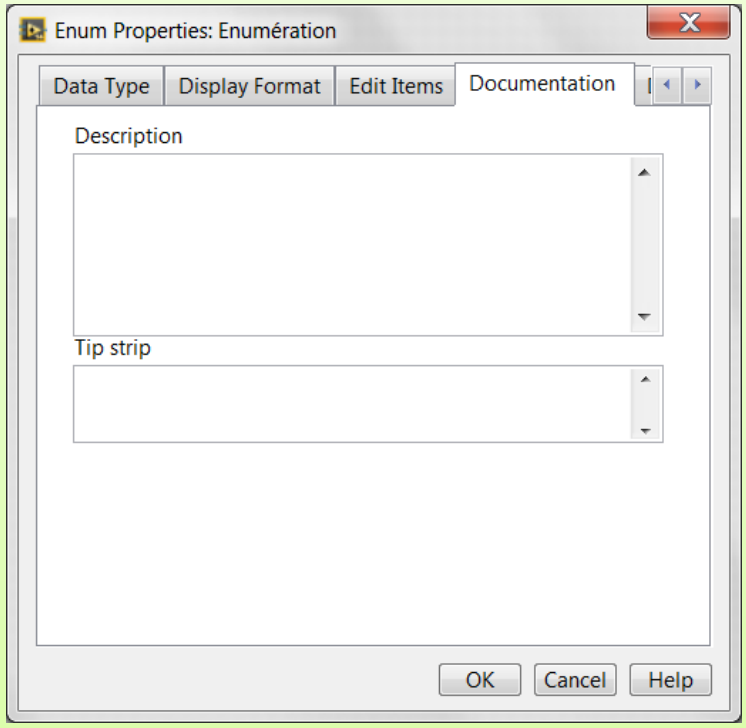
4. L'environnement de développement

4.6 La documentation : comment renseigner son code ?

Pour chaque objet, la fenêtre de propriétés donne accès à l'affichage d'une étiquette, et aussi à un onglet « Documentation », qui permet de renseigner une description et une info-bulle.

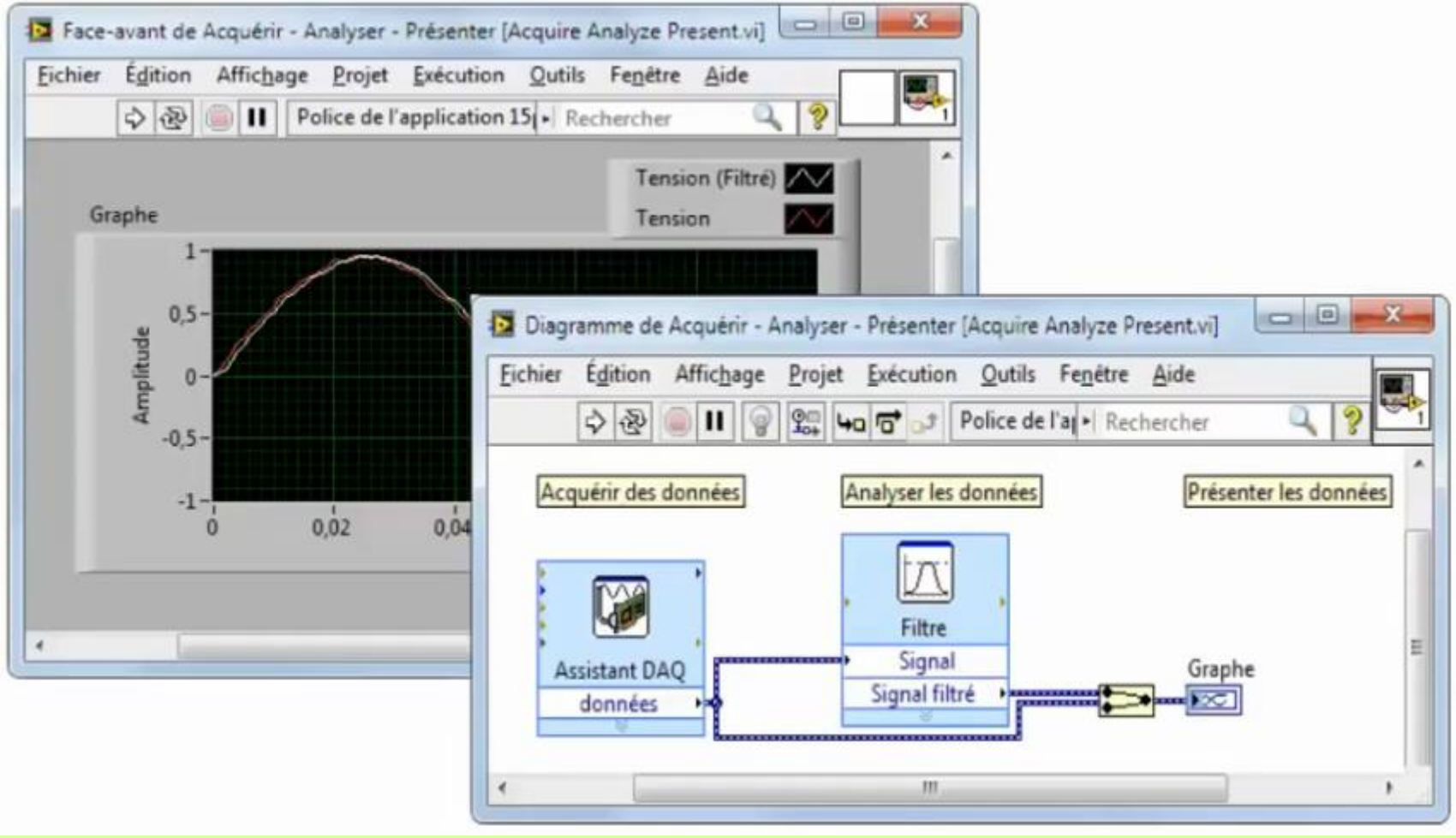
Au niveau du diagramme, les étiquettes liées peuvent être créées sur les terminaux, y compris les fils : il suffit pour cela d'utiliser le menu contextuel et de valider l'option « Eléments visibles/étiquette ».





5. Construction d’un VI simple

Objectif : acquisition, traitement et affichage avec les VI Express



5. Construction d'un VI simple

5.1. VI Express d'acquisition

VI Express Assistant DAQ

VI Express Assistant d'E/S instrument

VI Express Simuler un signal

VI Express Lire un fichier de mesure

5. Construction d'un VI simple

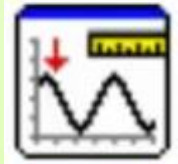
5.2. VI Express d'analyse

VI Express Mesures d'amplitudes et de niveaux

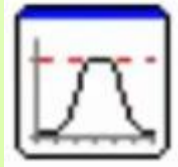
VI Express Statistiques

VI Express Mesures spectrales

VI Express Mesures de tons



VI Express Filtres



5. Construction d'un VI simple

5.3. VI Express de présentation de résultats

VI Express Afficher un message à l'utilisateur

VI Express Jouer une waveform

VI Express Rapport

VI Express Ecrire dans un fichier de mesures

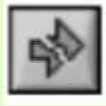
VI Express Rapport DIAdem

5. Construction d'un VI simple

5.4. Mode opératoire

1) **Placer les VI Express sur le diagramme**
2) **Configurer chacune des boites de dialogue**
3) **Câbler les VI Express les uns aux autres**
4) **Enregistrer le VI et exécuter-le**

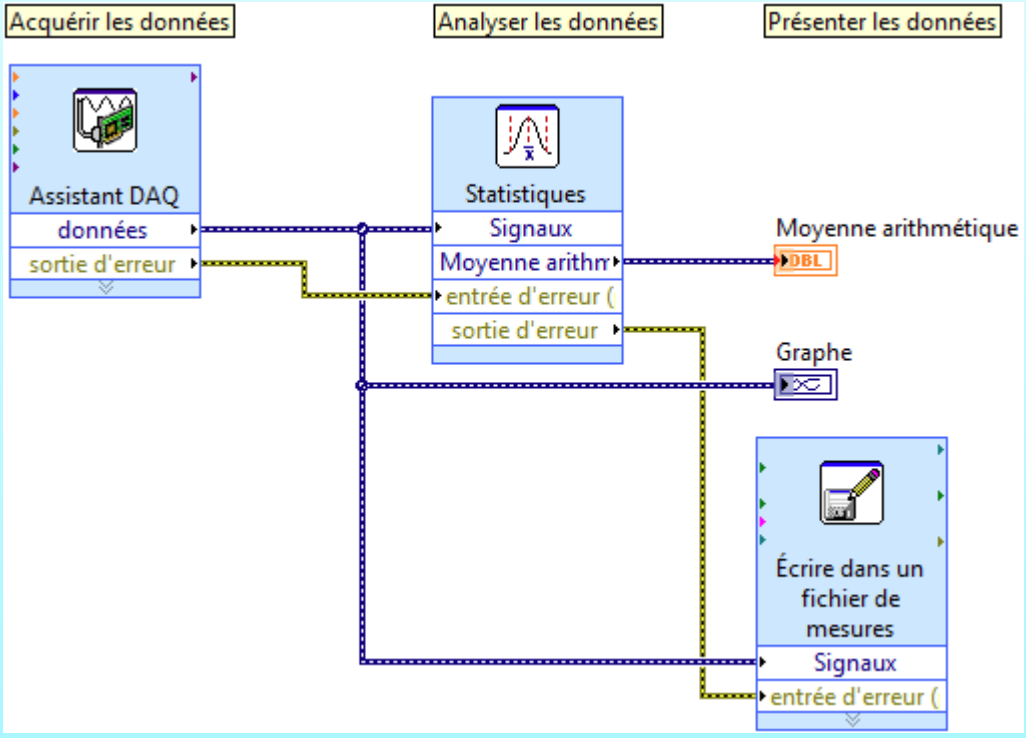
En cas d'erreur, le bouton *Exécuter* apparait brisé …



Micro-Travaux Pratiques dirigés

III. Création d'un VI simple avec VI Express

- Insérer sur le diagramme …
- Paramétrer chaque VI Express
- Créer les liens
- Exécuter le VI



6. Eléments de test et de « débuggage »

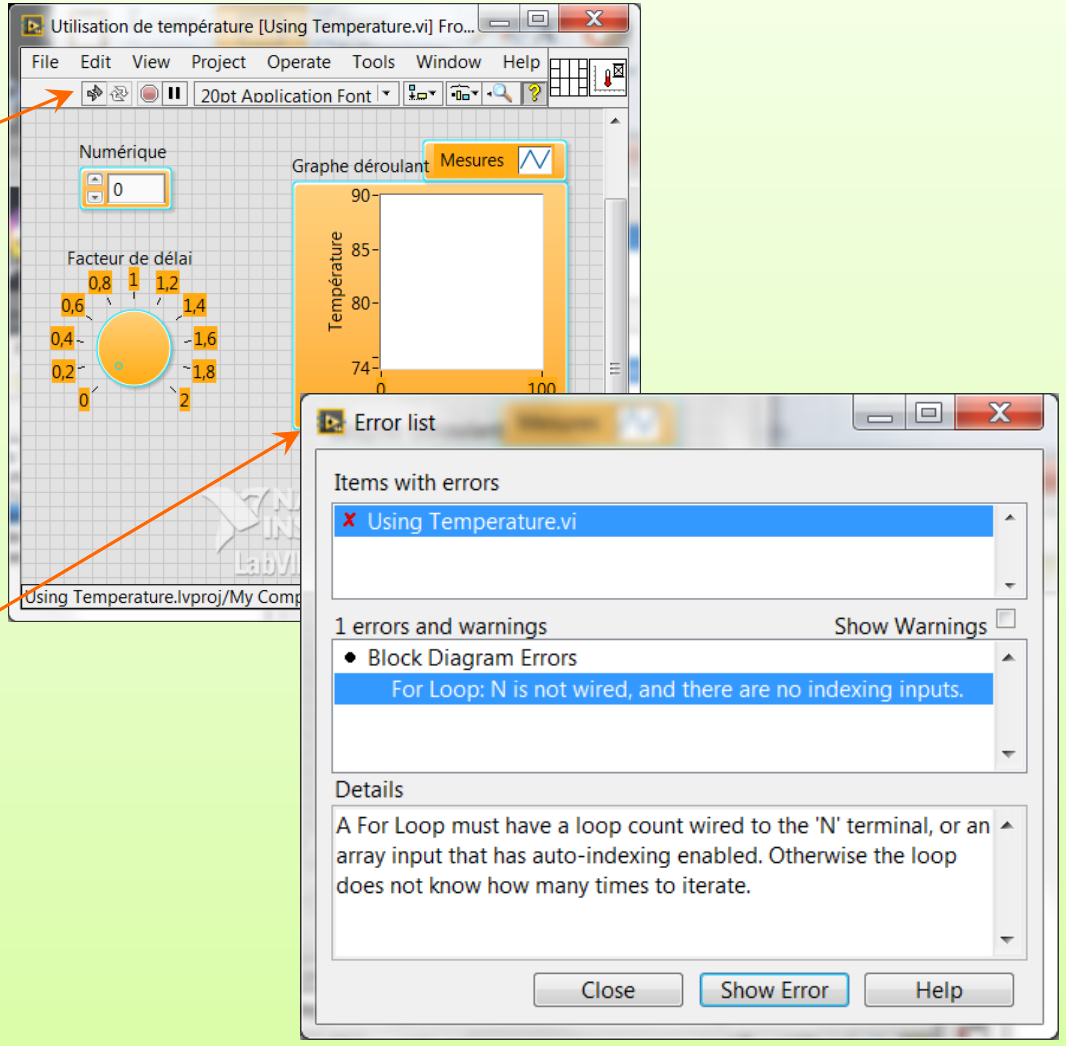
6.1. Correction des VIs brisés

Flèche d'exécution brisée

=> le VI ne peut pas être compilé
=> le VI ne peut pas être exécuté

Pour analyser la liste des erreurs

- **accès clic sur flèche d'exécution brisée**



6. Eléments de test et de « débuggage »

6.2. Causes fréquentes de VIs brisés

- **Des fils de liaison brisés existent sur le diagramme**
- **Vous avez câblé une commande chaine à un indicateur numérique**
- **Vous avez câblé une commande booléenne à une commande booléenne**
- **Un terminal nécessaire n'est pas câblé sur le diagramme**
- **Un sous-VI est lui-même brisé**

6. Eléments de test et de « débuggage »

6.3. Techniques de mise au point

Quelles questions doit-on se poser ?

- **Existe-t-il des sous-VIs non câblés ou masqués ?**
- **Les données par défaut sont-elles correctes ?**
- **Le VI transmet-il des données non définies ?**
- **Les représentations des commandes numériques sont-elles correctes ?**
- **Les nœuds sont-ils exécutés dans le bon ordre ? (respect de la notion de flux de données)**

6. Eléments de test et de « débuggage »

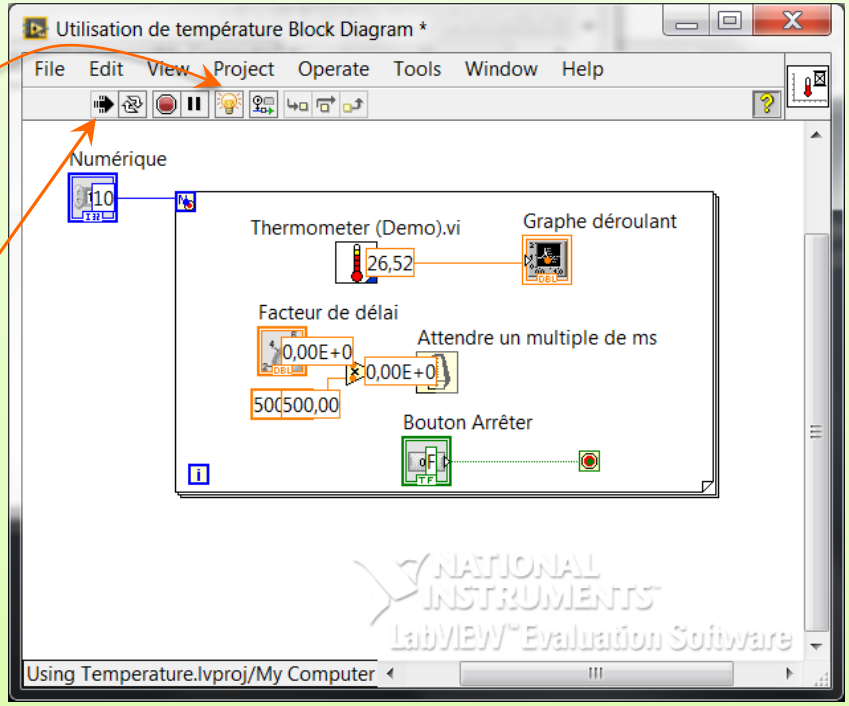
6.4. Animation de l'exécution

- **Elle sert à visualiser le flux de données (d'informations) sur le diagramme**

Sélectionnez l'icône d'animation

Puis

Sélectionnez l'icône d'exécution

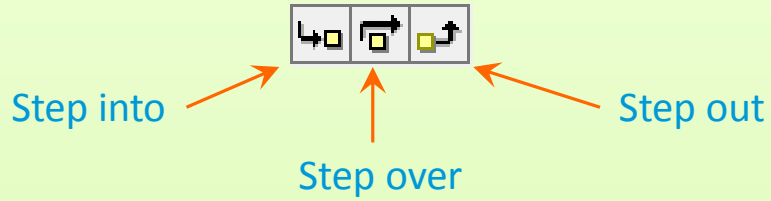


Si votre VI s'exécute plus lentement que prévu, vérifiez que l'icône d'animation est désactivée dans le sous-VI de votre diagramme maître …

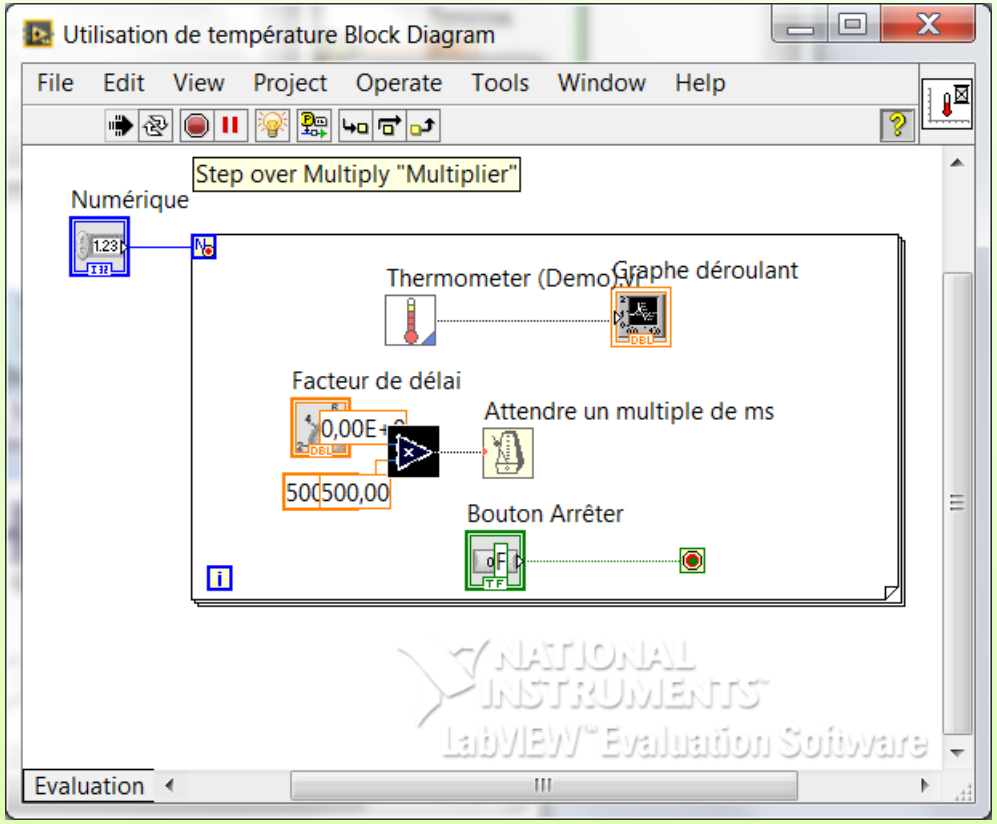
6. Eléments de test et de « débuggage »

6.5. Mode pas à pas

Ce mode permet de suivre l'exécution de chacune des "instructions" du code contenu dans le diagramme.



Il est alors possible d'éditer les valeurs de commandes, visualiser les valeurs des indicateurs, contrôler le nombre d'exécution ou revenir au début de l'exécution d'un sous-VI.



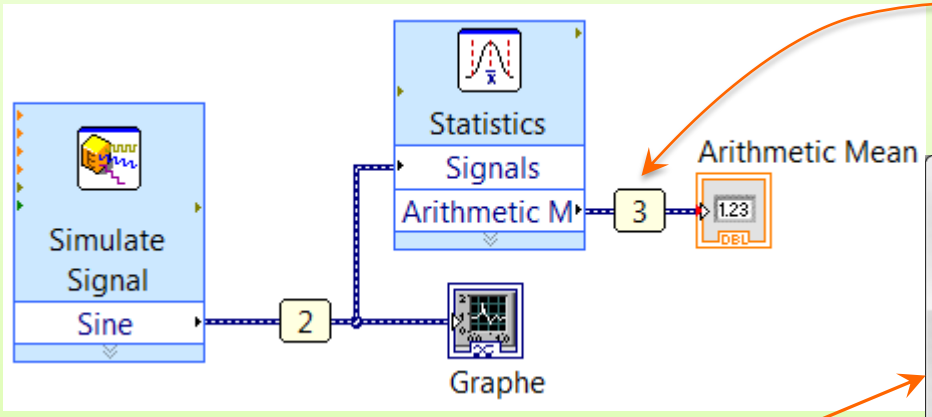
L'option *Suspendre lors d'un appel* est disponible pour chaque sous-VI dans l'onglet Exécution

6. Eléments de test et de « débuggage »

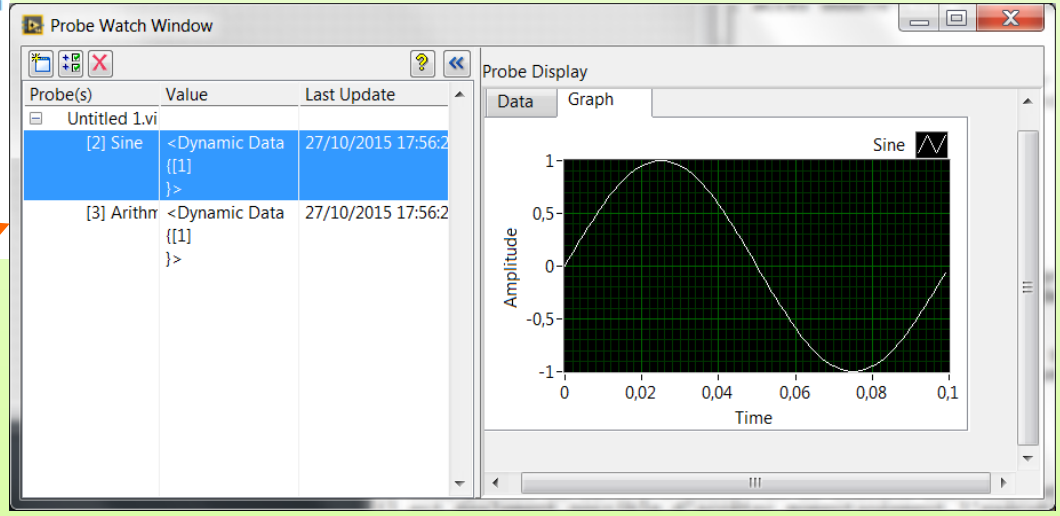
6.6. Sondes

Utilisez l'outil Sonde pour observer les valeurs de données intermédiaires et pour vérifier la sortie d'erreur des VIs et des fonctions, en particulier des opérations d'E/S.

Insère une sonde sur le diagramme



Visualisation des valeurs de sondes dans la fenêtre *Probe Watch*



6. Eléments de test et de « débuggage »

6.7. Points d'arrêt

Lorsque l'exécution atteint un point d'arrêt, le VI se met en pause et le bouton *Pause* passe au rouge.

Insère un point d'arrêt dans le diagramme

Vous pouvez alors :

- **Exécuter le VI en mode pas à pas,**
- **Sonder les fils de liaison pour vérifier des valeurs intermédiaires,**
- **Modifier les valeurs de commandes de la face-avant,**
- **Cliquer sur le bouton Pause pour exécuter le VI jusqu'au prochain point d'arrêt ou la fin du VI.**

6. Eléments de test et de « débuggage »

6.8. Données non valides

Les calculs mathématiques peuvent faire apparaitre ce type de données.

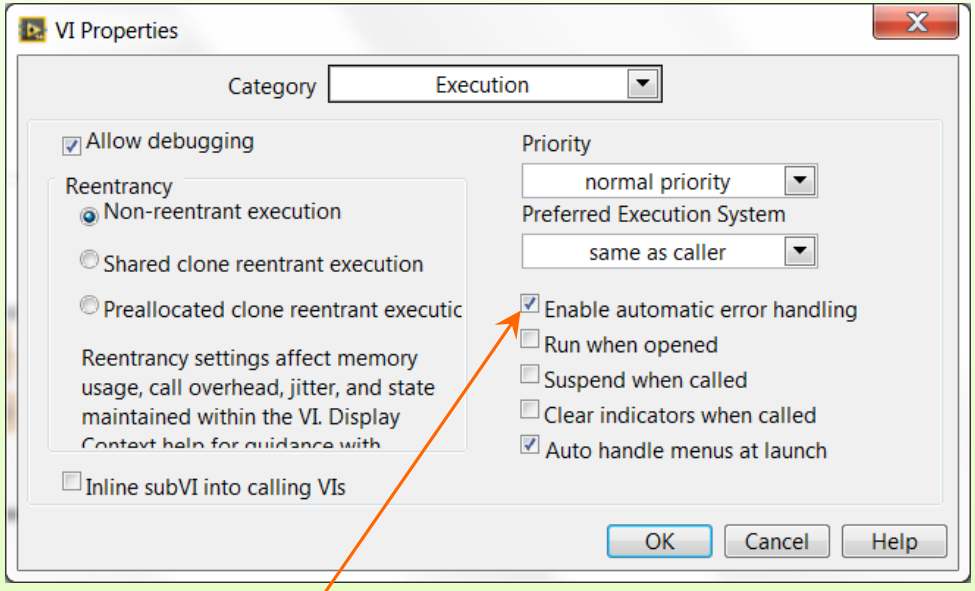
Deux cas de figure peuvent se présenter :

- **La valeur infinie, résultat potentiel d'une division par 0**
- **La valeur indéfinie *NaN*, ou Not a Number, n'est donc pas un nombre, comme dans l'exemple du calcul de la racine carrée d'un nombre réel négatif.**

6. Eléments de test et de « débuggage »

6.9. Gestion d'erreurs automatique

Par défaut cette option est validée dans la catégorie *Execution* des propriétés du VI en question. Lorsqu'une erreur survient, elle est automatiquement détectée, LabVIEW suspend l'exécution du VI, met en surbrillance le sous VI ou la fonction à l'origine de l'erreur, et affiche la fenêtre *Liste des erreurs*.



Sélection du mode automatique

L'inconvénient de ce mode de gestion d'erreurs est qu'il disparait des exécutables créés à partir de LabVIEW : on lui préfèrera la gestion manuelle, également souhaitable pour les applications professionnelles …

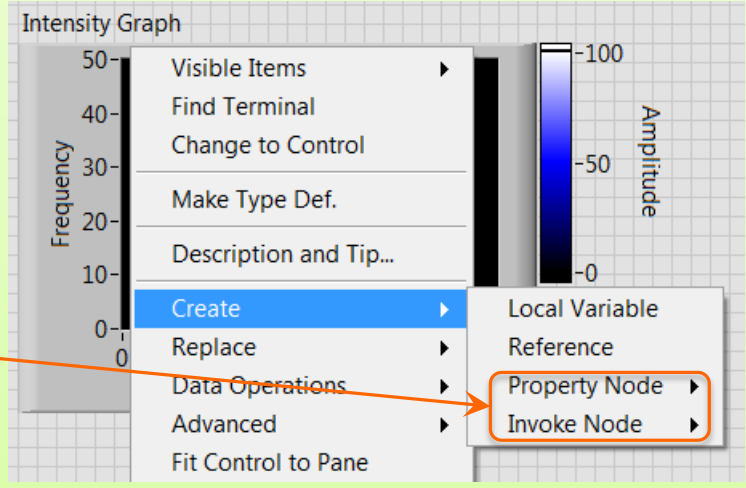
7. Types de données

Généralités

Tous les objets LabVIEW disposent d'un menu contextuel qui donne accès rapidement aux propriétés usuelles de l'objet en question.

Ce menu contextuel permet également d'accéder aux mêmes propriétés, et à d'autres en passant par la boite de dialogue, disponible à la fin du menu déroulant.

Enfin, d'autres propriétés sont modifiables et d'autres actions sont réalisables au travers des nœuds de propriétés et nœuds de méthodes …



7. Types de données

Notion de variable

Une variable peut être définie comme un élément du diagramme qui permet d'accéder à des données ou de les enregistrer. Elle peut prendre 4 formes distinctes :

- ✓ **Locale :** elle stocke les données dans des commandes ou des indicateurs de la face-avant.
- ✓ **Globale :** elle stocke les données dans des répertoires spéciaux accessibles à partir de plusieurs VIs.
- ✓ **Globale fonctionnelle (FGV) :** elle stocke les données dans les registres à décalage d'une boucle While.
- ✓ **Partagée :** elle transfère les données entre diverses cibles distribuées et connectées par un réseau.

Micro-Travaux Pratiques dirigés

IV. Test variable locale

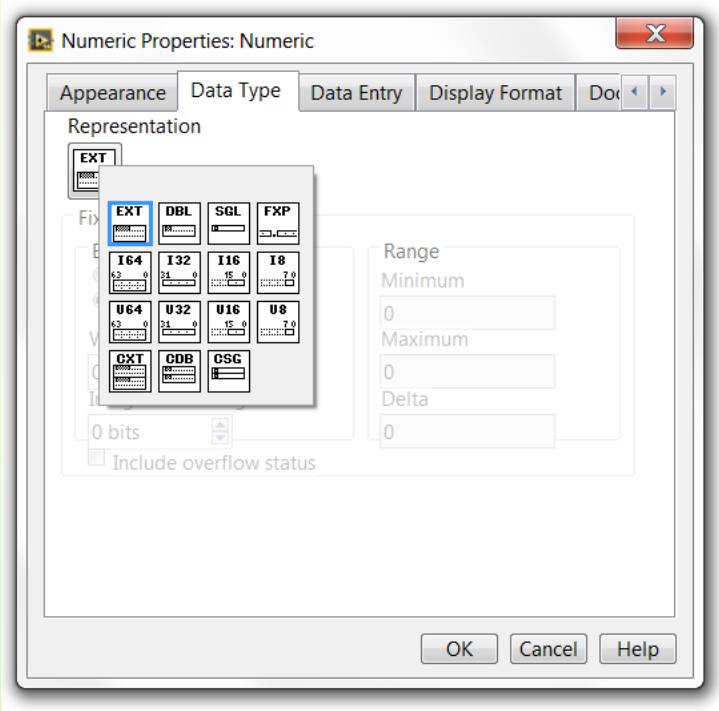
- Utilisation avec parcimonie, puisqu'elle rompt le flux de données …
- Utiliser la variable locale d'un contrôle numérique pour l'initialiser dans une séquence, avec le contrôle et son indicateur dans une boucle While.

7. Types de données

7.1. Numériques

Les nombres sont formatés selon une certaine représentation. LabVIEW offre 15 représentations différentes :

- ✓ « Fixed point » (sur FPGA …)
- ✓ Flottant en simple, double ou précision étendue
- ✓ Entier signé ou non-signé sur 8, 16, 32 ou 64 bits
- ✓ Complexe en simple, double ou précision étendue

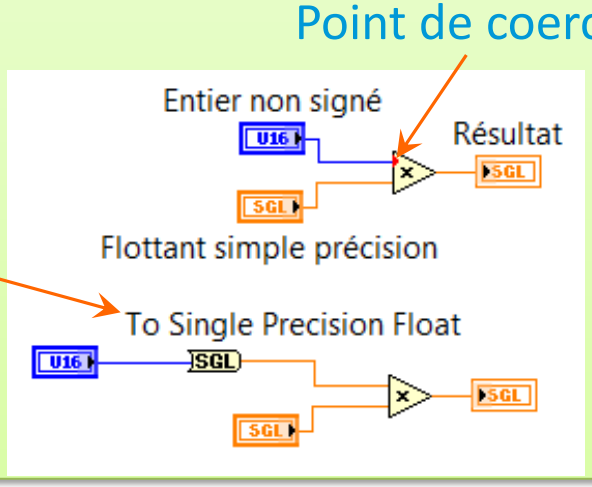


7. Types de données

7.1. Numériques (Suite)

La notion de *coercition* signifie que LabVIEW a converti la valeur transmise en une représentation différente de celle d'origine, et malheureusement souvent celle qui utilise le plus grand nombre de bit de représentation.

Une des règles de bonnes pratiques de programmation consiste, en l'occurrence, à s'affranchir de toute coercition, en effectuant une conversion par programmation.



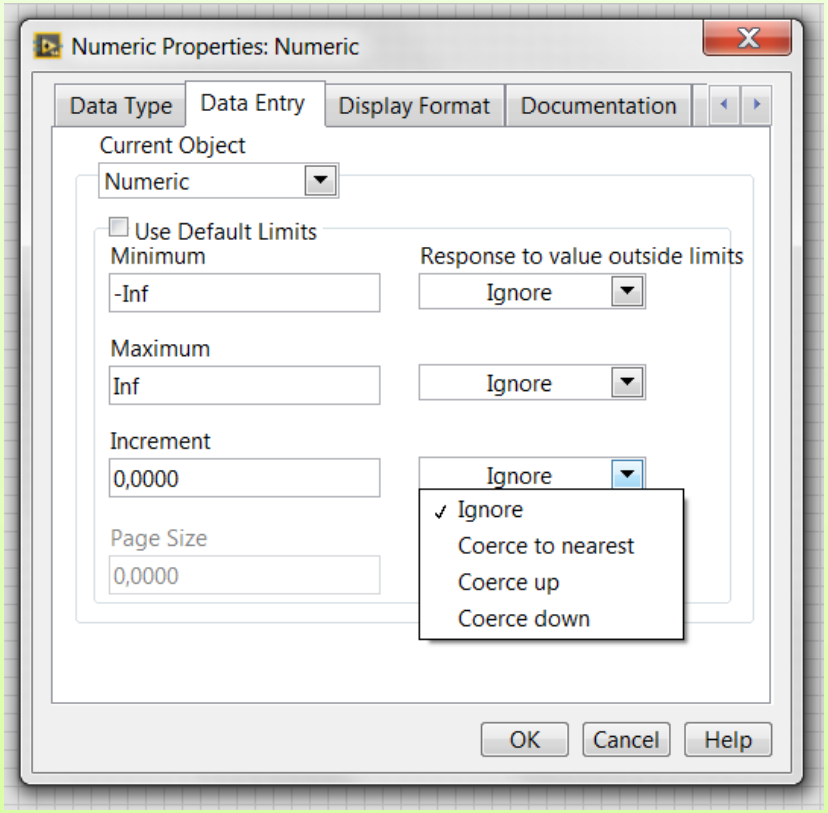
7. Types de données

7.1. Numériques (Suite)

LabVIEW permet également de verrouiller les contrôles numériques en imposant, si besoin, l'incrémentation et les limites de la variable numérique en question.

La boite de dialogue des propriétés permet ainsi de définir :

- ✓ Bornes inférieure et supérieure
- ✓ « Pas » d'incrémentation
- ✓ Réponse en cas de dépassement : ignorer ou contraindre …



7. Types de données

7.1. Numériques (Suite)

Table issue de l'aide LabVIEW

Numeric Data Types Table

The following table displays the numeric data types available in LabVIEW. LabVIEW stores each data type in a different way.

Terminal	Numeric Data Type	Bits of Storage on Disk	Approximate Number of Decimal Digits	Approximate Range
SGL	Single-precision, floating-point	32	6	Minimum positive number: 1.40e–45 Maximum positive number: 3.40e+38 Minimum negative number: –1.40e–45 Maximum negative number: –3.40e+38
DBL	Double-precision, floating-point	64	15	Minimum positive number: 4.94e–324 Maximum positive number: 1.79e+308 Minimum negative number: –4.94e–324 Maximum negative number: –1.79e+308
EXT	Extended-precision, floating-point	128	varies from 15 to 20 by platform	Minimum positive number: 6.48e–4966 Maximum positive number: 1.19e+4932 Minimum negative number: –6.48e–4966 Maximum negative number: –1.19e+4932
CSG	Complex single-precision, floating-point	64	6	Same as single-precision, floating-point for each (real and imaginary) part
CDB	Complex double-precision, floating-point	128	15	Same as double-precision, floating-point for each (real and imaginary) part
CXT	Complex extended-precision, floating-point	256	varies from 15 to 20 by platform	Same as extended-precision, floating-point for each (real and imaginary) part
FXP	Fixed-point	64, or 72 if you include an overflow status	varies by user configuration	varies by user configuration
I8	Byte signed integer	8	2	–128 to 127
I16	Word signed integer	16	4	–32,768 to 32,767
I32	Long signed integer	32	9	–2,147,483,648 to 2,147,483,647
I64	Quad signed integer	64	18	–1e19 to 1e19
U8	Byte unsigned integer	8	2	0 to 255
U16	Word unsigned integer	16	4	0 to 65,535
U32	Long unsigned integer	32	9	0 to 4,294,967,295
U64	Quad unsigned integer	64	19	0 to 2e19
TS	128-bit time stamp	128	19	Minimum time: 01/01/1600 00:00:00 UTC maximum time: 01/01/3001 00:00:00 UTC

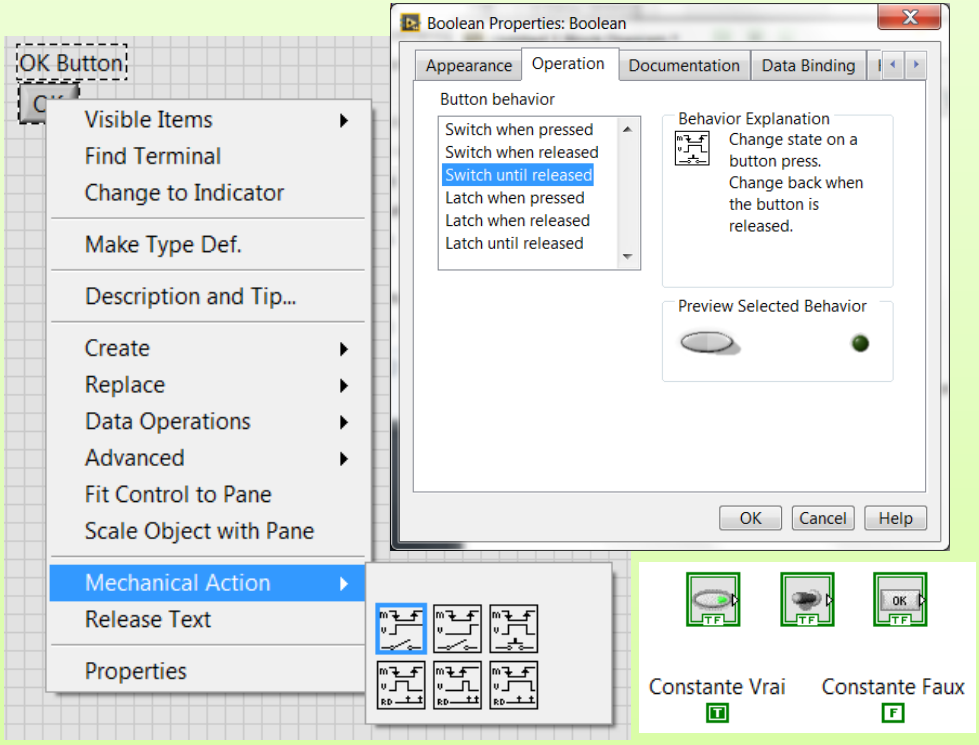
7. Types de données

7.2. Booléens

Quelle que soit la commande booléenne utilisée, la sélection du comportement de cette commande peut être choisie, soit par le biais du menu contextuel, soit par la fenêtre de propriétés. Les 6 comportements disponibles sont les suivant :

- **Commutation à l'appui**
- **Commutation au relâchement**
- **Commutation jusqu'au relâchement**
- **Armement à l'appui**
- **Armement au relâchement**
- **Armement jusqu'au relâchement**

6 comportements pour deux actions distinctes

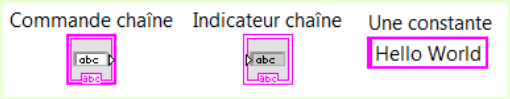


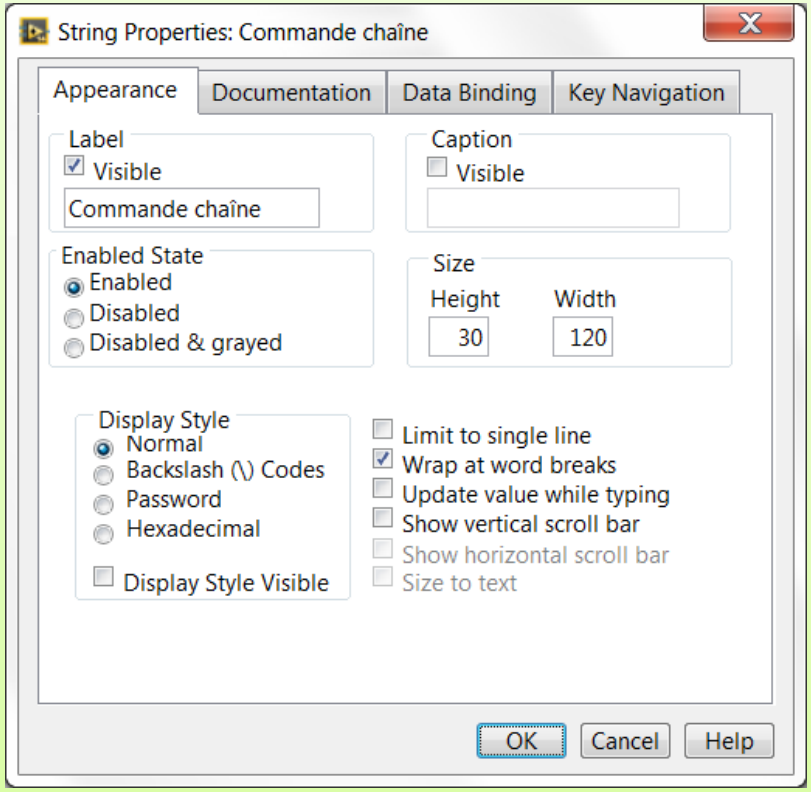
7. Types de données

7.3. Chaines de caractères

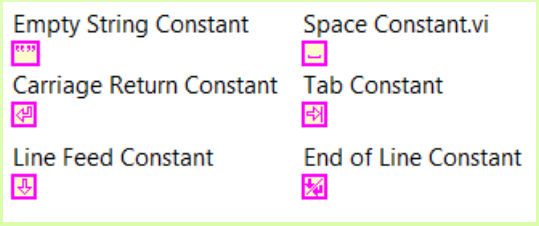
Elles sont constituées par une séquence de caractères ASCII. Chaque chaine dispose, entre autres options, de plusieurs styles d'affichage.

- **Normal**
- **Codes '\'**
- **Mot de passe**
- **Hexadécimal**



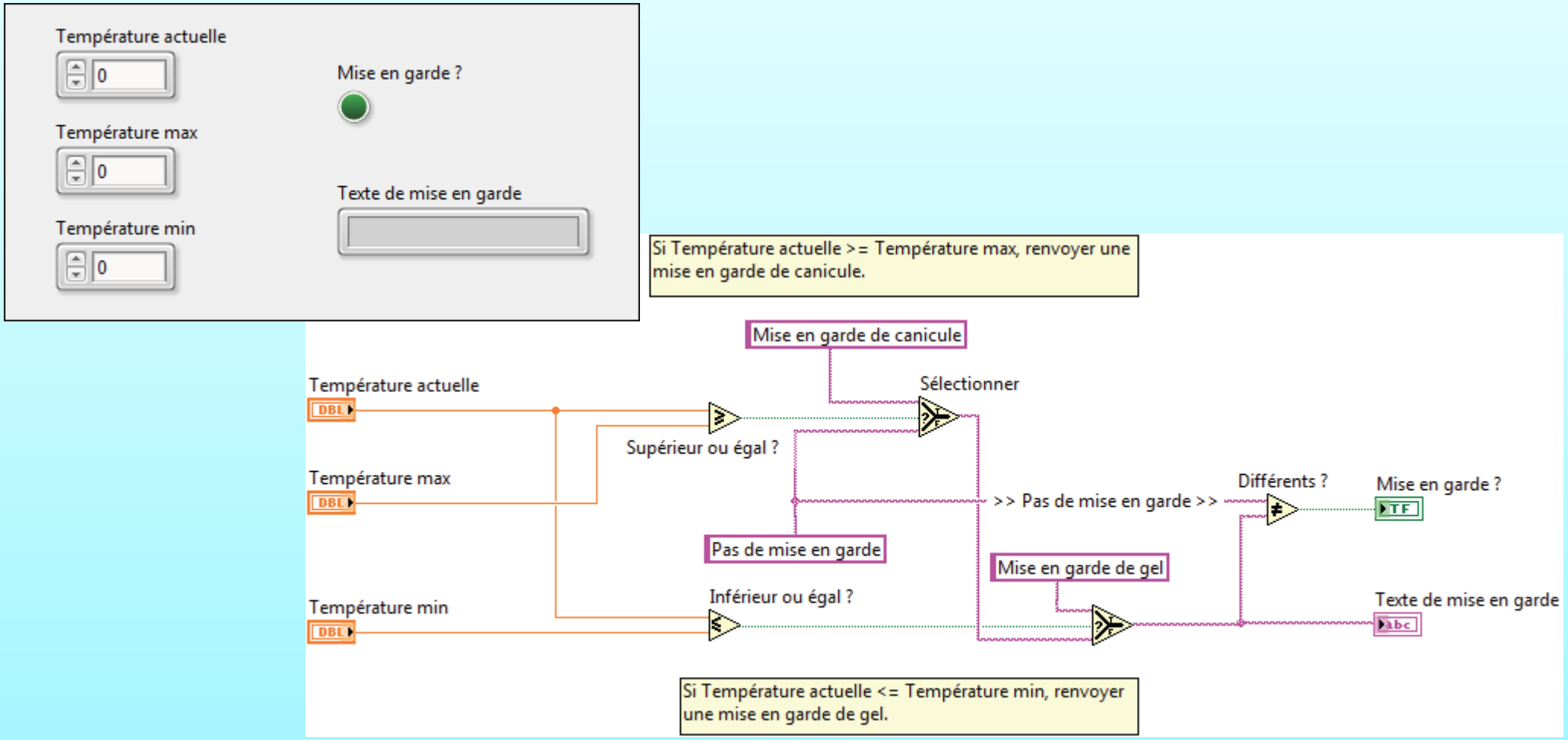


LabVIEW donne également un accès direct à certains caractères spéciaux



Micro-Travaux Pratiques dirigés

V. Utilisation du nœud « Sélectionner »

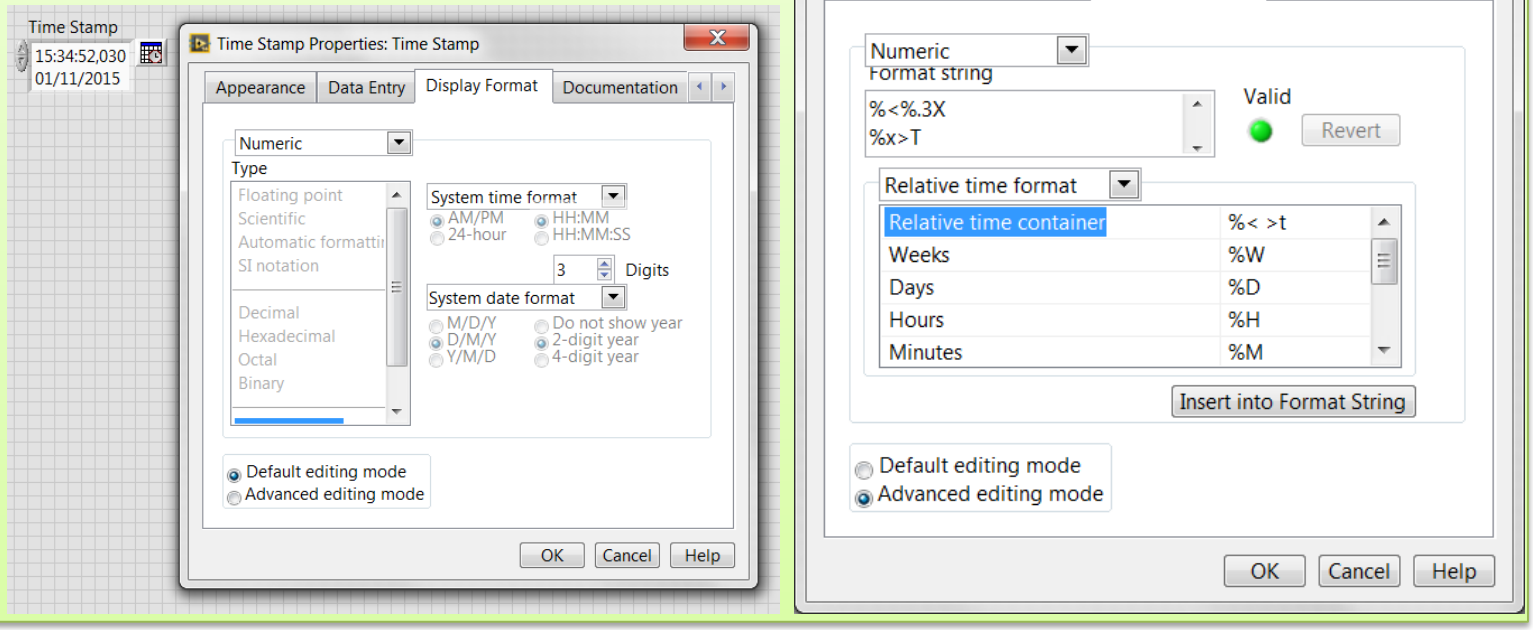


7. Types de données

7.4. Date/Heure

La représentation de la date et l'heure représente un type à part entière, dont la fenêtre de propriétés donne notamment accès aux différentes options d'affichage : par défaut ou avancé

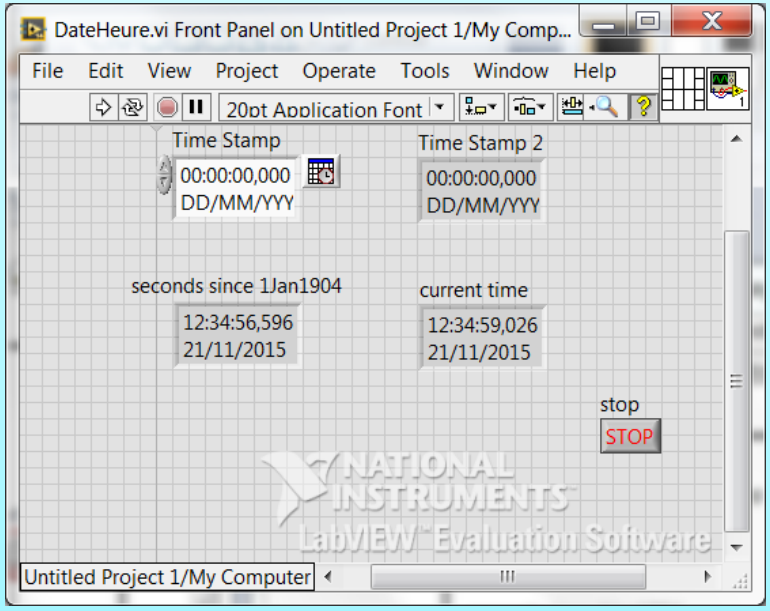
Plusieurs options de formatage de l'heure et de la date sont ainsi disponibles.

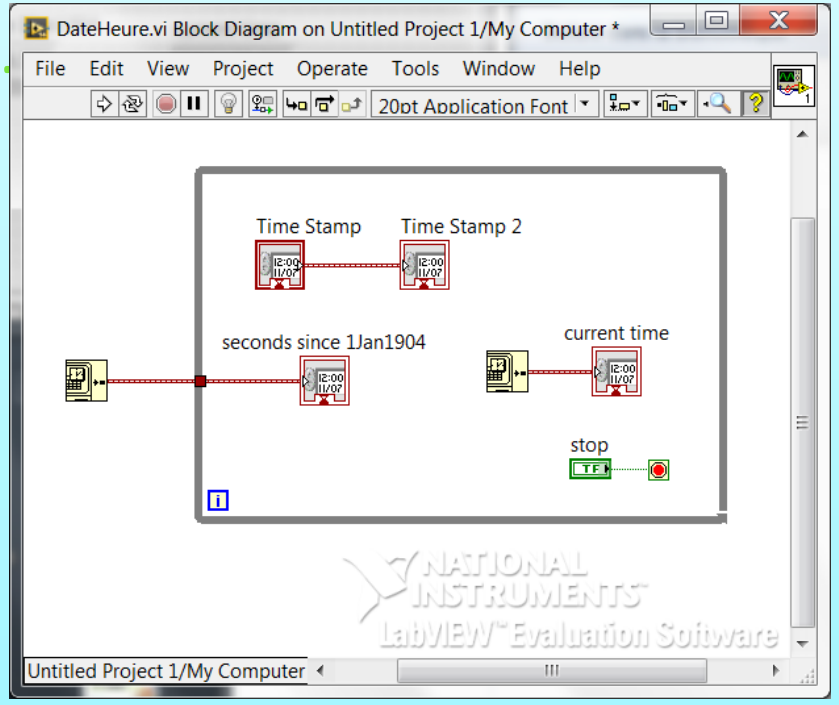


Micro-Travaux Pratiques dirigés

VI. Utiliser la fonction « Get Date/Time in seconds »

- Insérer la fonction à l'extérieur et à l'intérieur d'une boucle While.
- Relier un indicateur pour chacune, et une fonction wait.
- Tester
- Supprimer l'année, et paramétrer la date .

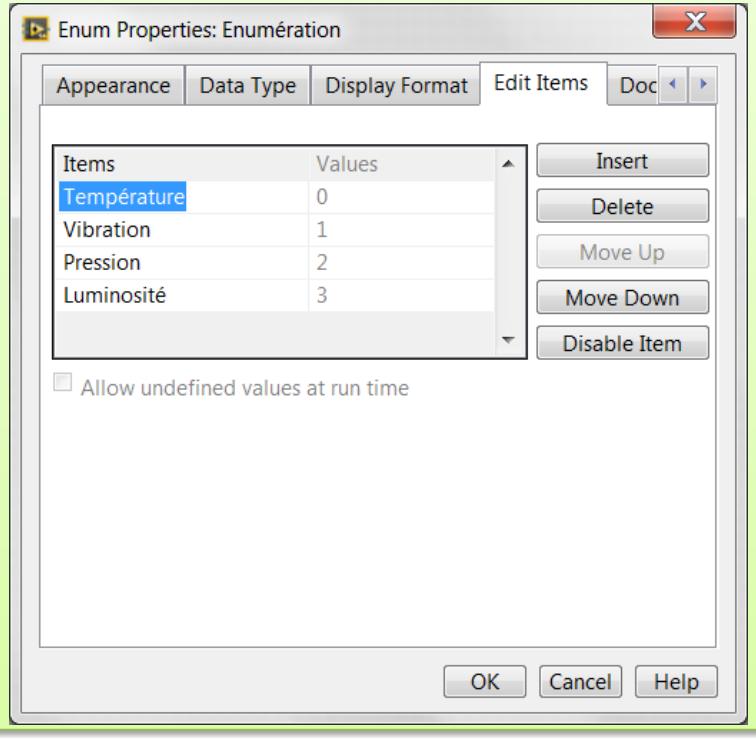


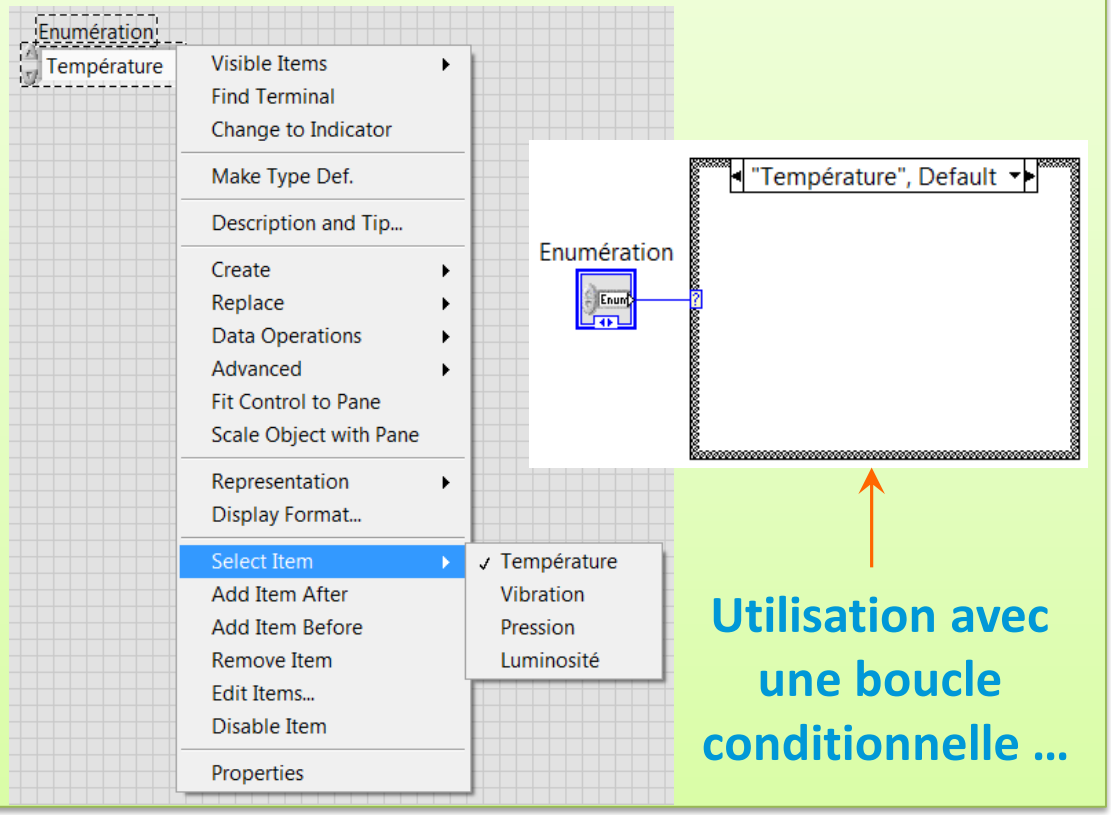


7. Types de données

7.5. Enumérations

Elles se présentent sous la forme d'une liste déroulante dans laquelle l'utilisateur peut sélectionner un élément. A chaque élément correspond une valeur numérique distincte.



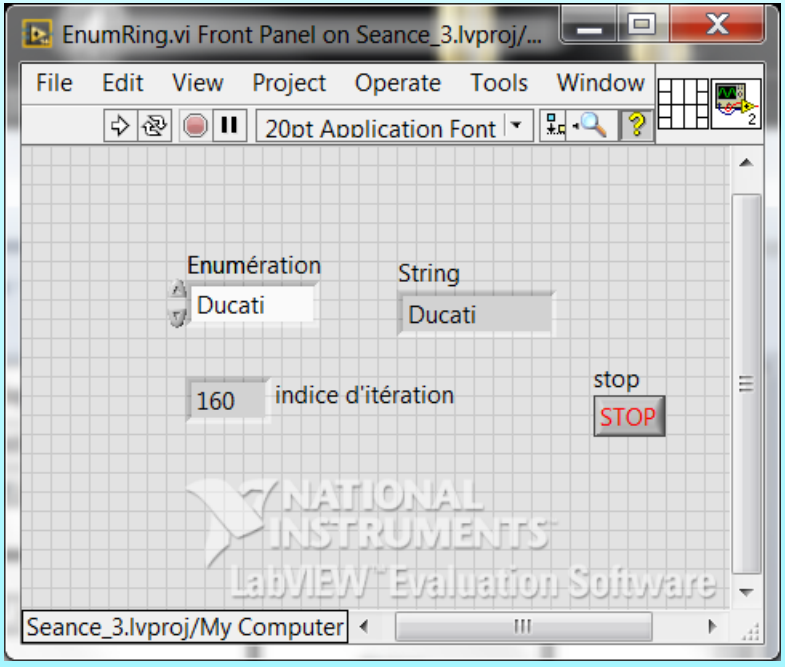


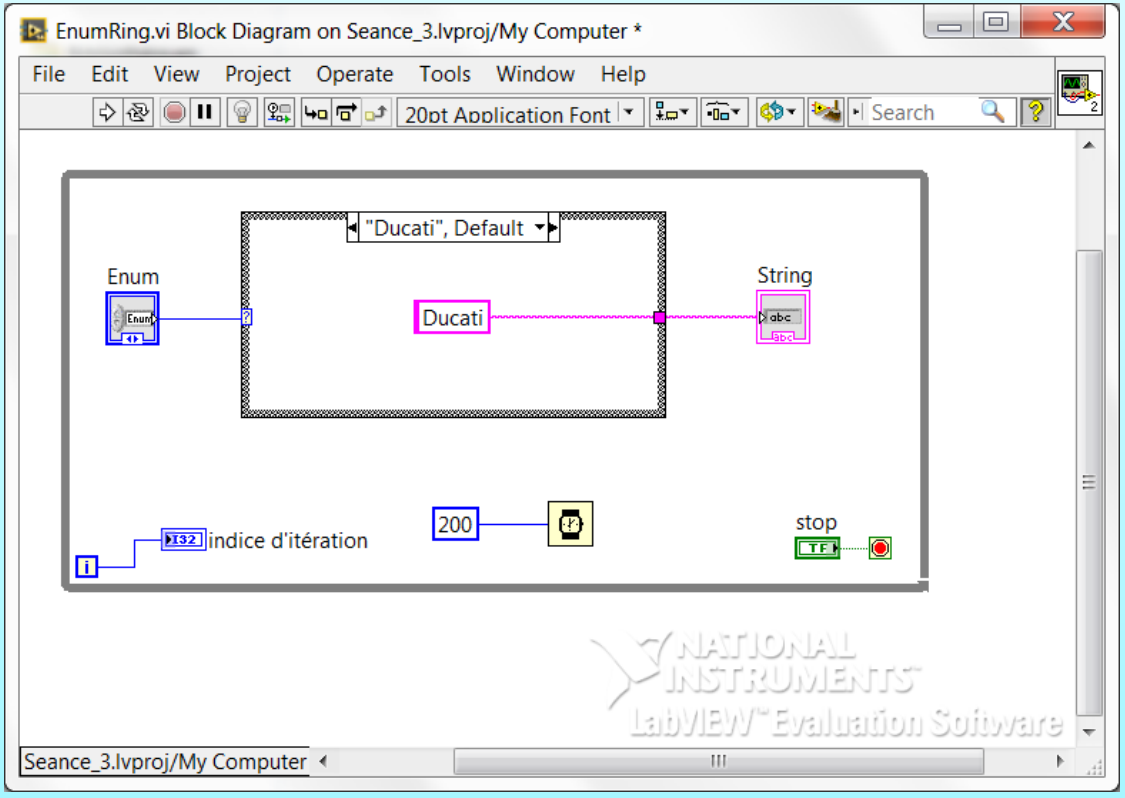
Utilisation avec une boucle conditionnelle …

Micro-Travaux Pratiques dirigés

VII. Enumération et liste

- Mise en évidence des différences entre énumération et liste avec une structure conditionnelle …
- Puis exercice ci-dessous





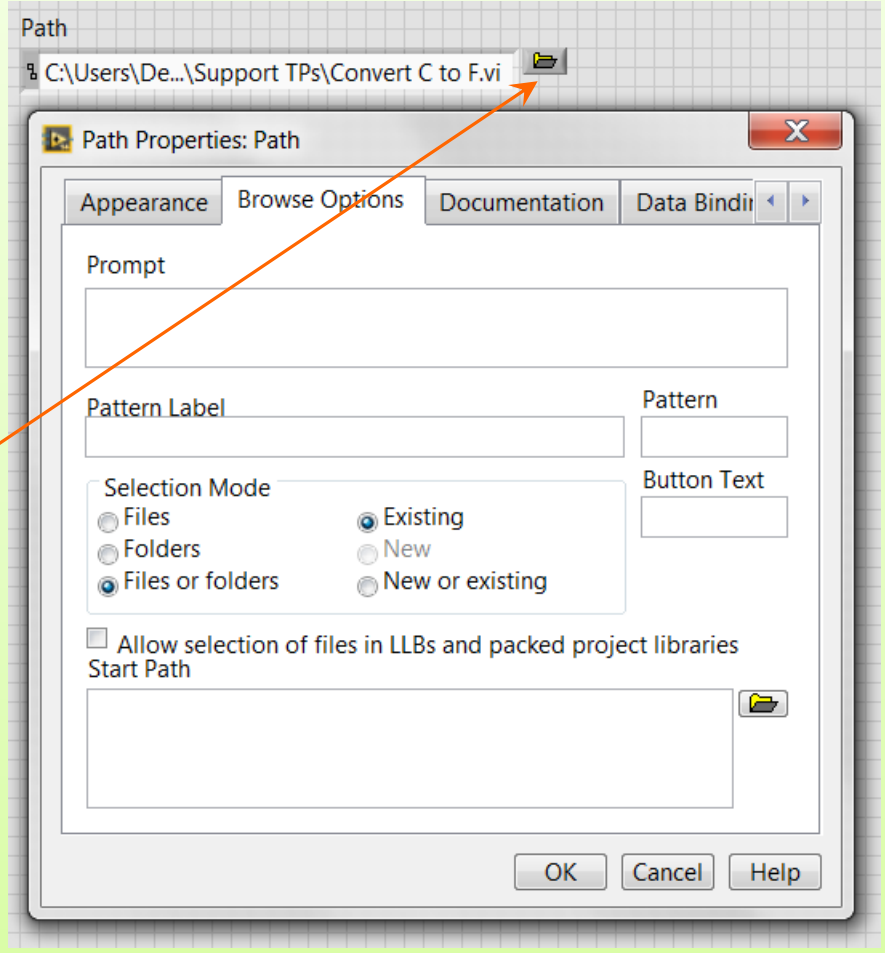
7. Types de données

7.6. Chemin

Il représente le type de l'emplacement ou le lieu de stockage d'un fichier ou d'un répertoire local ou distant selon la syntaxe de la plateforme utilisée.

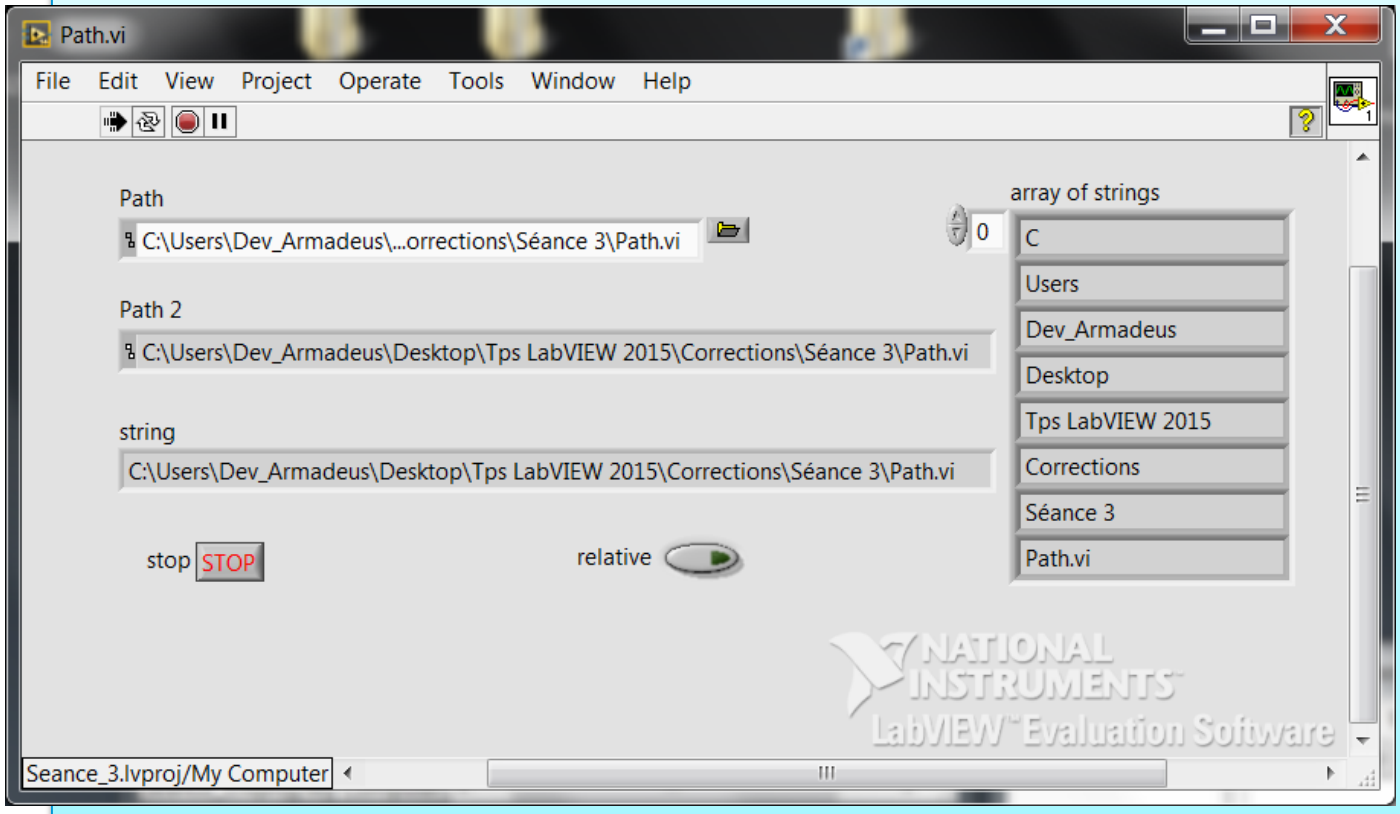
La fenêtre des propriétés permet d'accéder à des options de modes de sélection croisée.

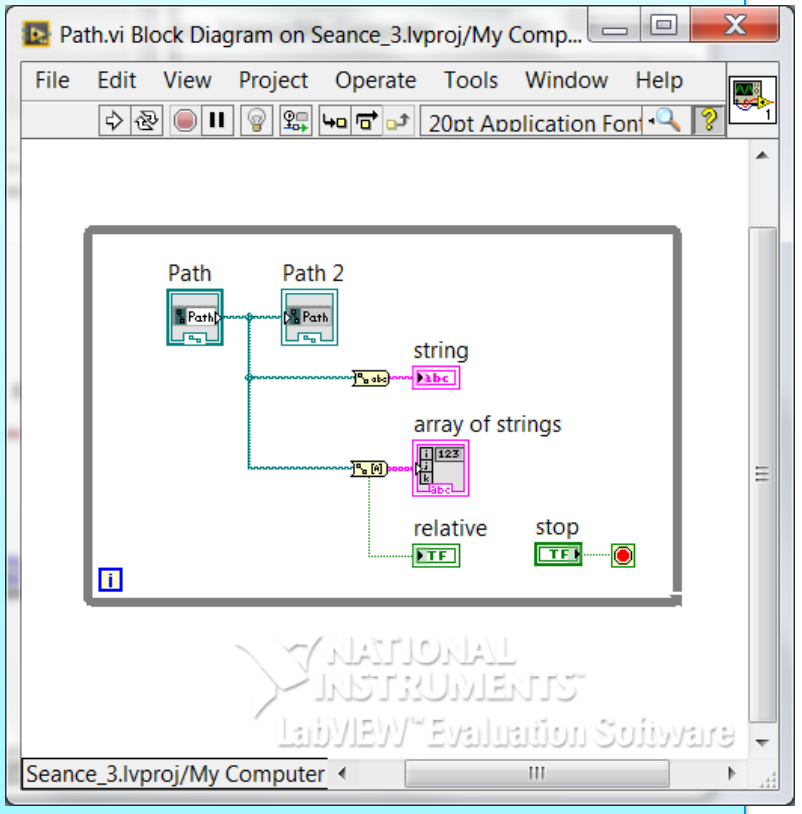
L'utilisateur dispose d'une icône spécifique pour effectuer sa sélection.



Micro-Travaux Pratiques dirigés

VIII. Chemin



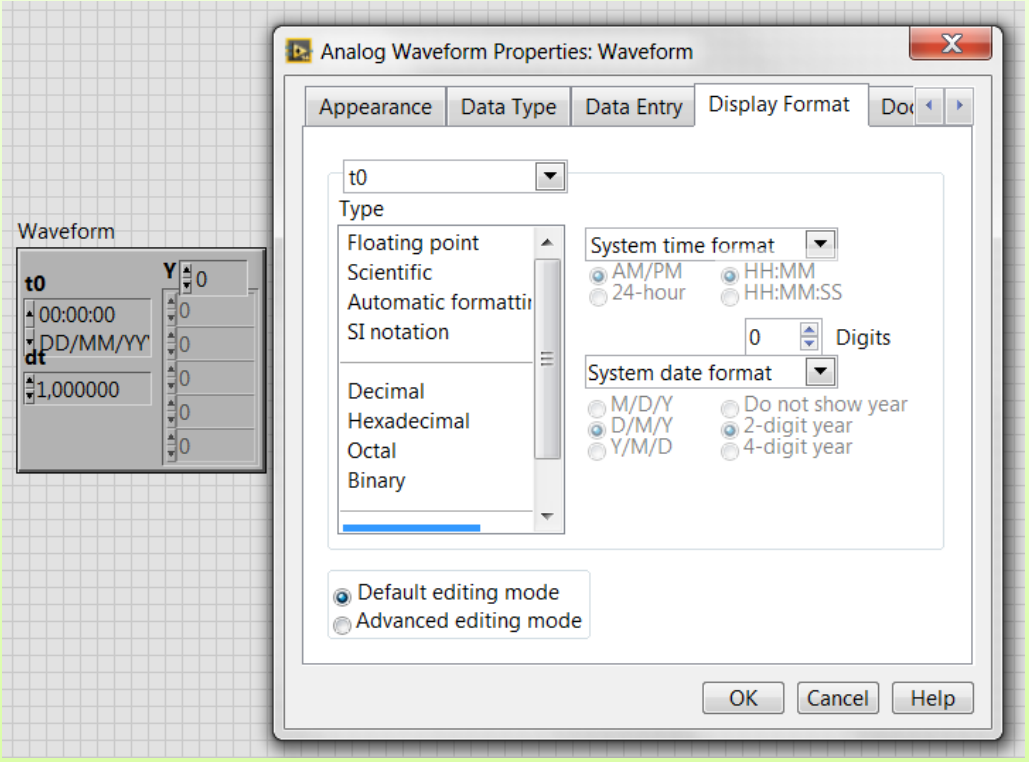


7. Types de données

7.7. « Waveform »

Représente le type d'un signal analogique ou numérique, composé de l'amplitude des échantillons, de l'instant originel et de la période d'échantillonnage.

La fenêtre des propriétés de l'objet donne accès à la représentation et au formatage des 3 éléments distincts de ce type « waveform ».



Micro-Travaux Pratiques dirigés

IX.Waveform

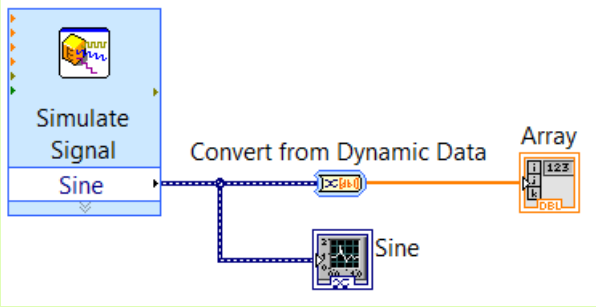
- Insérer une waveform à partir de la palette I/O sur FAV
- Modifier le dt et les premiers éléments du tableau
- Créer un indicateur de type graphe
- Créer une structure while et le bouton stop
- Tester

7. Types de données

7.8. Dynamique

Représente le type des données acquises ou générées par des VI Express.

Les outils de conversion « Convert from Dynamic Data » et « Convert to Dynamic Data » sont alors indispensables dans un contexte mixte avec des données de type signal ou « waveform ».



7. Types de données

7.9. Variant

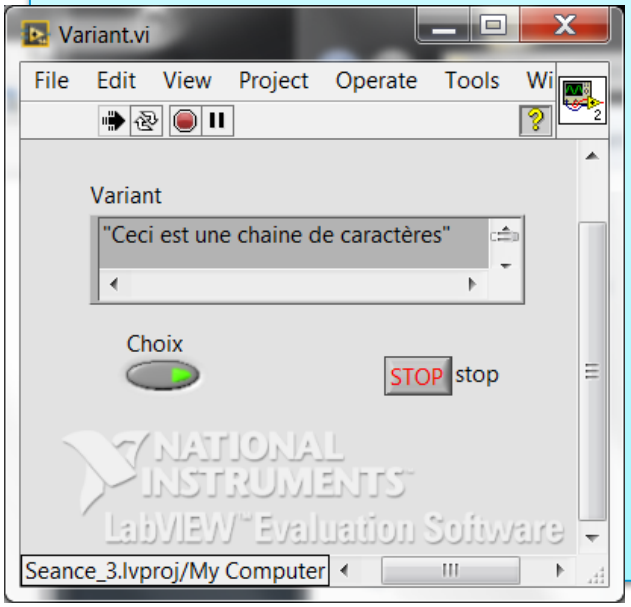
Le type de données « Variant » est un conteneur générique pour tous les types de données. L'intérêt majeur est lié à l'augmentation de la généricité des VIs. Considérons un ensemble de VIs qui manipulent chacun un type de données pour réaliser la même tâche. L'utilisation des variant conduit à obtenir un seul et même VI utilisable pour tous les types de donnés considérés.

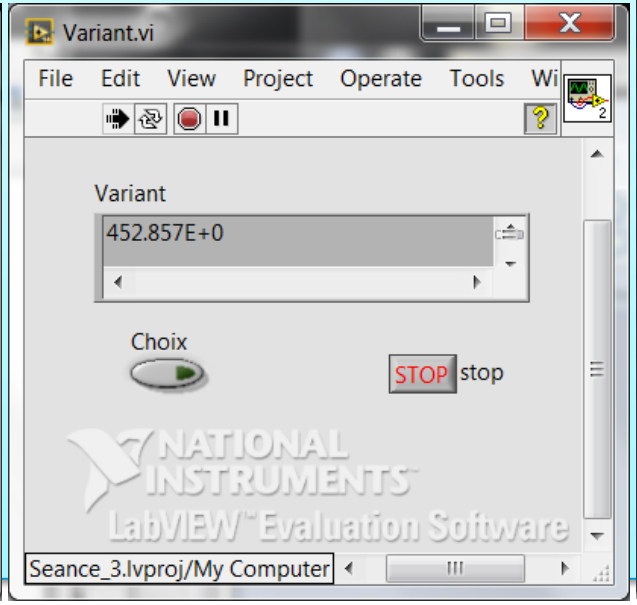
La conversion d'une donnée en un variant conduit à l'enregistrement de la donnée avec son type originel, ce qui permet à LabVIEW de réaliser une conversion inverse ultérieure avec succès.

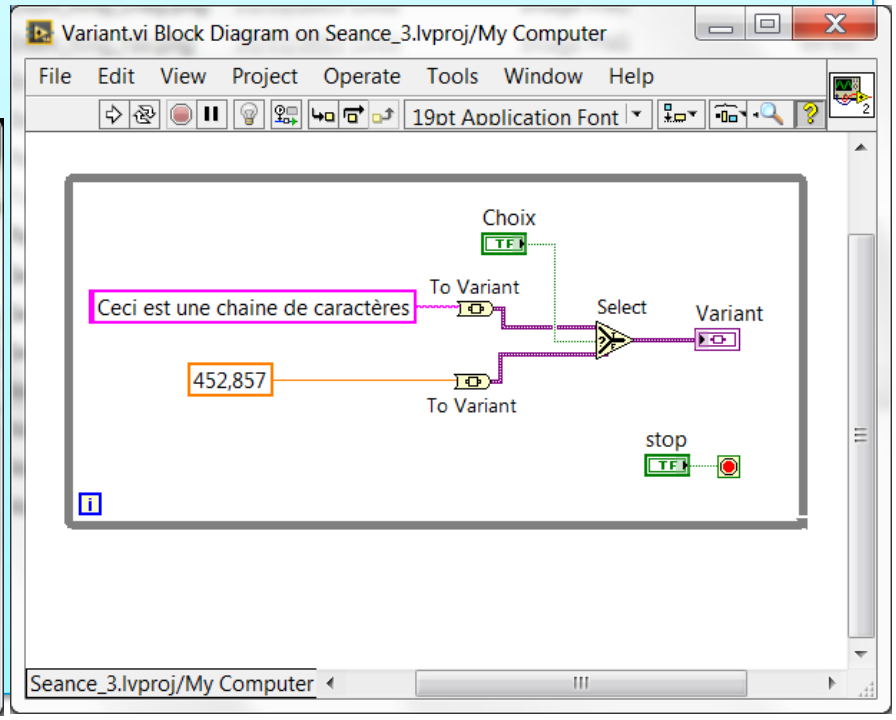
Micro-Travaux Pratiques dirigés

X. Variant

- Insérer un indicateur variant sur FAV
- Insérer un contrôle numérique et une chaîne
- Insérer un select et un contrôle booléen
- Câbler et tester (avec une boucle While)







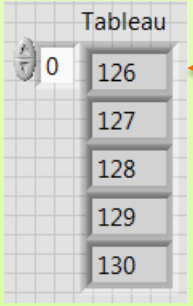
8. Structures de données

8.1. Tableau

Un tableau est un ensemble structuré d'éléments de même type.

Ses caractéristiques principales sont les suivantes :

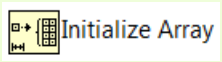
- **Il possède une ou plusieurs dimensions.**
- **Chaque dimension contient un maximum de 2^{31}-1 éléments.**
- **Chaque élément est adressable par son indice dans la dimension considérée.**
- **Le premier élément d'une dimension démarre à l'indice 0.**



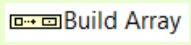
8. Structures de données

8.1. Tableau

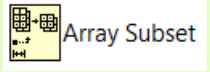
Fonctions courantes de la palette de fonctions « Tableau »

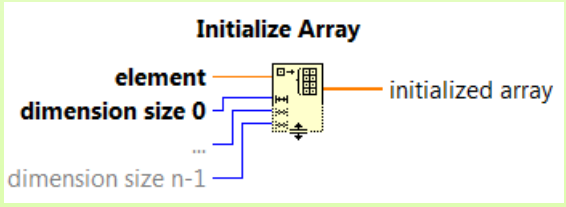
- **Initialisation d'un tableau** 

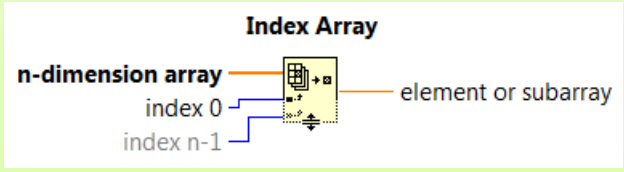
- **Taille d'un tableau** 

- **Construire un tableau** 

- **Indexer un tableau** 

- **Sous-ensemble d'un tableau** 





Micro-Travaux Pratiques dirigés

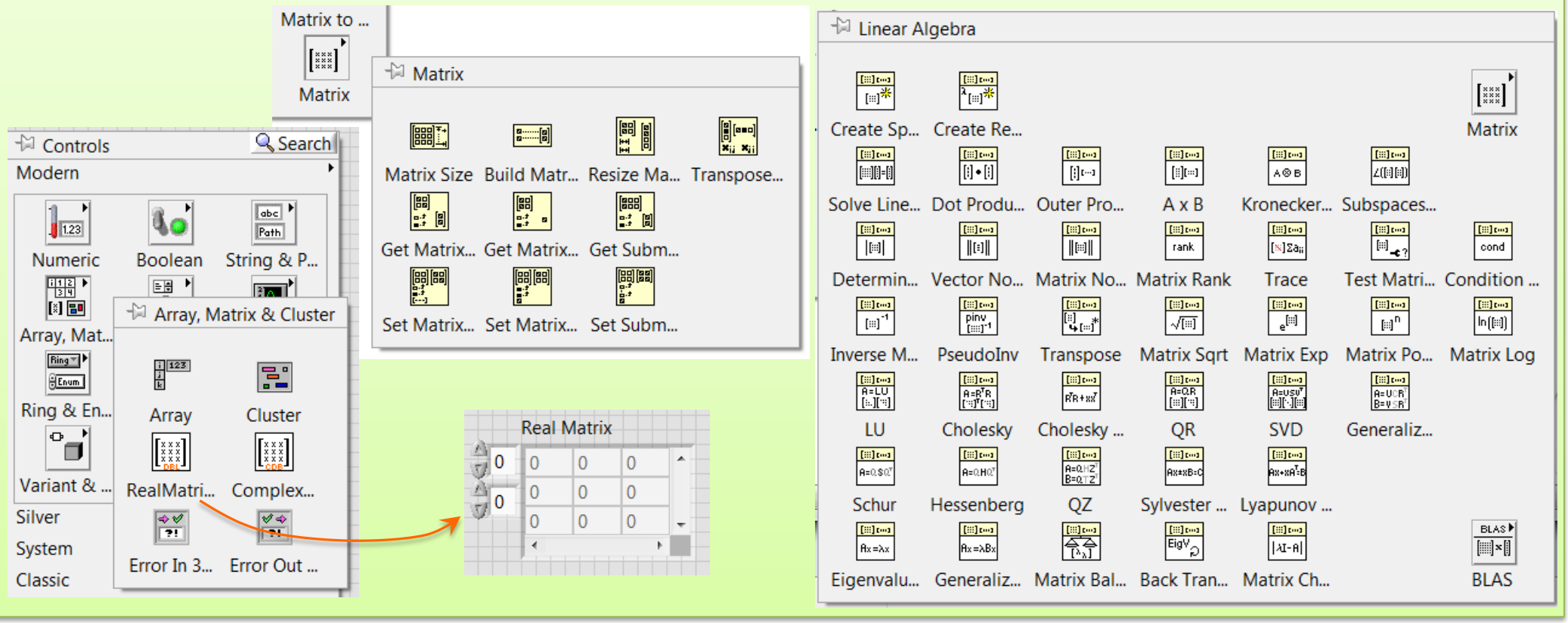
XI. Tableau

- Insérer un tableau à partir de la palette Modern ou Silver
- Intégrer un control numérique en enlevant les items visibles
- Redimensionner le tableau 1D
- Transformer en tableau 2D
- Changer les nombres par défaut
- Tester le décalage d'indice pour visualiser une case dans un tableau à grande dimension

8. Structures de données

8.2. Matrice

Les matrices font parti intégrante de LabVIEW, et disposent aussi de fonctions évoluées : *Algèbre linéaire* de la bibliothèque Mathématiques.



8. Structures de données

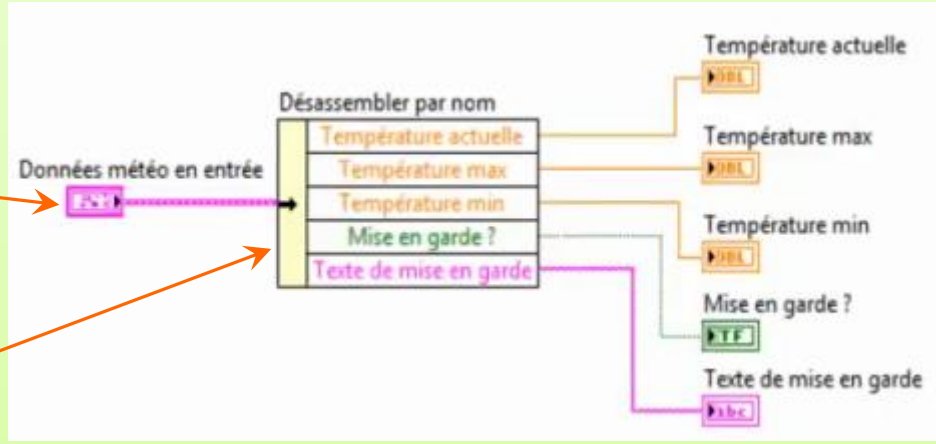
8.3. Cluster

Outil flexible d'une grande utilité, il regroupe des éléments de type différents, et s'apparente aux « record » et « struct » des langages de programmation textuels.

Il permet de regrouper de façon logique les éléments constitutifs d'un ensemble, et améliore nettement la lisibilité du code.

Cluster et son terminal

Fonction d'accès aux éléments

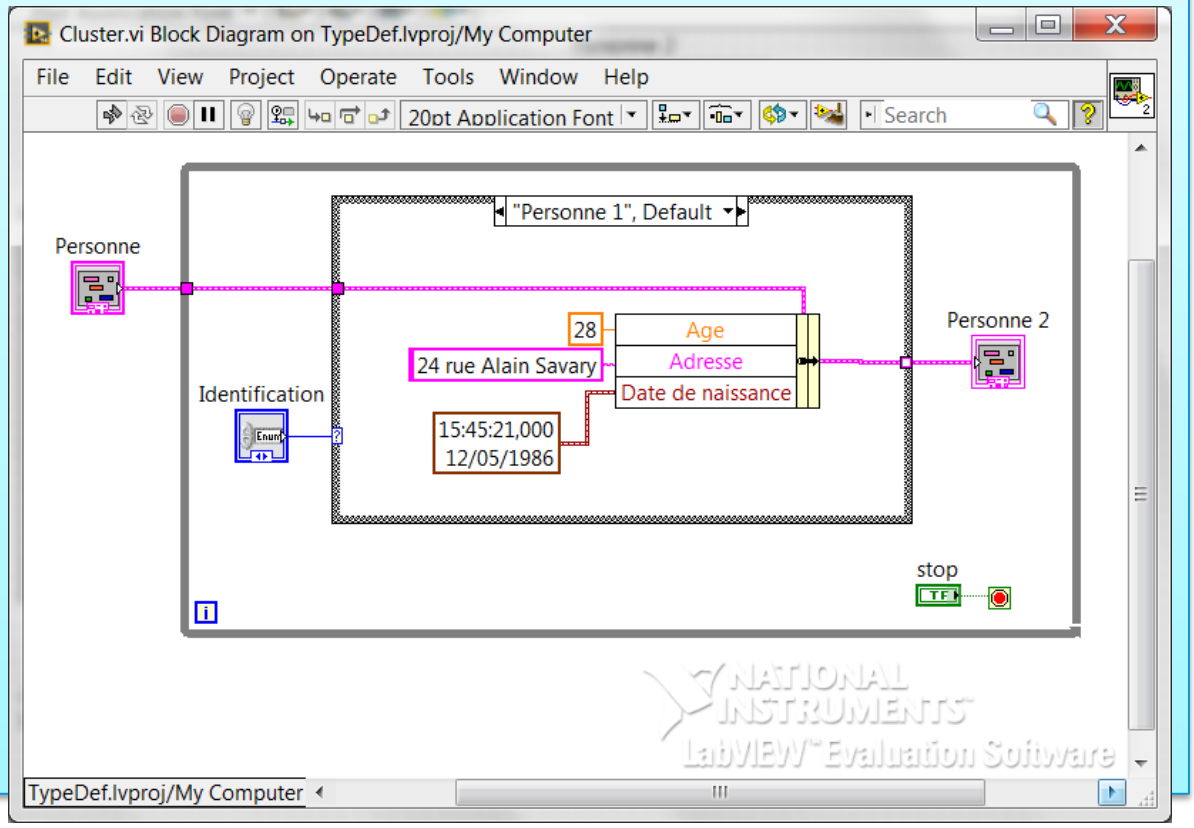


Micro-Travaux Pratiques dirigés

XII. Cluster

- Créer le cluster *Données météorologiques*
- Initialiser (Forcer la valeur d'une variable) les données avec une fonction *Bundle by name*
- Lire les données du cluster, pour les afficher, en utilisant la fonction bundle by name

Ou réaliser l'exemple suivant



8. Structures de données

8.4. « Type Definition »

La définition d'un type, réalisée à partir d'une commande, d'un indicateur ou d'une constante permet de créer un modèle de l'élément d'origine, puis de le modifier au besoin et d'en créer des instances utilisables dans un projet.

L'intérêt majeur est qu'une modification du modèle est applicable à l'ensemble des instances utilisées de ce modèle.

Cet outil s'apparente de la notion d'héritage en langage objet.

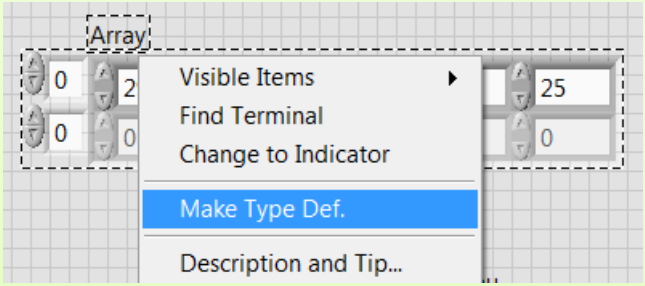
L'option « En faire une def. de type » (Make Type Def.) des menus contextuels permet de réaliser la création du modèle qui doit ensuite être enregistré au sein du projet comme un autre fichier natif LabVIEW, et avec l'extension « .ctrl ».

8. Structures de données

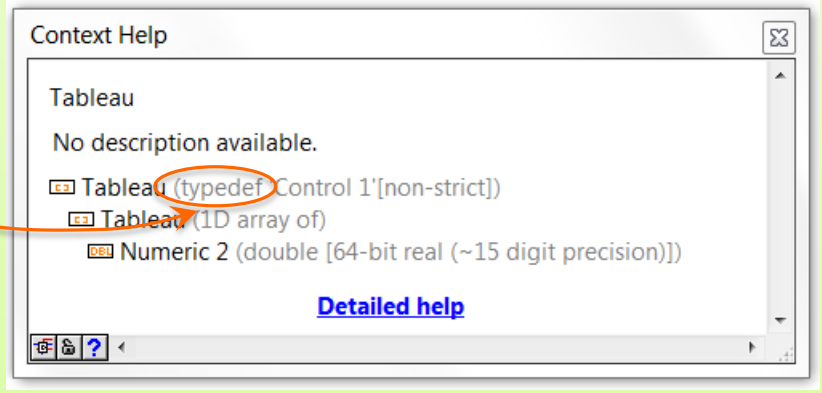
8.4. « Type Definition » (Suite)

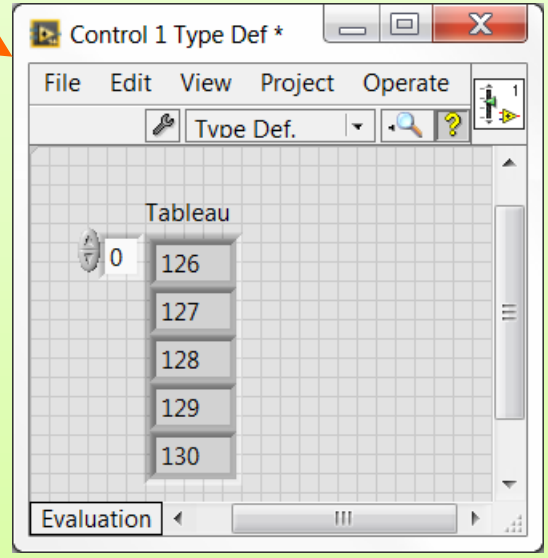
Un triangle noir en haut et à gauche d'un terminal indique que ce dernier est une instance de « Type Def. ».

Ceci est un Type Def.



Fenêtre de modification et d'enregistrement du Type Def.





Micro-Travaux Pratiques dirigés

XIII. Type Def

- Créer un projet vide et un VI nommé VI principal
- Créer une énumération de 2 éléments sur ce VI
- Transformer cet enum en type def et l'enregistrer (.ctrl)
- Créer un VI secondaire
- Insérer deux instanciations du type def sur ce VI secondaire
- Ajouter un troisième élément au type def
- Vérifier la mise à jour automatique des deux instanciations

9. Formats de fichiers

Les 4 formats reconnus

LabVIEW permet de créer, d'écrire et de lire des données, et plus généralement de manipuler les 4 formats de fichiers suivant :

- **Binaire :** ce format permet d'accéder à une compacité extrême, d'où son utilisation particulièrement adaptée à l'acquisition de données haute vitesse et multivoies,
- **ASCII :** les données sont codées sous forme de chaines de caractères (texte en clair), ce qui prédispose le format à une utilisation en acquisition basse vitesse (accessible par traitement de texte),
- **LVM :** données ASCII (champs) sont délimitées par des tabulations, ou fichier de mesures LabVIEW accessible par un tableur,
- **TDMS :** données binaires structurées dont les propriétés sont stockées dans un fichier annexe (format propriétaire NI).

Micro-Travaux Pratiques dirigés

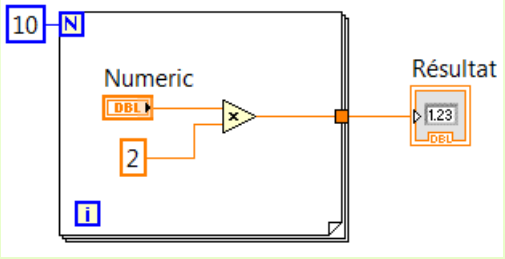
XIV. Fichiers de données d'acquisititon

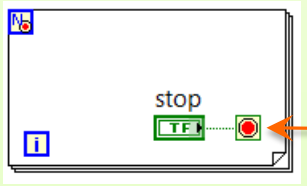
- VI Express input simulate (sinusoïde 1000 points)
- File I/O Write to mesurement file en lvm
- Test et ouverture fichier avec wordpad
- Puis, modifier write au format tdms
- Insérer read from … à partir de file I/O
- Créer un indicateur graphe
- Test

10. Structures logiques

10.1. Boucle « For »

Cette structure permet d'exécuter un traitement N fois. Ces N itérations sont numérotées au travers de l'indice i, qui lui varie de 0 à N-1.





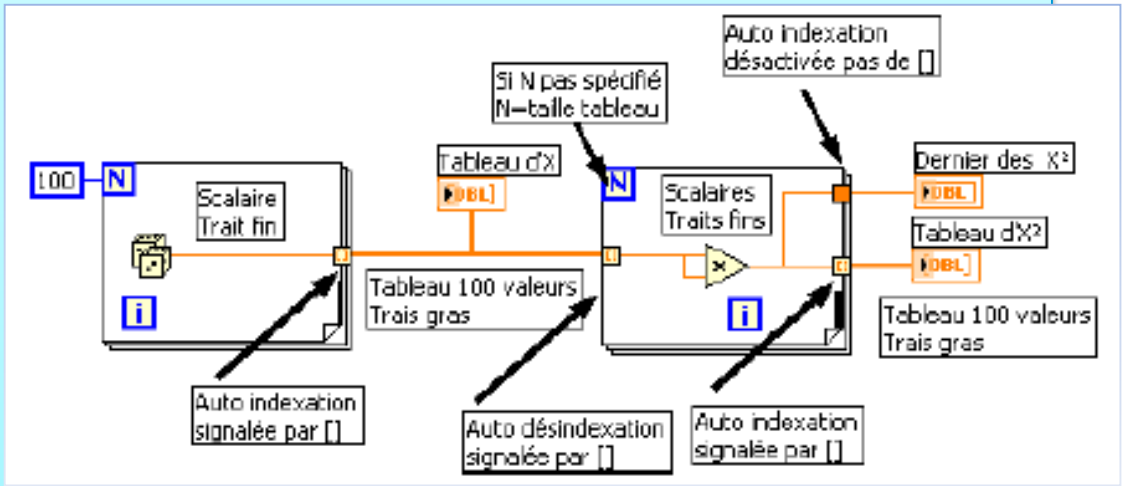
Les boucles « For » disposent de certaines particularités :

- **Un terminal conditionnel peut faire stopper la boucle de manière prématurée.**
- **Les registres à décalages sont utilisables …**
- **Les itérations sont potentiellement parallélisables.**

Micro-Travaux Pratiques dirigés

XV. Boucle « For »

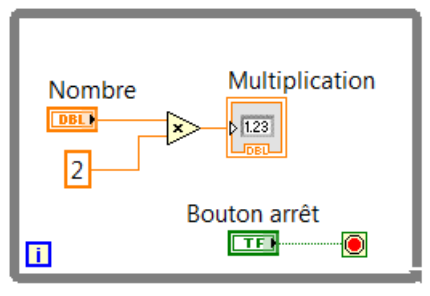
- Insérer un subdiagram label
- Tester le fonctionnement standard
- Insérer un terminal conditionnel avec control booléen
- Insérer une fonction wait
- Tester l'arrêt « prématuré »
- Modifier le terminal conditionnel de stop if true en continue if true
- Tester les 3 options de mode de tunnel



10. Structures logiques

10.2. Boucle « While »

La structure « Tant que » permet d'exécuter un traitement jusqu'à ce que le terminal conditionnel soit activé, en règle général via un contrôle de type booléen ou bouton situé en face-avant.

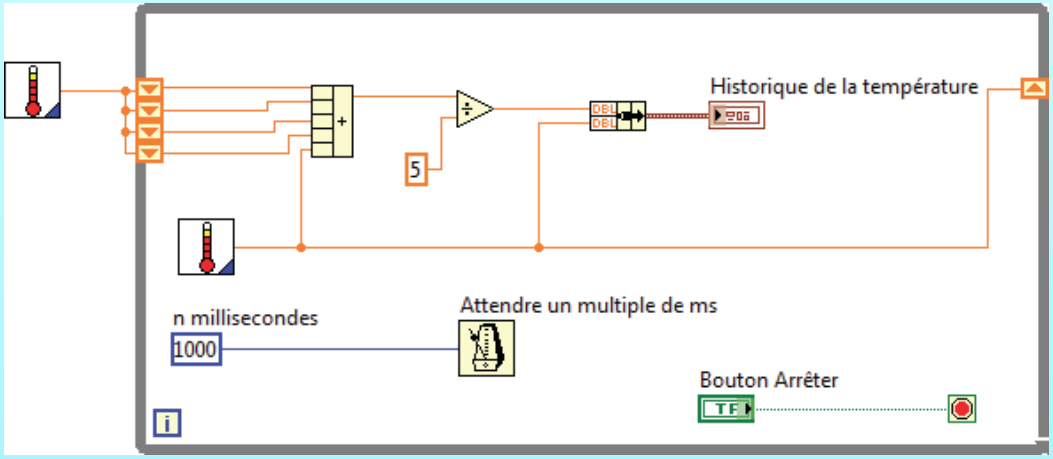


Les boucles « While » disposent également de certaines particularités :

- **Le terminal conditionnel peut être activé sur une condition Vrai ou Faux (comme pour la boucle « For »).**
- **Les registres à décalage sont également disponibles.**

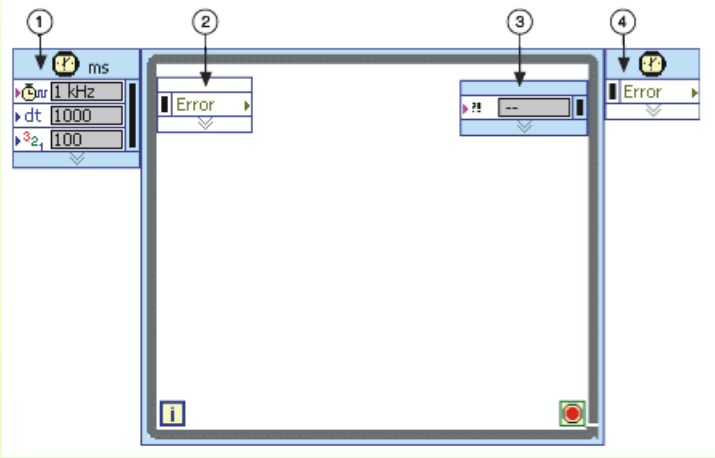
XVI. Boucle « While »

- Insérer un subdiagram label
- Tester le fonctionnement standard
- Modifier le terminal conditionnel en continue if true
- Tester les registres à décalage en ajoutant plusieurs éléments en entrée



10.3. Structure de boucle cadencée

La boucle cadencée est une boucle While, dont on définie la période de temps qui sépare deux itérations consécutives.

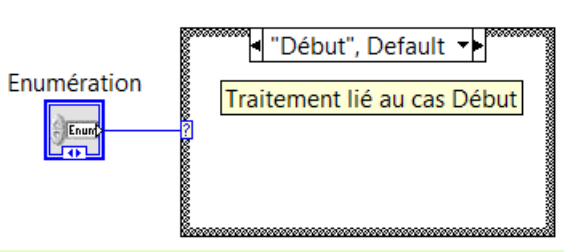


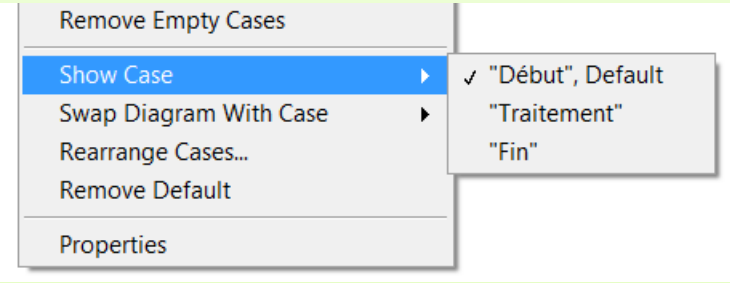
1 Nœud d'entrée
2 Nœud de données gauche
3 Nœud de données droit
4 Nœud de sortie

Cette boucle dispose de nœuds d'entrées et de sorties, ainsi que les nœuds de données (internes) gauche et droit. Chacun de ces nœuds met à disposition des paramètres comme la période d'itération, un « timeout », « Finished late ? », les erreurs …

10.4. Structure conditionnelle

La structure conditionnelle permet d'exécuter un traitement spécifique au cas activé par la valeur de la condition (d'entrée).





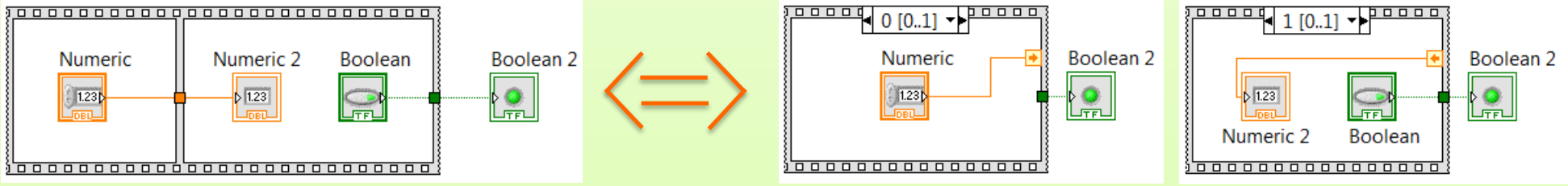
Extrait du menu contextuel

Une des caractéristique de cette structure est lié à l'aspect adaptatif de la condition d'entrée, qui s'adapte au type de la donnée qui lui est connecté, ainsi qu'au nom de chacun des cas ici les éléments de l'énumération.

10. Structures logiques

10.5. Structure séquentielle

Cette structure permet de morceler le code en portions de code organisées pour s'exécuter dans un certain ordre. Chaque séquence s'exécute en commençant par celle située le plus à gauche, et dans l'hypothèse ou une données est transférée d'une séquence à une autre, une séquence ne s'exécute que si la donnée en entrée est « valide ».

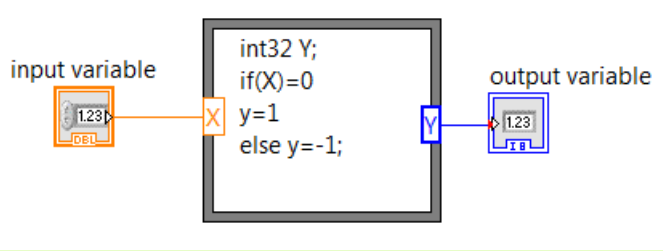


Cette structure revêt trois formes, une dite aplatie (par défaut), l'autre empilée, à l'image d'une structure conditionnelle, et la dernière cadencée.

10. Structures logiques

10.6. Nœud de formules

Cette structure permet d'évaluer des formules et expressions mathématiques similaires à celle utilisées en langage C.



Outre les fonctions mathématiques suivantes, la structure permet de connecter des entrées et des sorties : abs, acos, acosh, asin, asinh, atan, atan2, atanh, ceil, cos, cosh, cot, csc, exp, expm1, floor, getexp, getman, int, intrz, ln, lnp1, log, log2, max, min, mod, pow, rand, rem, sec, sign, sin, sinc, sinh, sizeOfDim, sqrt, tan, tanh.

11. Notion de flux de données

11.1 Principe

Directement liée au transfert d'informations, sous la contrainte de l'exécution d'une seule « instruction » à un instant t, cette notion fondamentale doit être intégrée très tôt par le développeur, afin que son code s'exécute selon un schémas qu'il se doit d'imaginer et d'organiser.

La programmation séquentielle constitue le concept de base, qui tient au fait qu'une fonction, qu'une portion de code ne doit s'exécuter que si ses entrées sont dites valides, c'est-à-dire le fruit du résultat de la fonction précédente dans un schémas d'exécution bien établi.

La programmation graphique avec l'utilisation des liaisons peut facilement leurrer le développeur novice qui ferait l'impasse sur ce concept fondamental.

11. Notion de flux de données

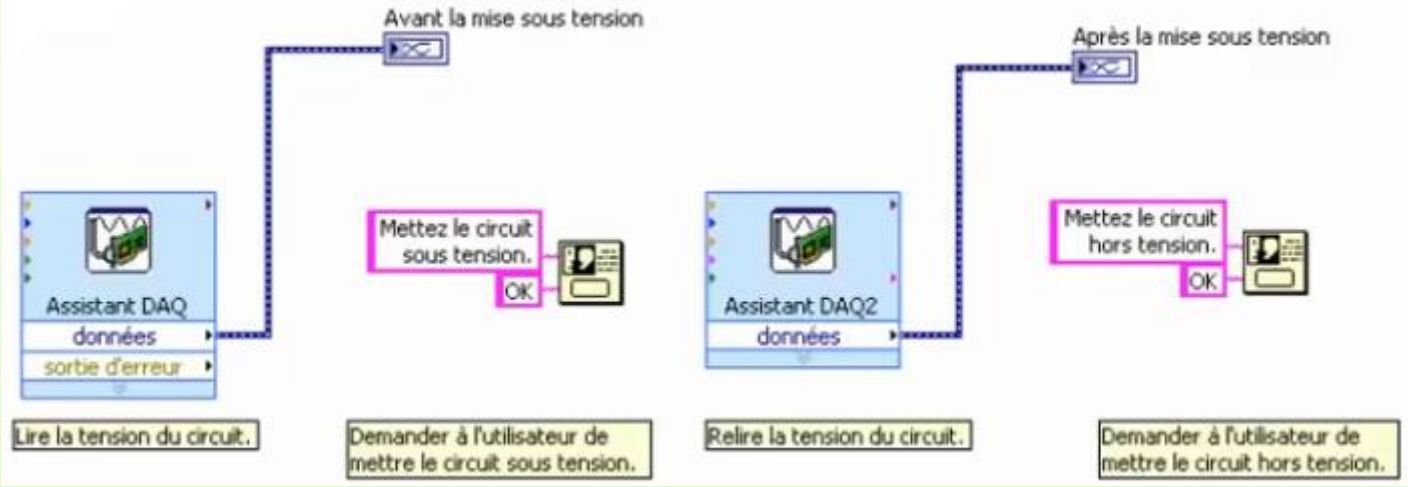
11.2 Bonnes pratiques

- **Concevoir modulaire : utilisation judicieuse des sous-Vis**
- **Utilisation des fils d'erreurs des fonctions pour imposer un parcours d'exécution**
- **Utilisation des séquences avec parcimonie**
- **Utilisation conjointe des fils d'erreurs et des structures conditionnelles**
- **Un diagramme ou « sous-diagramme » doit tenir sur une page écran**
- **…**

11. Notion de flux de données

11. 3 Exemple

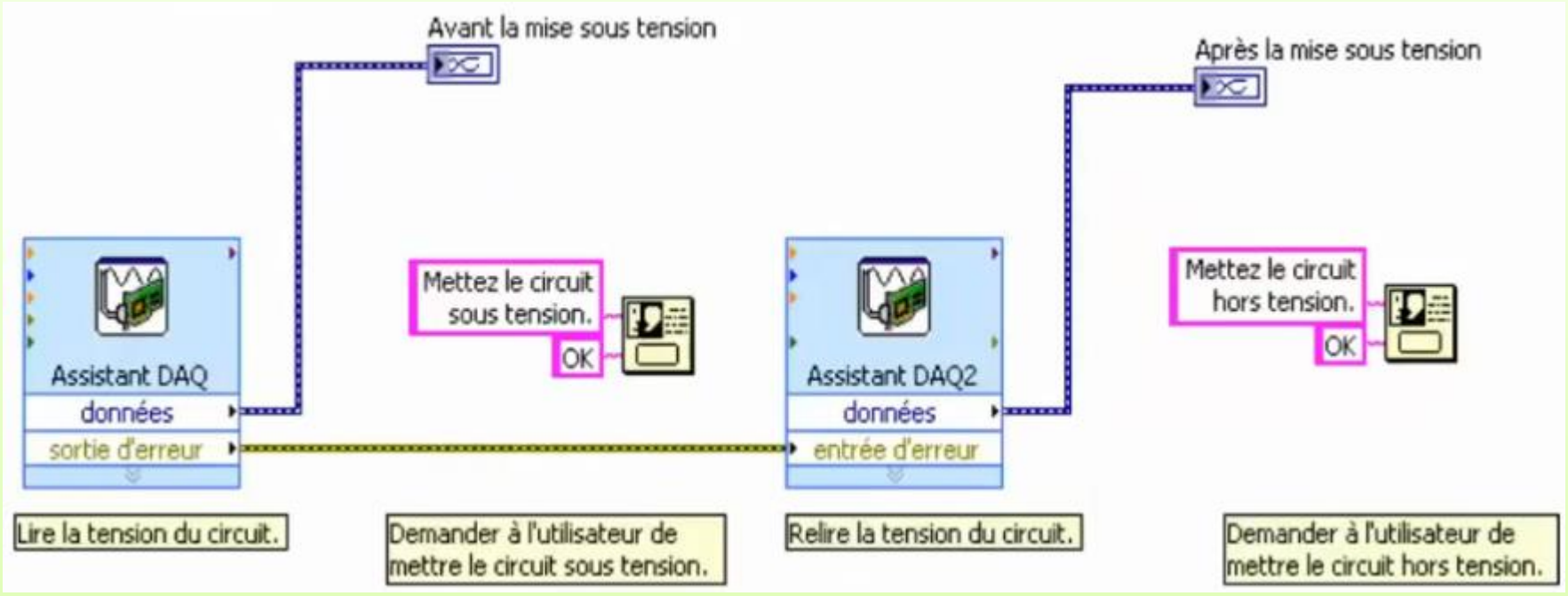
Voici un exemple de code constitué de 4 étapes à séquencer dans l'ordre indiqué : 2 VI Express, et 2 boites de dialogue



Etape 1 Etape 2 Etape 3 Etape 4

11. Notion de flux de données

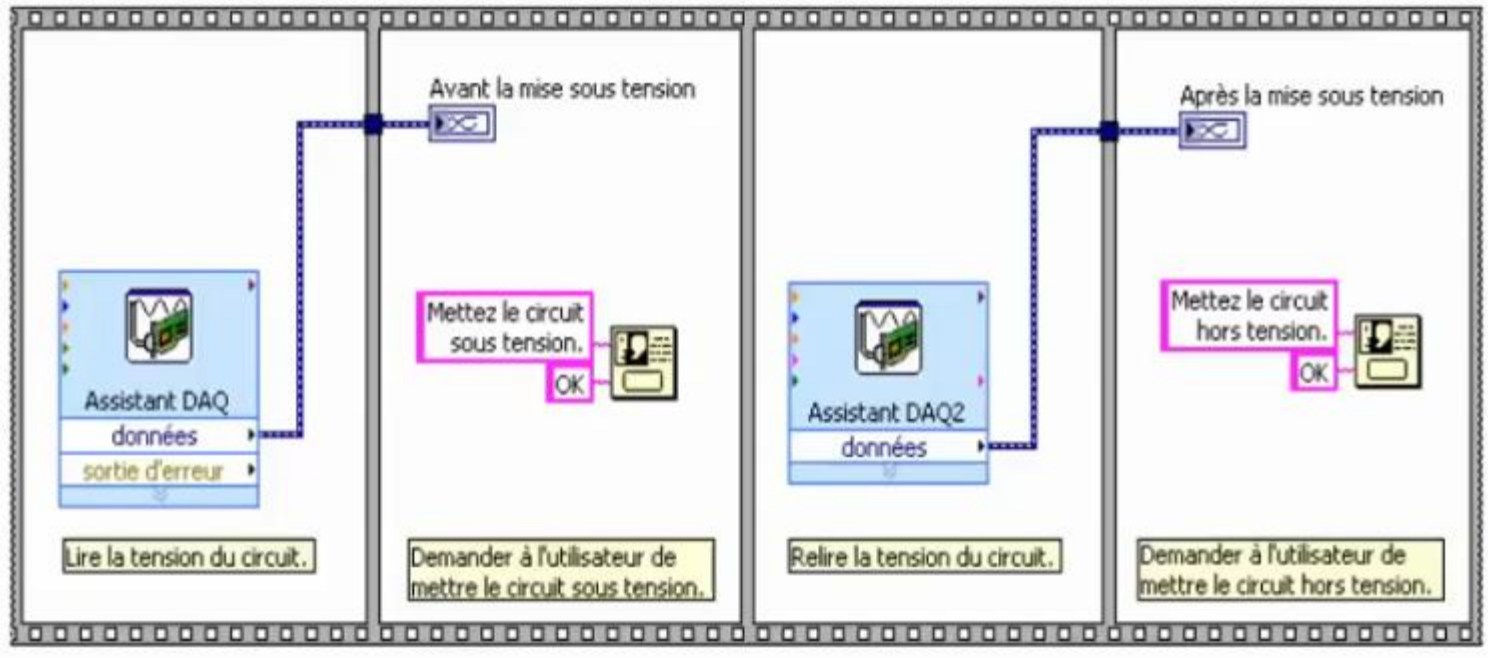
11. 3 Exemple



Le fil d'erreur relie la sortie d'erreur d'un VI Express à l'entrée d'erreur du second : l'ordre d'exécution est ainsi bien défini. En revanche, rien n'est défini concernant les deux boites de dialogue.

11. Notion de flux de données

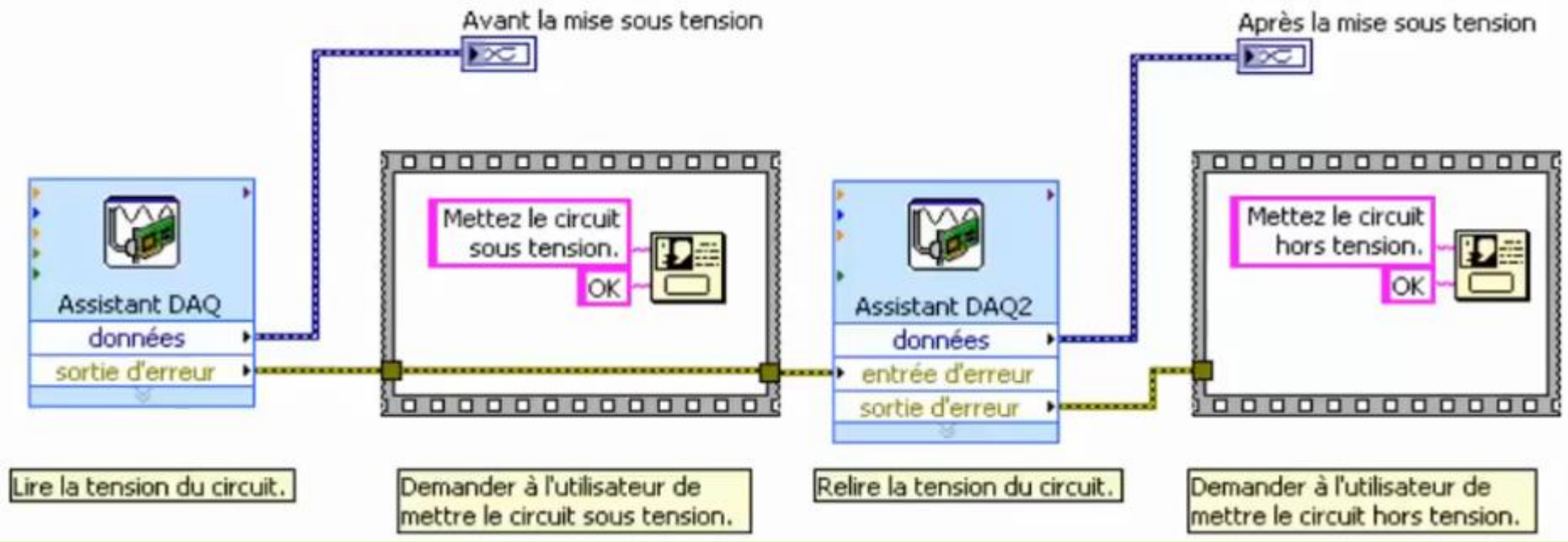
11. 3 Exemple



Les séquences permettent bien de définir un ordre d'exécution : une séquence ne « démarre » que si la précédente est totalement terminée.
En revanche, les erreurs ne sont pas gérées …

11. Notion de flux de données

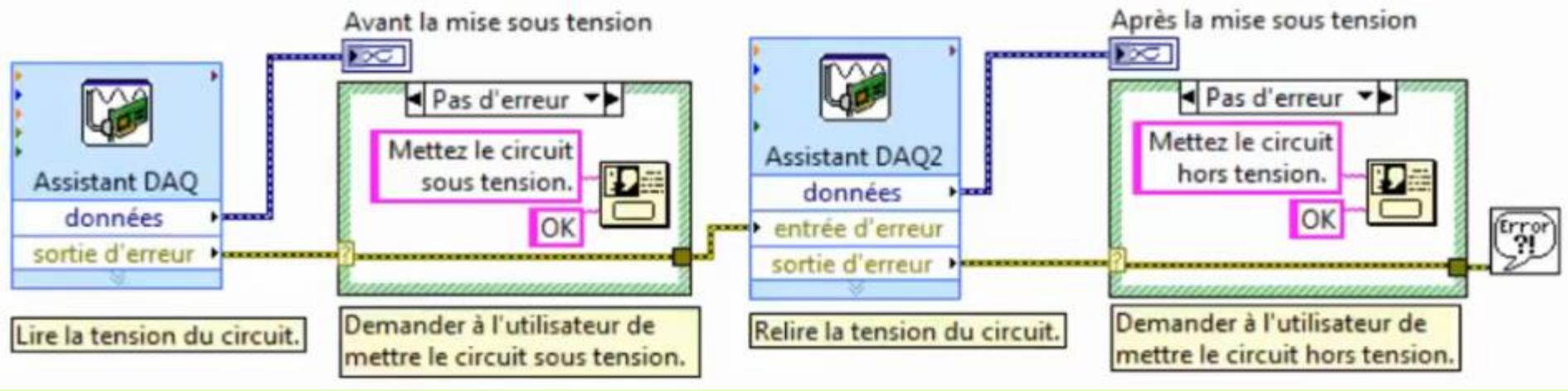
11. 3 Exemple



La suppression de deux structures séquentielles, rendue possible par l'utilisation des fils d'erreurs, permet de gagner en lisibilité.
En revanche, une erreur qui survient n'est pas traitée, mais seulement propagée …

11. Notion de flux de données

11. 3 Exemple



Cette solution est meilleure que les précédentes car, en plus des avantages précédents, elle traite une erreur au moment où elle peut apparaitre. Le fil d'erreur est câblé sur l'entrée conditionnelle d'une structure du même nom, et la boite de dialogue est insérée dans la condition « Pas d'erreur ».

L'autre condition sert à sécuriser le système en cas d'erreur sur le premier VI Express, en n'autorisant pas la mise sous tension du système en question.

12. Notion de modularité

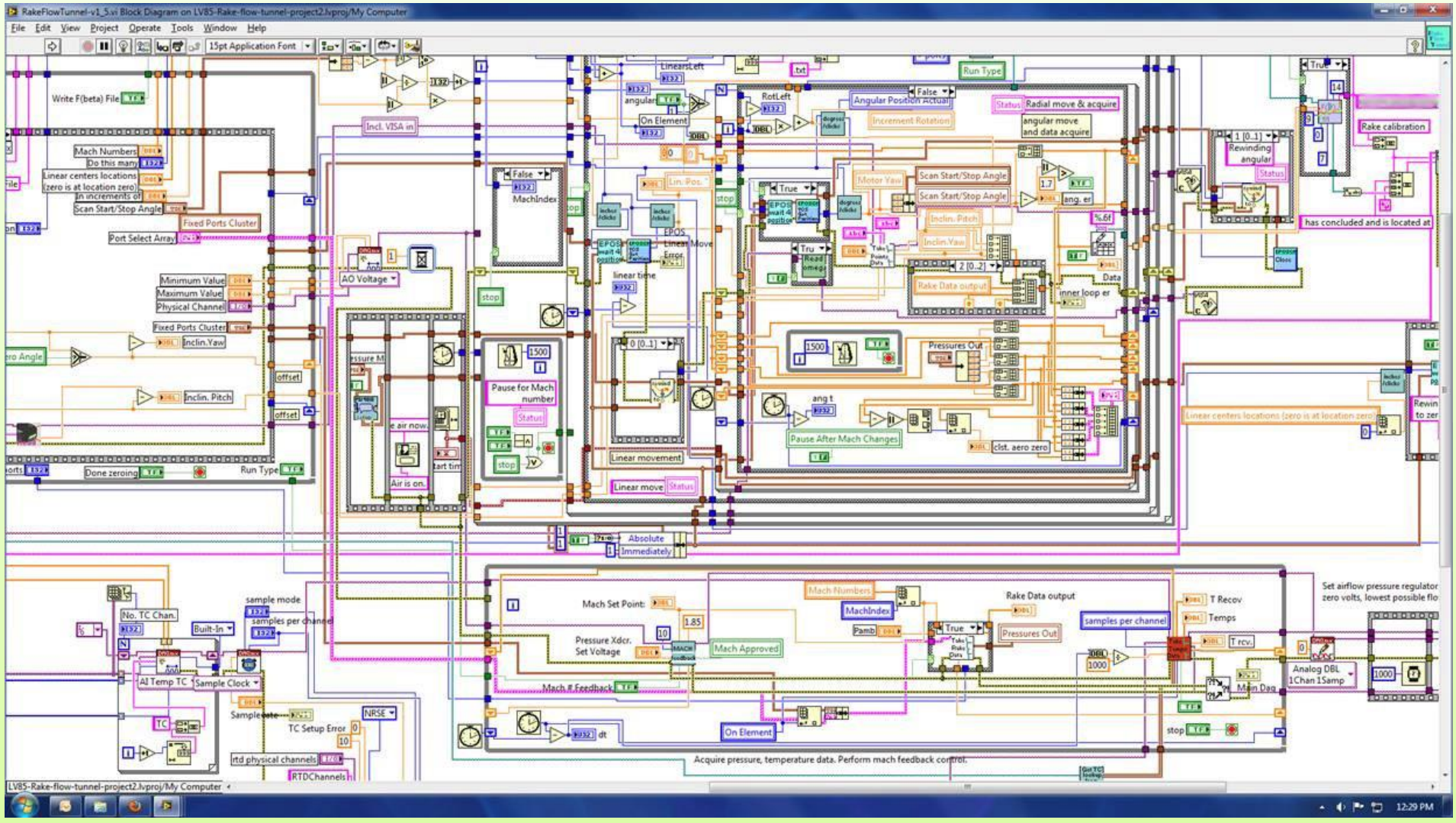
Principe

A l'image des sous-programmes en langage textuel, les sous-VI constituent les briques de base de la modularité : l'idée étant d'une part, de créer un sous-VI pour chaque portion de code qui définie une tâche spécifique avec ses entrées, ses sorties et son fonctionnement propre; et d'autre part de réutiliser ce sous-VI à chaque fois que la tâche en question est utilisée dans le VI « maître ».

Le sous-VI est en tout point identique à un VI, avec sa face-avant, l'icône et le connecteur, et son diagramme.

La modularité permet d'accéder à un niveau de programmation structurante et nécessaire au développement d'applications d'envergure certaine, en veillant à la lisibilité du code, sa « réutilisabilité », et surtout dans un premier temps afin d'éviter … ça ->

12. Notion de modularité (Suite)

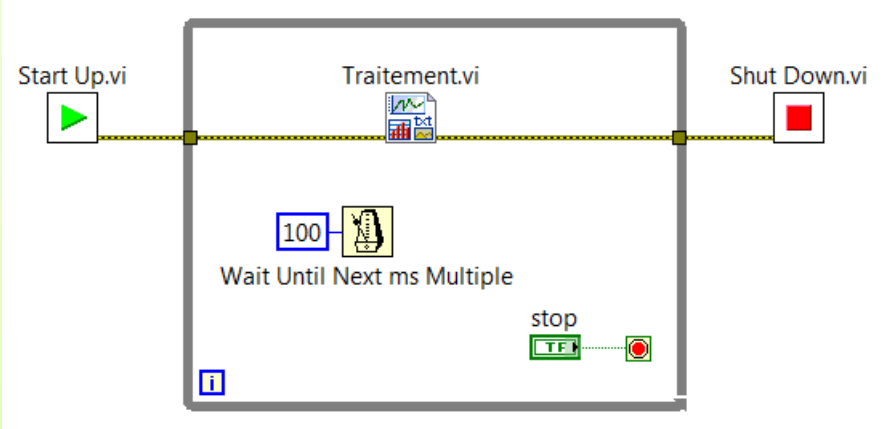


12. Notion de modularité

Suggestion

Le diagramme suivant représente ce que pourrait être l'architecture général d'un VI « maître » avec :

- ✓ Le sous-VI d'initialisation
- ✓ Une structure « tant que » cadencée avec un sous-VI de traitement contenant le cœur de votre application
- ✓ Le sous-VI de clôture de l'application



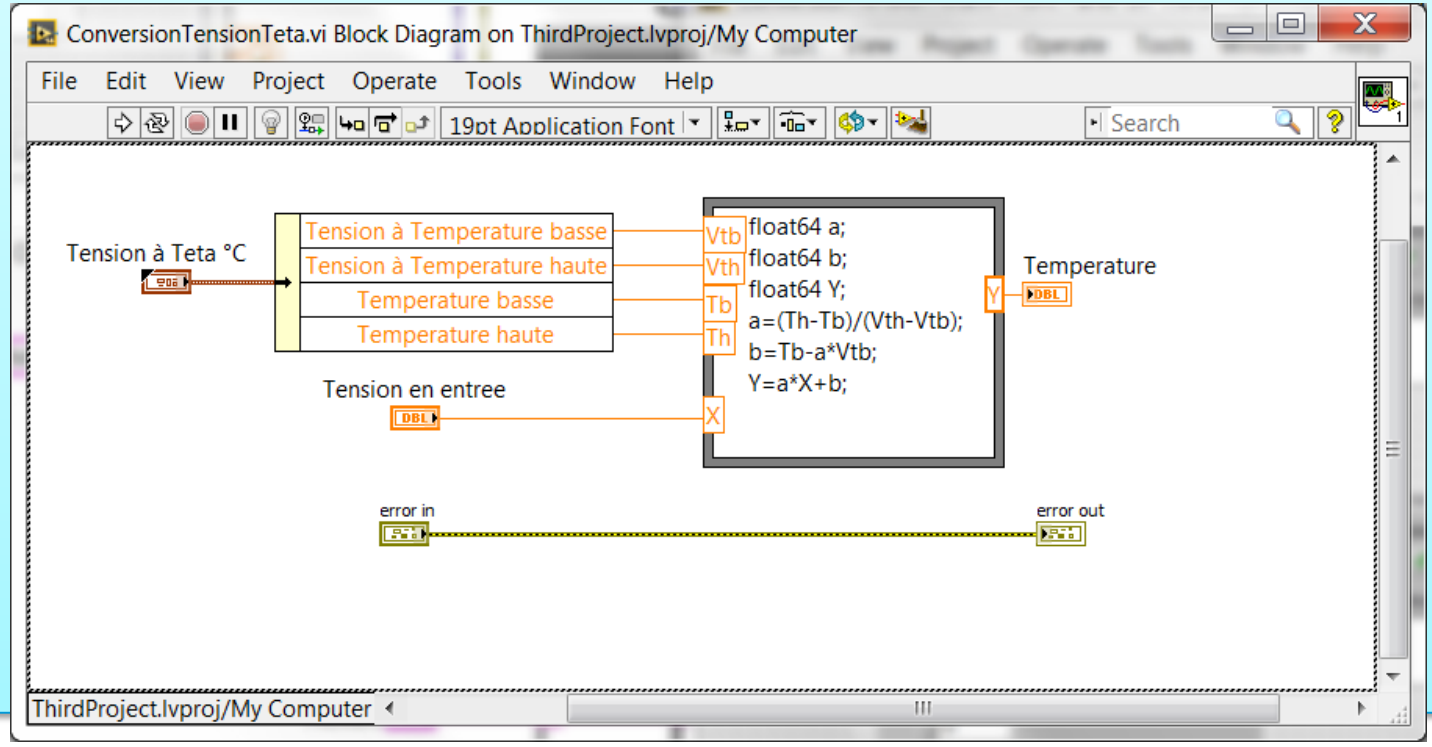
Le VI traitement s'exécute de manière itérative

Micro-Travaux Pratiques dirigés

XVII.Créer un sous-VI

Créer un sous-VI qui réalise une conversion d'une tension en la valeur de la grandeur physique, en intégrant le modèle de la fonction de transfert

Boucle conditionnelle sur le fil d'erreur !... => bonnes pratiques

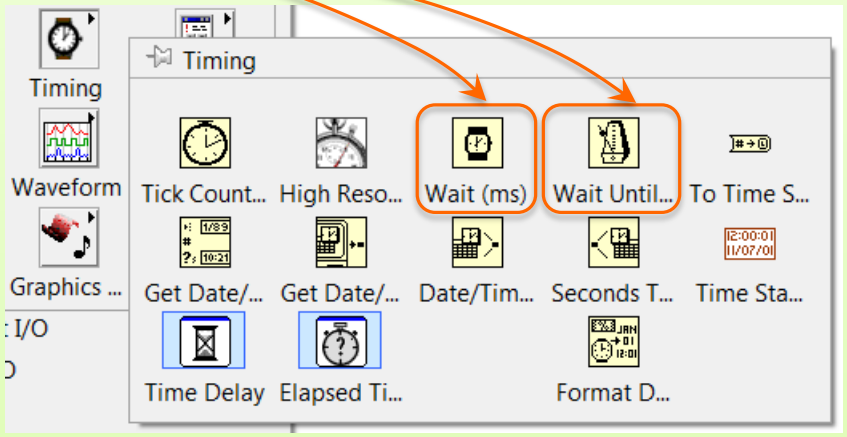


13. Notions de cadencement

13.1 Cadencement de l'exécution

Le fait d'imposer la fréquence à laquelle une boucle s'exécute, permet de donner du temps au processeur afin d'effectuer d'autres tâches, comme la gestion de l'interface utilisateur.

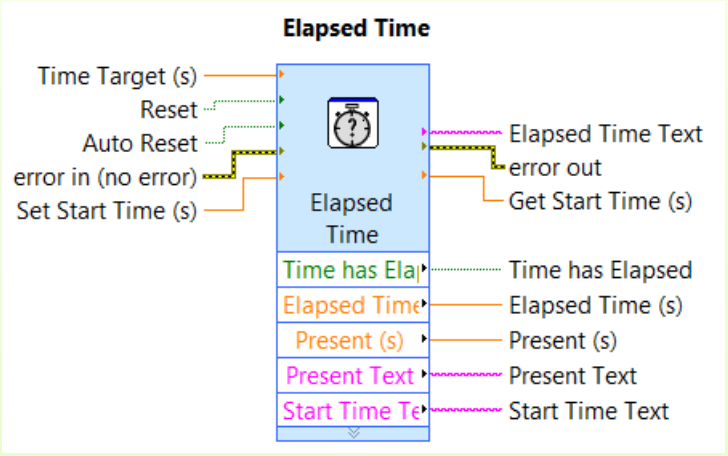
Utiliser les fonctions d'attente à l'intérieur d'une boucle permet d'atteindre cet objectif.



13. Notions de cadencement

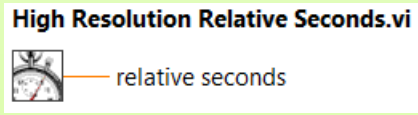
13.2 Cadencement contrôlé par « soft »

L'utilisation du VI Express « Temps écoulé » permet de détecter la fin d'un compte à rebours, de manière à exécuter une autre tâche.



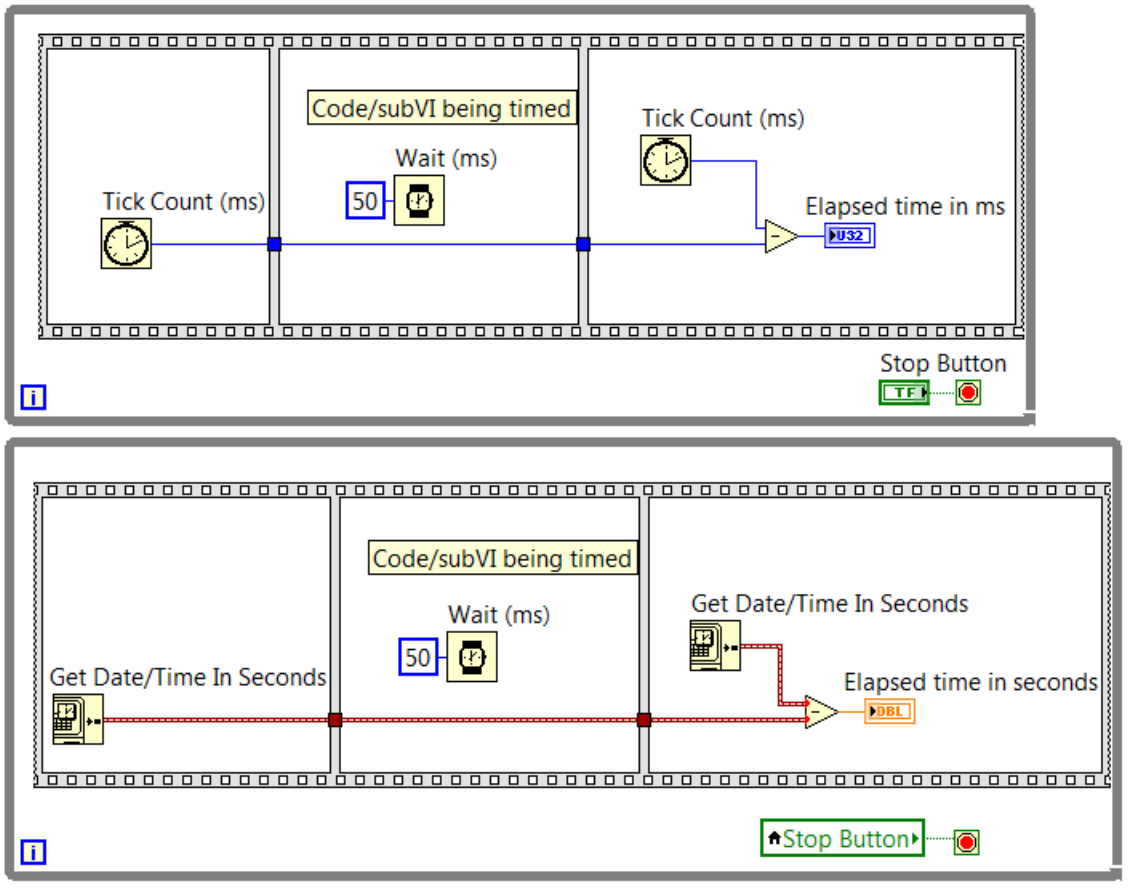
L'utilisation judicieuse de la fonction « Tick Count » (ms) permet, en outre, de quantifier la durée d'exécution d'une tâche dans une structure séquentielle.

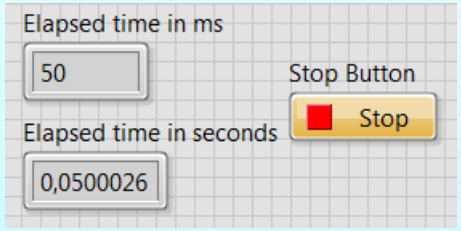
La fonction suivante permet elle de disposer de 14 digits de précision après-la virgule !

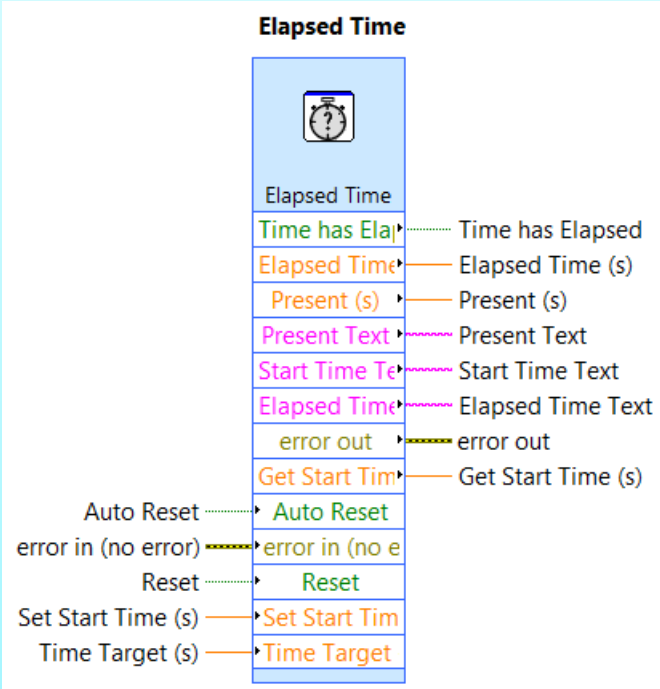


Micro-Travaux Pratiques dirigés

XVIII. Tester wait, wait until, tick count et elapsed time







14. Modèles de conception

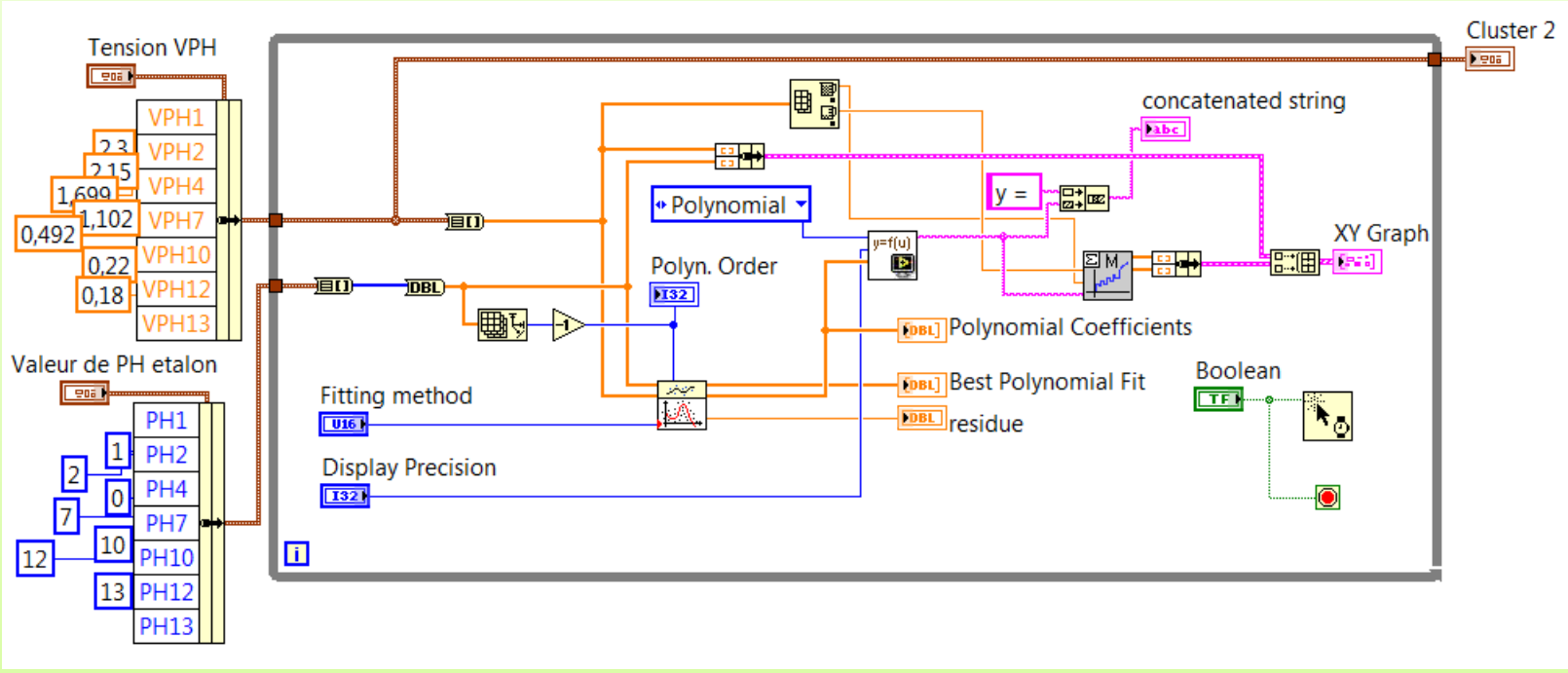
14.1 Principe

- **Ils représentent des techniques et des implémentations de code constituant une réponse à un problème spécifique de conception logicielle.**
- **Ils peuvent être utilisés comme point de départ d'une application.**
- **L'architecture du code ainsi développé devient plus facile à reconnaitre (par l'ensemble des intervenants développeurs) => maintenabilité accrue.**
- **LabVIEW intègre plusieurs exemples de modèles de conception.**

14. Modèles de conception

14.2 Architecture à boucle simple

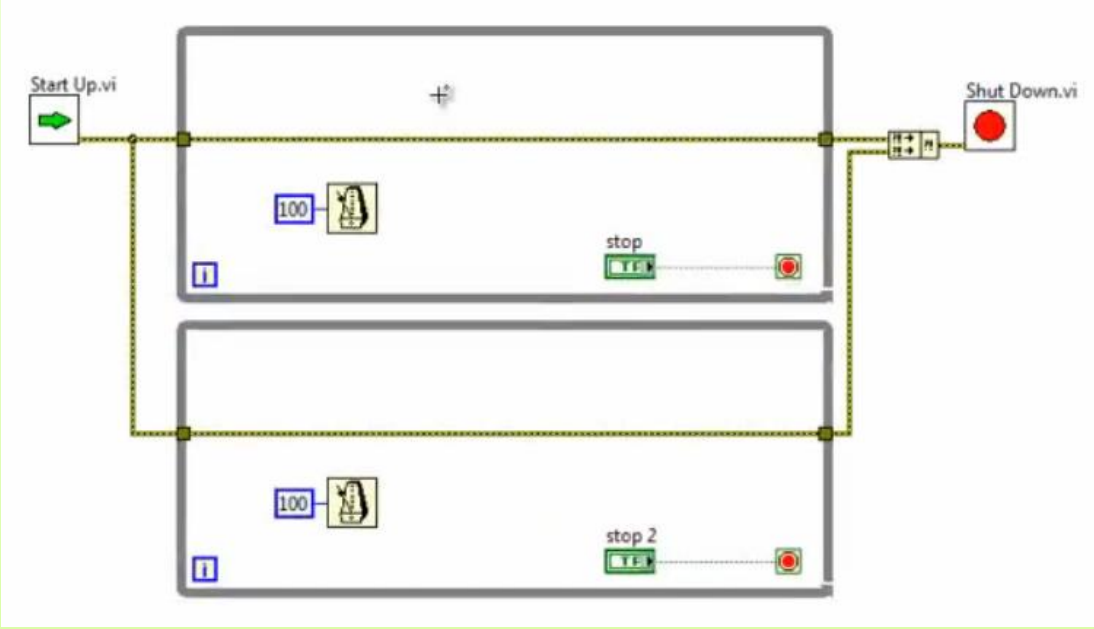
- **Elle utilise un seul VI pour une application simple telle qu'un calcul ou une mesure rapide.**
- **Elle peut devenir un sous-VI dans une application plus complexe.**



14. Modèles de conception

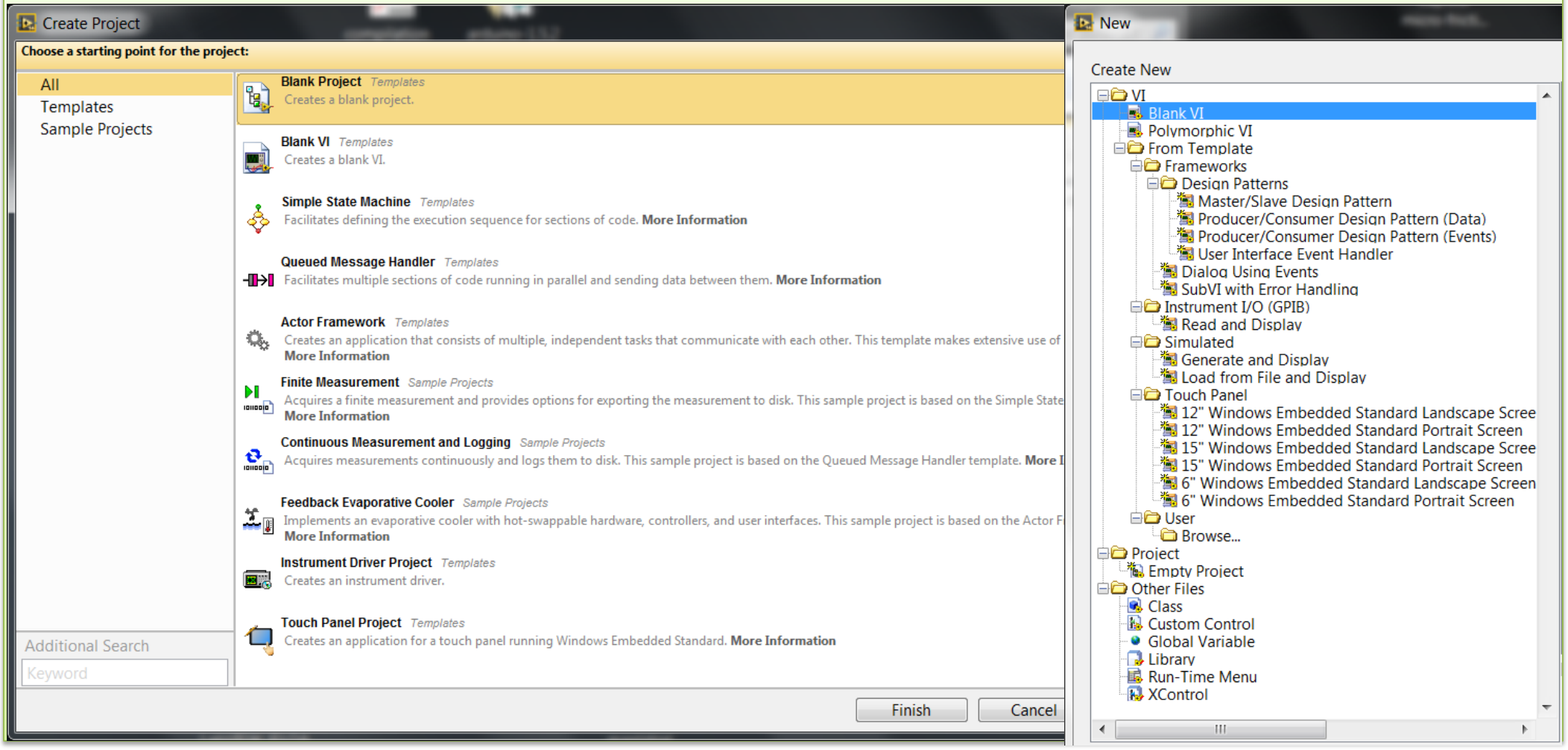
14.3 Architecture à boucles multiples

- **Chacune des deux boucles (ou plus) peut ainsi contenir un code, dont l'exécution est totalement indépendante de celle de l'autre code.**
- **En revanche, il est fondamental de cadencer judicieusement les boucles de sorte que le « processeur » puisse exécuter chacun des deux codes dans le timing imposé par le cadencement des deux boucles.**
- **Aucun échange de données n'est possible d'une boucle à l'autre …**



14. Modèles de conception

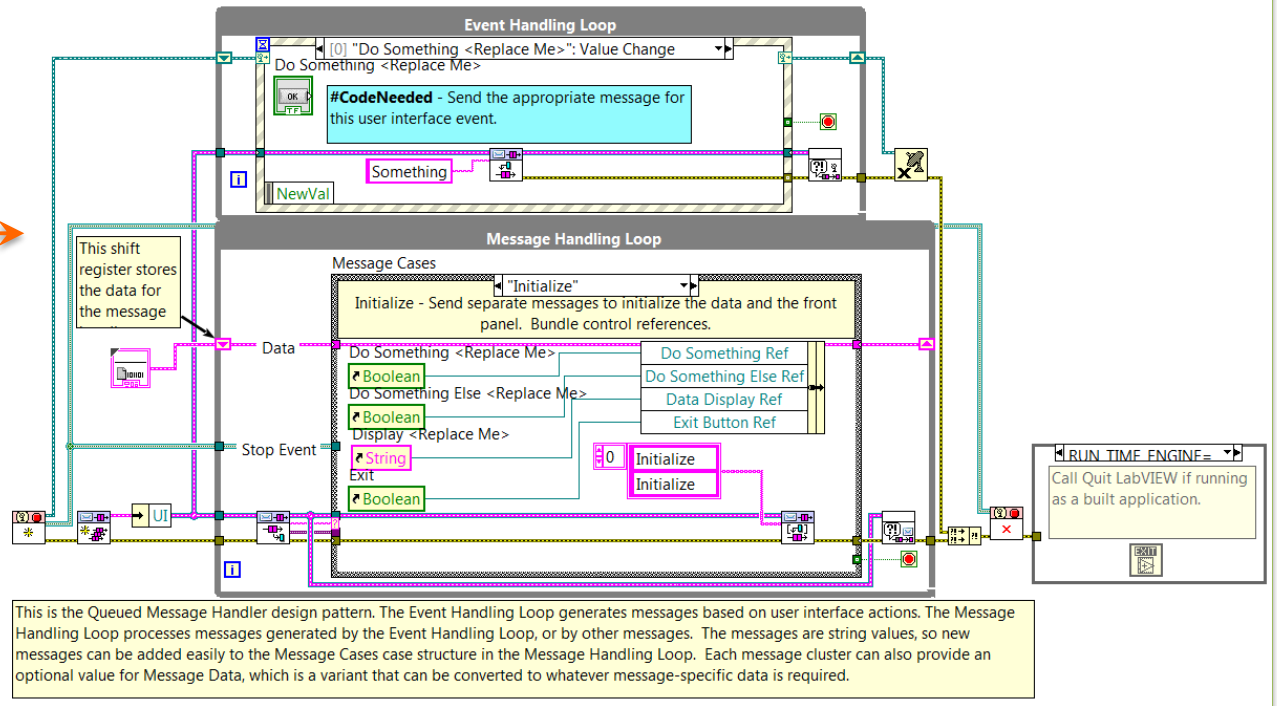
14.4 Création d'un nouveau projet à partir d'un modèle natif

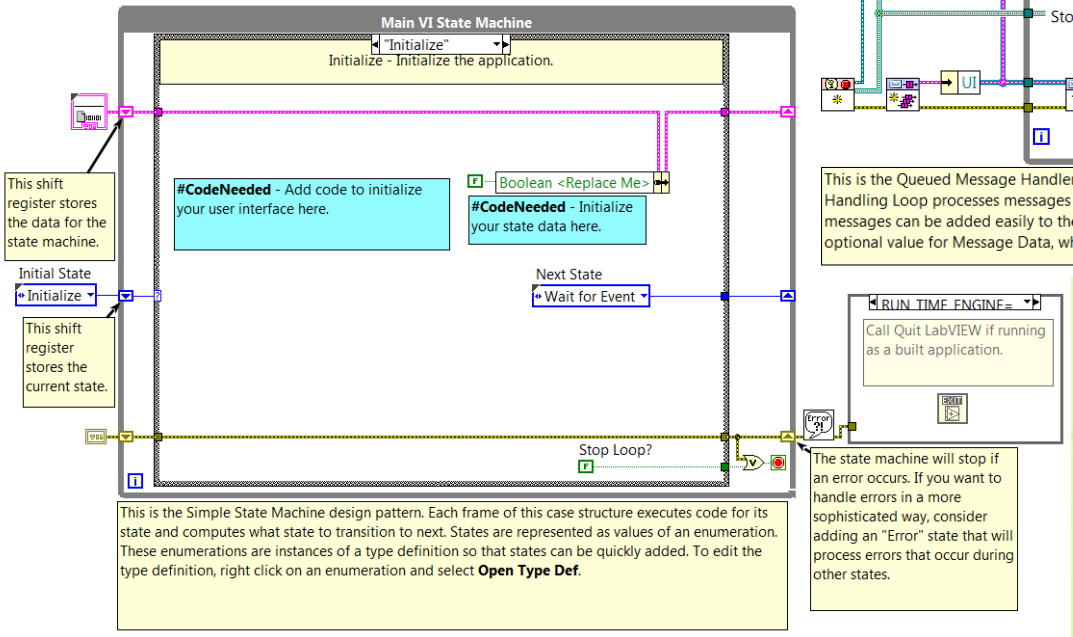


14. Modèles de conception

14.5 Deux exemples de modèles

Modèle évènementiel avec fil d'attente de messages





Modèle de machine à états

15. Machine à états

15.1 Principe

Constitué d'un début, d'états, de transitions et d'une fin, la machine à état permet de décrire un processus complet, en permettant au système modélisé de passer d'un état (de fonctionnement) à un autre en intégrant la notion de transition, qui peut être automatique, temporisée ou conditionnée par les entrées et/ou les sorties du système.

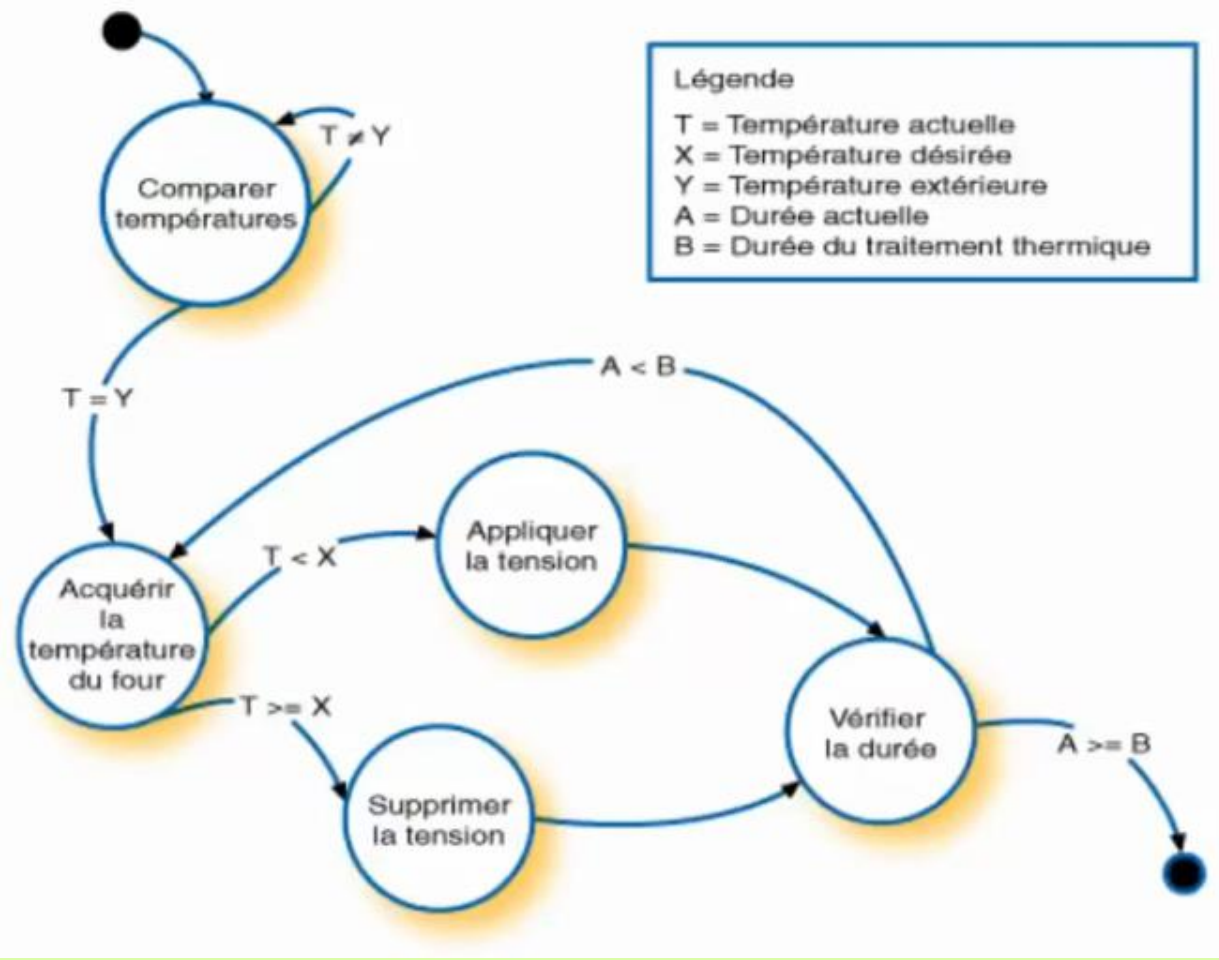
Un état s'apparente à un mode de fonctionnement spécifique caractérisé par les paramètres d'entrées et de sorties du système.

15. Machine à états

15.2 Exemple de processus

Ce diagramme d'état modélise le comportement d'un four.

Certaines transitions sont conditionnées, et certains états peuvent aussi disposer d'un retour sur état, lui-même conditionné.



15. Machine à états

15.3 Problématiques visées

- **Modifier l'ordre d'apparition d'un état.**
- **Répéter un état à plusieurs reprises**
- **Rendre une ou plusieurs transitions conditionnelles**
- **Mettre fin à la machine à état de manière prématurée**

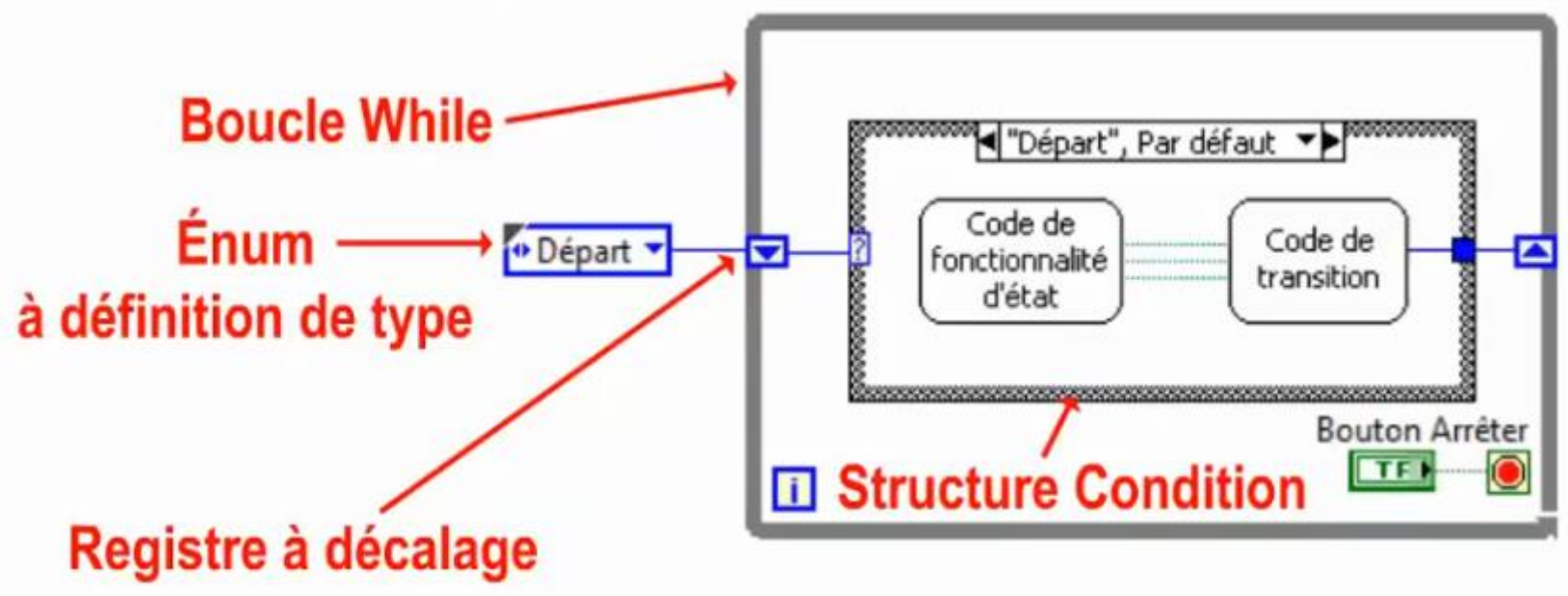
15.4 Usages courants

Les machines à états sont essentiellement utilisées au niveau de la conception des interfaces Homme/Machine, et également la conduite de processus machine.

15. Machine à états

15.5 Mise en œuvre et fonctionnement

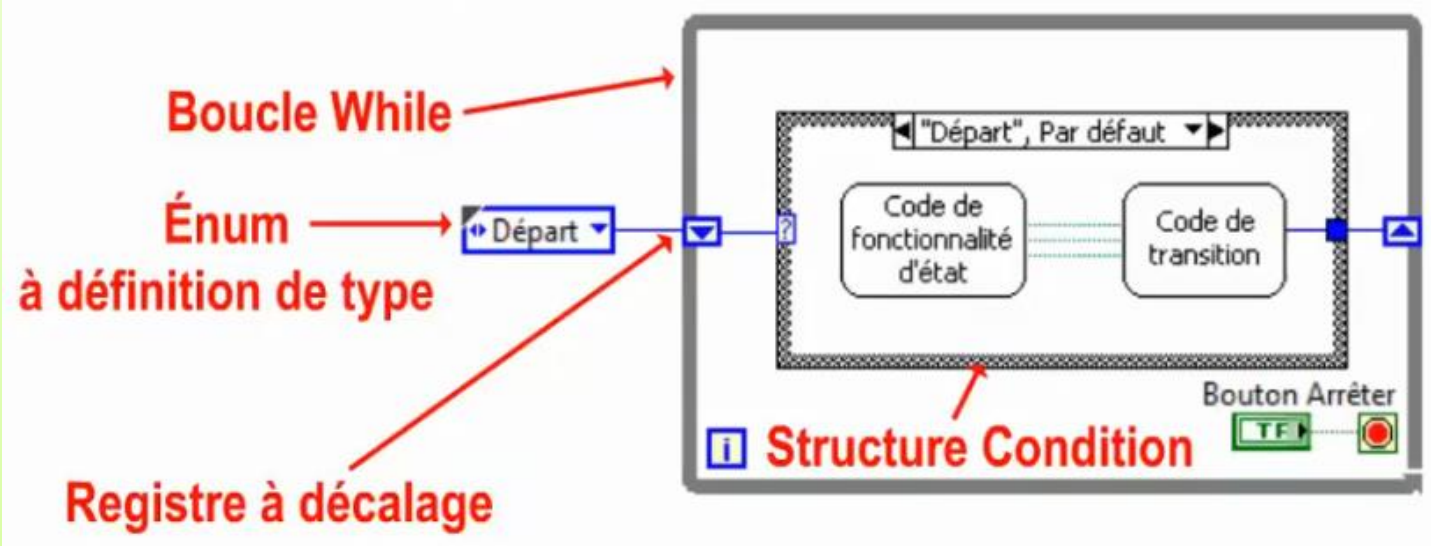
La structure de base repose sur une boucle While, associée à un registre à décalage et une boucle conditionnelle. Le « tout » est en règle général associé à une énumération, dont on a fait un « type def » par commodités ...



15. Machine à états

15.5 Mise en œuvre et fonctionnement (Suite)

L'énumération contient la liste des noms des états du système. Le registre à décalage permet de transférer de l'itération N à l'itération N+1 le nom de l'état suivant. Chacun des cas de la structure conditionnelle a pour condition le nom d'un des états de la liste de l'énumération, et tous les états font l'objet d'un cas de cette structure conditionnelle.

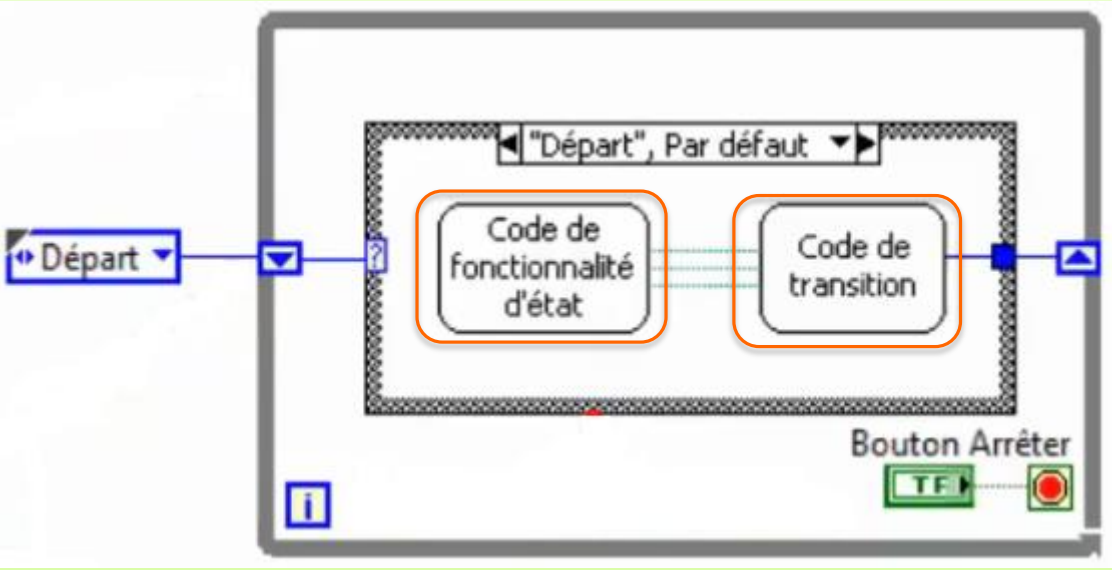


15. Machine à états

15.5 Mise en œuvre et fonctionnement (Suite)

Chacun des cas de la structure intègre d'une part le code lié aux fonctionnalités de l'état, et aussi un code de transition, qui dans sa plus simple expression est constitué d'une instance de l'énumération avec la sélection du nom de l'état suivant dans le diagramme.

L'exemple le plus simple à réaliser et tester consiste à définir quelques étapes avec transitions non-conditionnelles et d'intégrer dans chaque étape une boite de dialogue simple avec le numéro de l'étape en question.



16. Structure évènementielle

Principe

Un évènement est une notification qui stipule que quelque chose s'est produit en cours d'exécution d'un VI ou d'une application : ce peut être une interaction asynchrone de l'utilisateur avec la face-avant (IHM).

Le programme attend qu'un évènement se produise pour exécuter un code spécifique.

Les différents types d'évènements prennent les formes suivantes :

- **l'utilisateur interagit sur la face-avant via les périphériques d'entrées comme le clavier (appui sur touche) et la souris (click …)**
- **les entrées-sorties matérielles (E/S externes), ainsi que les trigger et les timer**

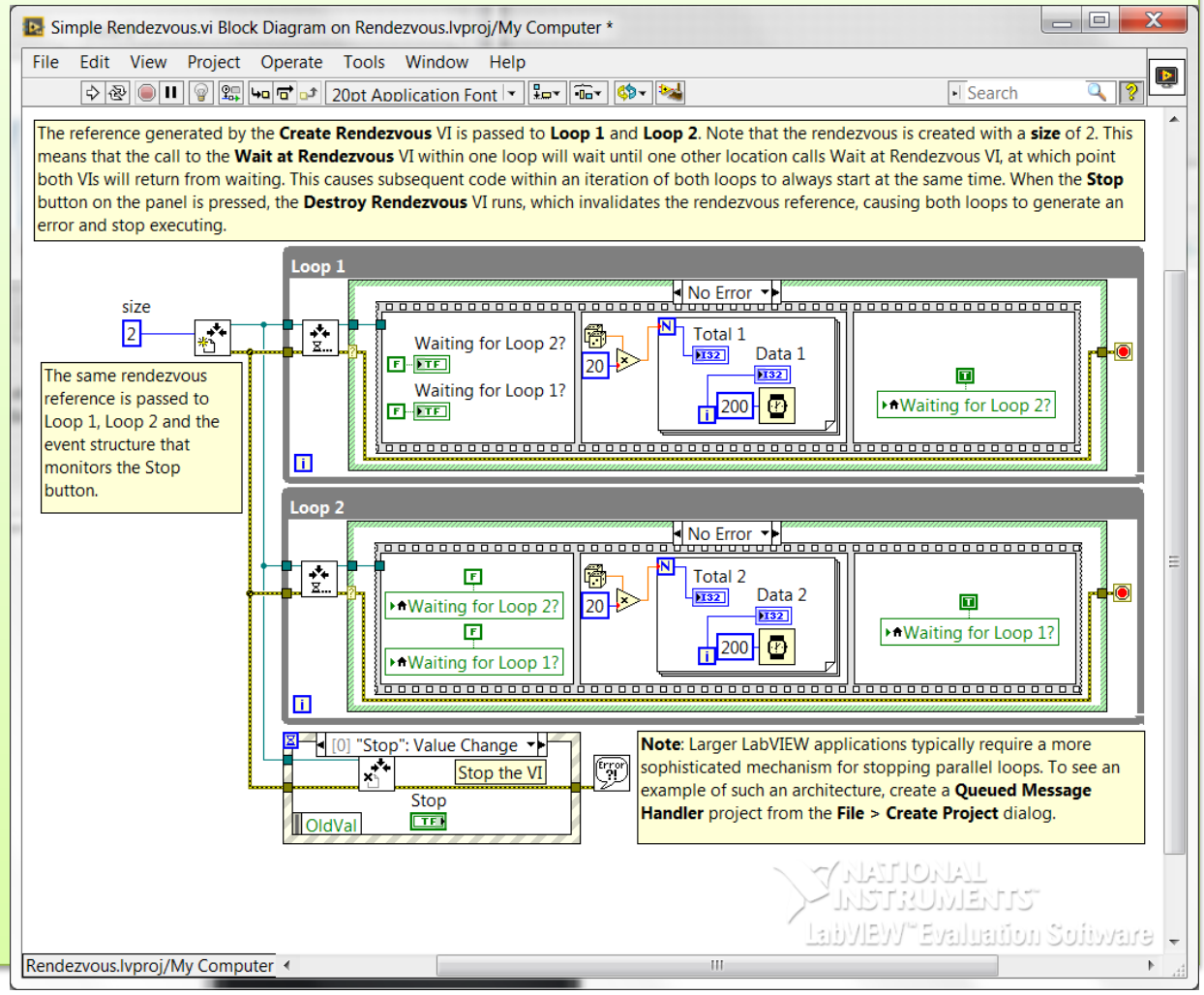
17. Outils de synchronisation

Principe Transmettre des informations entre processus parallèles

17.1 Rendez-vous

- Examples\Synchronisation\Rendezvous.lvproj

Synchroniser le début de traitements dans des boucles parallèles différentes et pour chaque itération

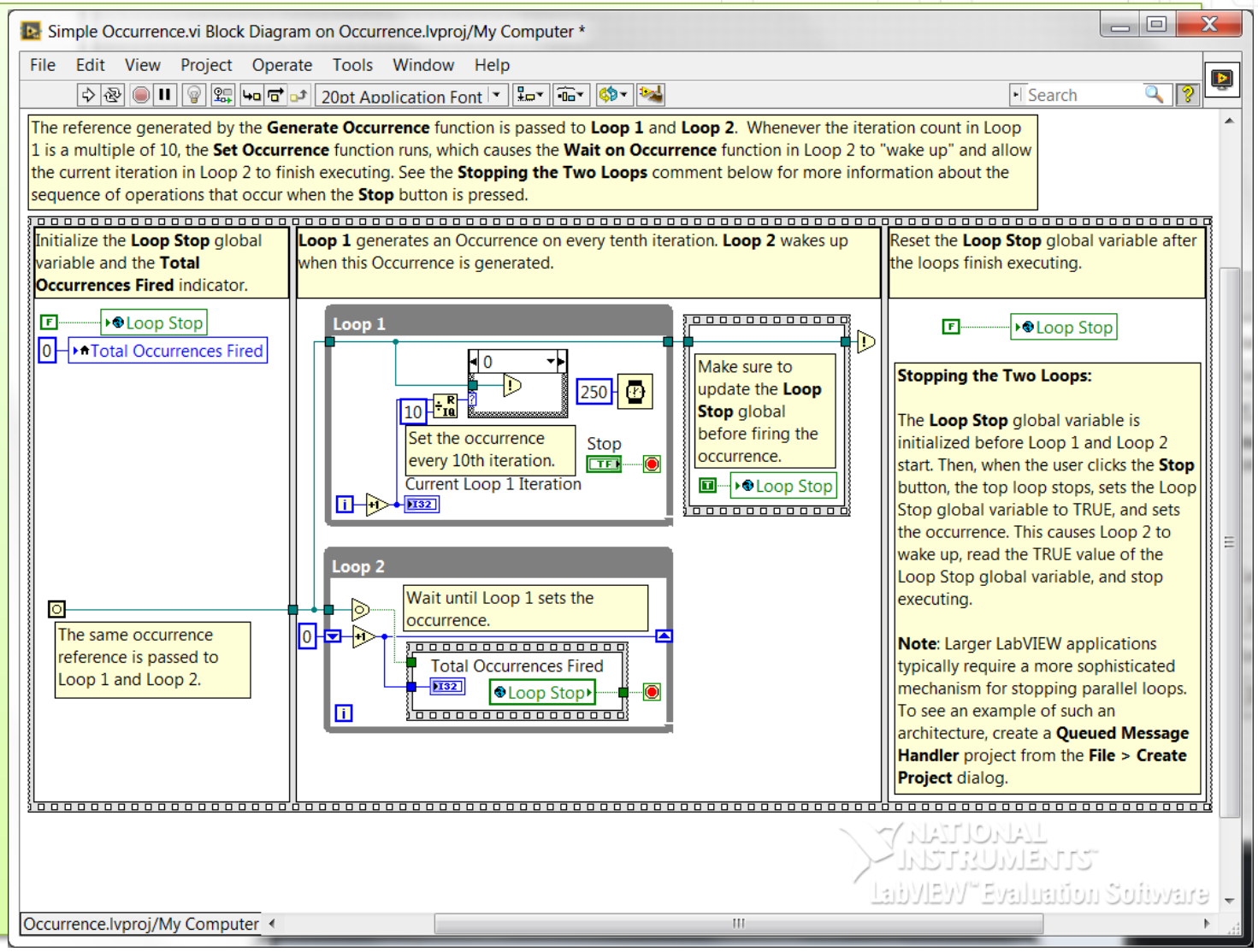


17. Outils de synchronisation

17.2 Occurence

- Examples\Synchronisation \Occurrence.lvproj

Déclencher l'exécution d'un traitement d'une boucle à partir d'une autre boucle;
La seconde boucle n'est pas temporisée.

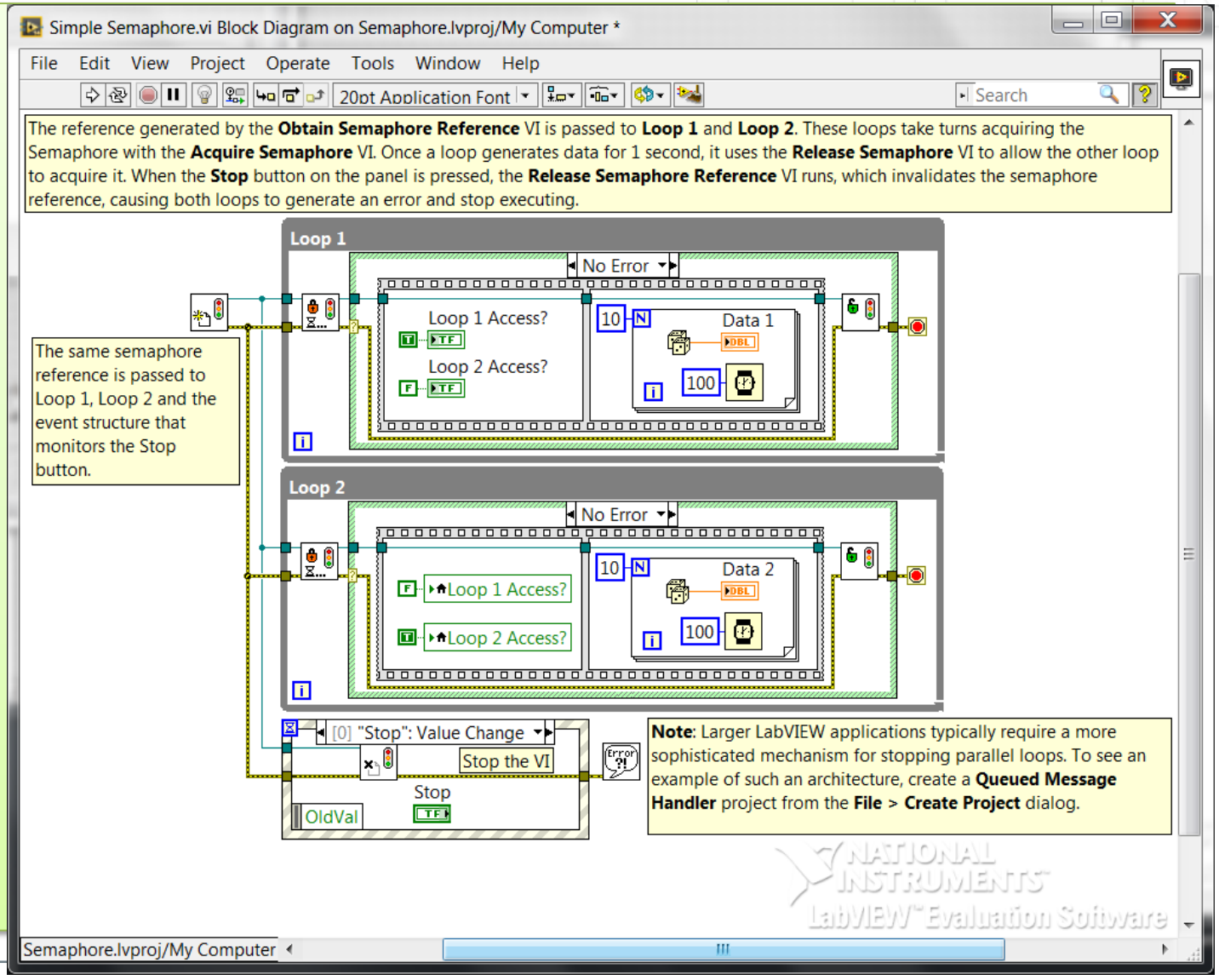


17. Outils de synchronisation

17.3 Sémaphore

- Examples\Synchronisation\Semaphore.lvproj

Permet l'exécution du traitement de boucles les unes après les autres sans perte de temps lié à un cadencement dans une ou plusieurs boucles. Fonctionnement sur le principe du jeton que l'on distribue à une seule entité. Les processus s'enchainent sans perte de temps.

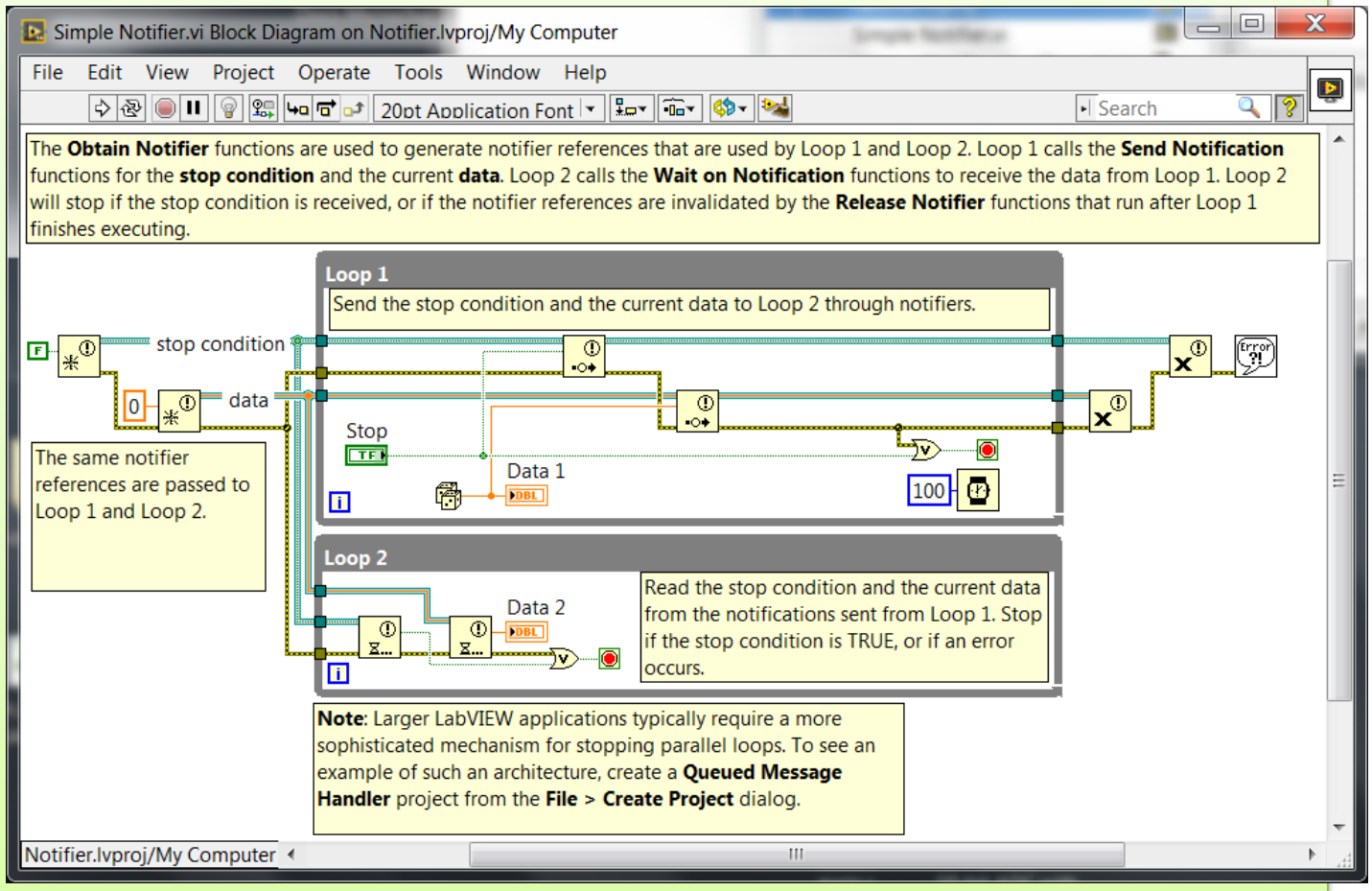


17. Outils de synchronisation

17.4 Notificateur

- Examples\Synchronisation\Notifier.lvproj

Transmission d'une information d'une boucle à une autre avec la notion d'attente dans la seconde boucle.
Le traitement de la seconde boucle doit s'exécuter en un temps inférieur à l'espace qui sépare l'envoi de deux notificateurs successifs.
Les notificateurs ne sont pas mémorisés : les pertes sont donc possibles.



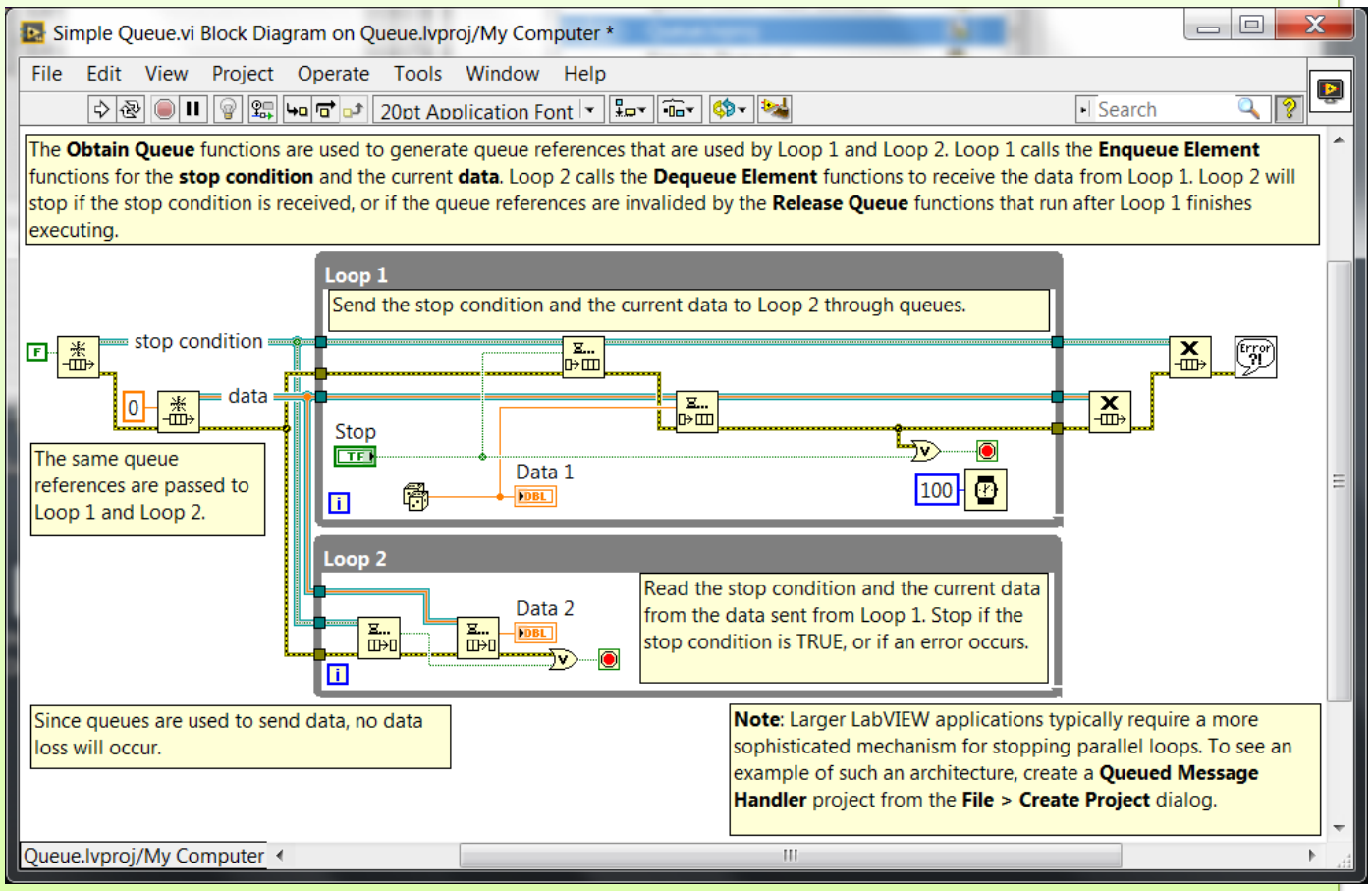
17. Outils de synchronisation

17.5 File d'attente

- Examples\Synchronisation\Queue.lvproj

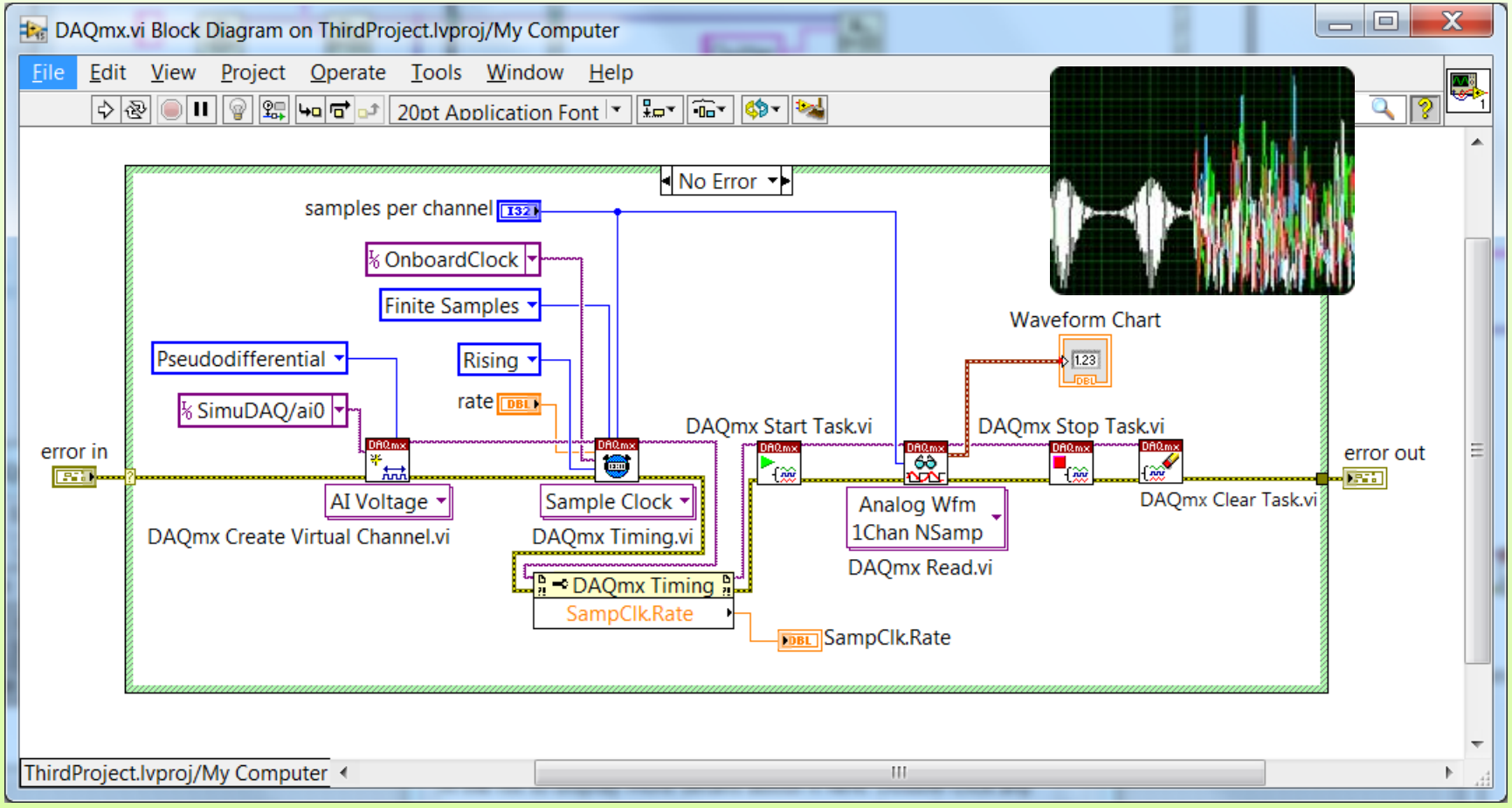
Modèle Producteur/Consommateur

Transmission d'une information d'une boucle à une autre avec la notion d'attente dans la seconde boucle.
Ici, les éléments placés en file d'attente sont mémorisés dans une pile de type FIFO.



18. Acquisition DAQmx

Principe



Petit recueil de bonnes pratiques

Rapidité de développement

- **Utiliser les modèles de conception**
- **Utiliser les librairies existantes**
- **Utiliser les « Type Definition »**
- **Utiliser les raccourcis-clavier**
- **Pour le placement des objets du diagramme, utiliser le « Quick drop »**

Modularité du code

- **Définir les sous-VI**
- **Structurer les variables dans des clusters**

Petit recueil de bonnes pratiques

Performance du code

- **Refermer les références**
- **Pas de nœuds de propriété sur les indicateurs**
- **Placer le terminal des indicateurs à l'endroit où le rafraichissement doit avoir lieu**
- **Identifier la nécessité de définir un timeout sur les structures évènementielles (notion de scrutation ou polling)**

Lisibilité du code

- **Aligner les fonctions sur le fil d'erreur**
- **Organiser la lecture de gauche à droite**
- **Placer les contrôles à gauche et et les indicateurs à droite dans les sous-VI**
- **Retravailler l'icône des sous-VI**

Petit recueil de bonnes pratiques

Documentation du code

- **Intégrer un descriptif dans les propriétés des VIs**
- **Faire apparaitre le label des sous-VIs**
- **Faire apparaitre le label des structures (While, …)**
- **Faire apparaitre le label des fonctions**
- **Faire apparaitre un label sur les fils de liaison**

Documentation de l'IHM (Face-avant)

- **Nommer les contrôles et indicateurs de manière judicieuse**
- **Créer des info-bulles sur les objets**

Petit recueil de bonnes pratiques

Gestion d'erreur

- **Le fil d'erreur traverse bien tous les nœuds**
- **Les sous-VI doivent disposer d'un connecteur d'entrée associé à un cluster « entrée d'erreur » et un connecteur de sortie associé à un cluster « sortie d'erreur »**
- **Le VI Main se termine par l'exécution du gestionnaire d'erreur simple**

femto-st SCIENCES & TECHNOLOGIES

Merci de votre attention …

LabVIEW

Développement de **S**ystèmes Modulaires d'**I**nstrumentation et de **C**ontrôle/commande **E**mbarqués …

UFC Université de Franche-Comté — cnrs — ensmm — utbm université de technologie Belfort-Montbéliard