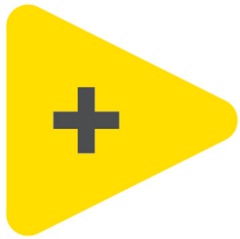
CHARLES CARNUS
LYCÉE TECHNIQUE
ENSEIGNEMENT SUPÉRIEUR

TD : Instrumentation programmable

Prof. : K. Boudjelaba



LabVIEW™

BTS SN–1 2022–2023

TD1 – Initiation à LabVIEW : VI et sous VI

1 Objectif du TD

Ce TD ne présente aucune difficulté scientifique et permet de se familiariser avec l'environnement LabVIEW pour des applications de calculs scientifiques simples.

2 Conversion temporelle

On souhaite dans un premier temps pouvoir convertir une durée définie en heures, minutes et secondes en une durée en secondes.
La formule permettant de le faire à la main est très simple. Si l'on note h, m et s respectivement le nombre d'heures, de minutes et de secondes on a alors :
$s = s + m * 60 + h * 3600 = s + 60 * (m + 60 * h)$
Dans la première partie du TD nous allons créer un VI (programme LabVIEW) qui pourra être utilisé ensuite en sous VI (sorte de fonction réutilisable dans un autre VI).
Ce VI devra pouvoir réaliser cette opération en prenant en entrée trois commandes définissant respectivement h, m et s et en fournissant en sortie un affichage du nombre de secondes.

1. Lancer LabVIEW et créer un nouveau VI vide.
2. Sélectionner Fenêtre / Mosaïque verticale pour afficher côte à côte la face avant et la face arrière.

2.1 Mise en place du Front Panel (face avant) : Commandes et affichages

La face avant est la fenêtre avec une grille qui permet le positionnement des éléments graphiques. Sur la face avant :

1. Créer autant de contrôles numériques que nécessaires pour entrer les valeurs de temps. Il faudra placer ces contrôles et les renommer pour faciliter leur utilisation. Pour cela faites un clic droit sur la face avant et cherchez la boite "commande numérique".
2. Créer un indicateur numérique pour afficher le résultat de la conversion. Renommez-le également. Pour cela faites un clic droit sur la face avant et cherchez la boite "indicateur numérique".

Vous remarquerez que LabVIEW crée les contrôles et les indicateurs correspondant sur le diagramme de face arrière. Les terminaux (boites) représentent le type de données des contrôles et des indicateurs.
Par exemple, un terminal de type DBL représente un contrôle ou un indicateur de type flottant avec double précision.

2.2 Mise en place du Block Diagram (face arrière) : Diagramme de fonctionnalité

Sur la face arrière, nous allons coder la fonction désirée. Le diagramme va se présenter de façon visuelle à la façon de portes logiques à connecter :

1. Placer autant d'opérateurs "multiplier" et "additionner" que nécessaires pour réaliser la fonction :
   $s = s + m * 60 + h * 3600$ ou $s = s + 60 * (m + 60 * h)$
2. Placer et initialiser autant de constantes que nécessaires.
3. Connecter les boites
4. Sauvegarder votre VI (nous le réutiliserons).

2.3 Test de votre VI

Sur la face avant :

1. Entrer des valeurs dans vos commandes numériques.
2. Lancer le VI avec la commande "Exécuter" (flèche blanche).
3. Vérifier le résultat.
   Avec ce mode d'exécution, il est nécessaire de relancer l'exécution à chaque test.
4. Lancer le VI avec la commande "Exécuter en continu"
   Avec ce mode, le programme se relance automatiquement à la fin de son exécution à la façon d'une boucle infinie.
5. Sur le diagramme (face arrière), sélectionnez l'option "Animer l'exécution" l'ampoule doit s'allumer.
6. Lancer le VI et observer la face arrière.
   Les points orange qui se déplacent représentent les données qui se propagent dans le diagramme. Il est important de comprendre que l'exécution d'un VI se fait à la façon d'un flux de données (de façon parallèle) et non de façon séquentielle. Chaque "boite" attend que l'ensemble de ses entrées soient disponibles avant de lancer l'exécution.
7. Lancer le VI de façon détaillée (Exécution/Exécuter de façon détaillée).
8. Passer les différentes étapes avec les trois dernières icônes (exécuter xxx sans détailler / exécuter xxx en détaillant / finir l'exécution).

2.4 Empaquetage du VI en sous VI

Ce code pourra être réutilisé par la suite comme une "boite noire". Pour rendre sa réutilisation plus simple nous allons créer une apparence pour cette boite : une icône.

1. Faites un clic droit sur l'icône située dans le coin supérieur droit sur la face avant et sélectionnez "Editer l'icône" dans le menu. La boîte de dialogue "Editeur d'icône" apparaît avec l'icône par défaut.
2. Vous pouvez dessiner une icône à la façon d'un éditeur d'image classique.
3. Faites une icône représentative pour cette fonction. Pour cela vous pouvez utiliser des symboles préexistants disponibles sous l'onglet symboles.
4. Quand l'icône est terminée, cliquez sur "OK" pour fermer la fenêtre de dialogue de l'éditeur d'icônes.
   L'icône apparaît dans le coin supérieur droit de la face avant et du diagramme de face arrière.

Á ce stade vous avez créé une apparence, pour rendre cette "boite" fonctionnelle, il faut définir ses entrées et sorties :

1. Sur la face avant, à gauche de l'icône se trouve le schéma d'implantation des terminaux. Un clic droit permet d'accéder à différents "Modèles".
2. Choisissez un modèle à 3 entrées (à gauche) et une sortie (à droite).
3. Affectez les terminaux aux contrôles et indicateurs numériques. Pour cela, cliquez d'abord sur une des cases du terminal puis sur la commande/indicateur numérique que vous souhaitez attribuer.
4. Sélectionnez Fichier / Enregistrer pour sauvegarder ce VI.

2.5 Réutilisation du VI comme sous VI

1. Fermer ce VI et en créer un nouveau.
2. Sur la face arrière (diagramme) du VI, ajouter un VI existant "Sélectionner un VI..."
3. Positionner le VI et relier le à trois commandes numériques et à un indicateur numérique.
4. Tester votre VI. On souhaite ajouter les jours dans la conversion, c'est à dire convertir : jours, heures, minutes et secondes en secondes.
5. Ajouter sur ce VI la commande numérique pour les jours et les opérations nécessaires pour compléter le VI.

Á ce stade vous savez créer des programmes simples et les réutiliser dans d'autres programmes. Sous LabVIEW, il existe énormément de programmes déjà codés et utilisables sous la forme de "boites" comme celle que vous avez créé.

TD2 – Initiation à LabVIEW : Gestion des boucles

1 Objectif du TD

Ce TD ne présente aucune difficulté scientifique et permet de se familiariser avec l'environnement LabVIEW pour des applications de calculs scientifiques simples.

2 Génération aléatoire de nombres

On souhaite dans un premier temps pouvoir générer des nombres de façon aléatoire jusqu'à ce que la somme de ces nombres soit supérieure ou égale à une valeur donnée par l'utilisateur.

1. Lancer LabVIEW et créer un nouveau VI vide.
2. Sélectionner Fenêtre / Mosaïque verticale pour afficher côte à côte la face avant et la face arrière.

2.1 Mise en place de la face avant : Commandes et affichages

1. Créer un contrôle (commande) numérique pour entrer la valeur cible, c'est à dire la valeur minimale à atteindre. Á renommer pour faciliter l'utilisation.
2. Créer autant d'indicateurs numériques pour afficher le résultat de la somme mais également le dernier nombre aléatoire tiré. Renommez-les également.
   On souhaite que ces commandes et indicateurs soient de type double (DBL) mais n'affichent que la valeur entière. De plus on veut que l'utilisateur ne puisse pas demander une valeur supérieure à 500.
3. Entrer dans les propriétés de la commande "valeur cible".
4. Chercher comment limiter la plage de valeurs possibles.
5. Entrer dans les propriétés des indicateurs numériques. Veiller à ce que seule la partie entière du résultat s'affiche (regarder l'onglet Format et Précision)
   Proposer une méthode et tester que votre commande et vos afficheurs respectent ces consignes.

2.2 Mise en place de la face arrière : Diagramme de fonctionnalité

Sur la face arrière, nous allons coder la fonction désirée. Il va falloir réaliser une boucle qui à chaque itération génèrera un nombre aléatoire, l'affichera et l'ajoutera à la somme. Si la somme dépasse la valeur cible, le système doit s'arrêter et afficher la somme ainsi que le nombre d'itérations qui a été nécessaire.

1. Créer une boucle de type "While"
2. Placer un bloc de génération aléatoire de valeurs.
3. Placer et initialiser autant de constantes que nécessaires.
   Il est à noter que le générateur de nombres aléatoires génère une valeur entre 0 et 1. Nous souhaitons générer une valeur entre 0 et 10. Pour cela il vous faudra utiliser un multiplieur ainsi qu'un module faisant l'arrondi. De plus, pour conserver une valeur d'une itération à l'autre, il est nécessaire d'utiliser un registre à décalage. Pour l'ajouter, il faut faire un clic droit sur le contour vertical de la boucle. Ce registre va fonctionner comme suit : si à l'itération k, vous mettez une valeur x sur le registre de droite, à l'itération $k+1$, le registre de gauche prendra la valeur x. La toute première valeur utilisée à la première boucle doit être donnée depuis l'extérieur de la boucle.
4. Connecter les boites.
   La condition de fin de boucle est représentée par le petit rond rouge. C'est à ce point rouge que doit être connecté votre test de fin d'opération (ici la comparaison de votre somme avec la valeur cible).
5. Sauvegarder votre VI.

2.3 Test de votre VI

Sur la face avant :

1. Entrer la valeur cible dans votre commande numérique.
2. Lancer le VI avec la commande "Exécuter" (flèche blanche).
3. Vérifier le résultat.
   L'exécution du programme est trop rapide, il faut mettre dans la boucle une temporisation pour ralentir l'exécution.
4. Ajouter un temporisateur de 1 seconde dans la boucle.
5. Tester votre programme.
6. Modifier votre programme jusqu'à ce qu'il marche.

Á ce stade vous savez utiliser la boucle "While". Il est également intéressant de savoir utiliser les boucles "For".

3 Calcul du Nième terme d'une suite

L'objectif est de calculer les N premiers termes de la suite suivante :

$$s[n+1] = 3 * s[n] - 3$$
$$\text{avec } s[0] = 2$$

1. Créer un affichage pour afficher le résultat s[n] de l'itération courante.
2. Tester pour différentes valeurs de N.
3. Comme précédemment, ajouter une temporisation pour pouvoir visualiser les résultats.
4. Remplacer la boucle "For" par une boucle "While" de façon à ce que le programme soit fonctionnel et donne un résultat identique.

TD3 – Programmation de calculs mathématiques simples

1 Objectif du TD

Utiliser les notions précédentes pour réaliser un programme plus complexe.

2 Résolution d'une équation du second degré

On se propose de réaliser un VI qui donnera la solution d'une équation du second degré du type : $ax^2 + bx + c = 0$. Avec a, b, et c des commandes numériques. Le discriminant ainsi que les deux racines devront être retournés comme affichage.

1. Mettre en place la face avant
2. Faire le calcul du déterminant
3. Utiliser une structure "Case" pour déterminer l'action à mener en fonction du signe du discriminant.
4. Calculer pour le cas positif ou nul la valeur des racines du polynôme.
5. Faites de même pour le cas négatif.

3 Utilisation des boites à calcul

LabVIEW est capable de calculer des formules littérales grâce à l'outil boite à calcul. Résoudre l'exercice précédent en utilisant la boite à calcul.

1. Mettre en place les entrées numériques et les indicateurs.
2. Créer une boite à calcul
3. Ajouter des entrées et des sorties sur la boite à calcul.
4. Ecrire les calculs nécessaires.
5. Proposer un VI pour gérer les trois cas avec ce système.

TD4 – Tableaux, clusters et matrices

1 Objectif du TD

Acquérir des notions sur les tableaux, clusters et matrices.

Remarque 1. Sous LabVIEW, un tableau contient un nombre N d'éléments du même type (booléen, entier, double, affichage etc.). Une matrice ne peut contenir que des variables de type numérique. Un cluster quant à lui peut regrouper des éléments de tous les types en son sein.
Nous allons tenter de manipuler ces différentes structures de données.

2 Matrices : Résolution d'un système d'équations à N inconnues

On veut résoudre un système de trois équations à trois inconnues grâce aux propriétés des matrices.
Les cours de mathématiques vous ont appris comment utiliser les propriétés matricielles pour résoudre des systèmes.

Soit le système suivant : $\begin{cases} 4x + 2y - z & = 2 \\ x + 4y + 2z & = 12 \\ 0.1x + y + 2z & = 10 \end{cases}$

1. Mettre le système sous forme matricielle.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad \Leftrightarrow \quad A * X = B$$

2. Les matrices A et B doivent être des entrées du système. Sous LabVIEW, créer une matrice de taille 3x3 et un vecteur de taille 3x1.
3. Initialiser ces matrices avec les bonnes valeurs.
4. Résoudre le système (en utilisant deux méthodes différentes) et trouver les valeurs de x, y et z.

3 Tableaux d'entiers et de booléens : Chenillard lumineux

L'objectif va être d'entrer une valeur numérique N entre 0 et 10 et de faire allumer les N premières LEDs successivement.

1. Créer une commande numérique entière qui peut prendre des valeurs entre 0 et 10.
2. Créer un tableau contenant 10 indicateurs booléens.
3. Créer un tableau contenant 10 indicateurs numériques.
4. Créer une boucle "For" qui comptera de 0 à la valeur entrée par la commande numérique. Intégrer une temporisation de 500 ms.
5. Initialiser le tableau de 10 indicateurs numériques à 0.
6. Á chaque itération de la boucle, initialiser à 1 la case du tableau numérique qui correspond à l'indice de boucle.
7. Chercher comment convertir ce tableau de valeurs numériques en tableau de booléens.
8. Faites afficher sur les diodes (LEDs) les résultats du VI.
9. On souhaite que les diodes précédentes s'éteignent dès lors que la nouvelle s'allume. Proposer un nouveau VI pour réaliser cette application.

4 Clusters : de la commande à l'affichage

Nous allons voir comment manipuler les clusters.

1. Créer un cluster sur la face avant. Ajouter à l'intérieur de ce cluster une commande numérique, une commande chaîne de caractère et une commande booléenne.
2. Sur le diagramme, une seule boite apparait, elle contient les trois commandes. Créer un indicateur pour afficher le résultat en sortie du cluster. On souhaite faire des opérations sur les différentes valeurs d'entrée.
3. Décomposer le cluster grâce à l'outil "Unbundle by name" (désassembler par nom).
4. Faire les opérations suivantes :
   - ajouter 5 à la valeur numérique,
   - donner l'inverse logique du booléen,
   - ajouter : "..." à la fin de votre chaine de caractère.
5. Afficher les résultats.
6. On souhaite regrouper ces résultats dans un cluster. Utiliser l'outil "Bundle" (assembler) pour rassembler les trois données. Créer un cluster contenant des indicateurs.

TD5 – Manipulation de fichiers

1 Objectif du TD

Acquérir des notions sur la manipulation des fichiers.

2 Séquençage d'un programme

Une application standard comprend souvent trois étapes :

1. L'initialisation (qui se fait une seule fois, au démarrage de l'application)
2. Le cœur du programme (qui s'exécute tant et aussi longtemps que l'application est en marche)
3. La fermeture (qui se fait une seule fois, juste avant l'arrêt complet de l'application)

Pour s'assurer d'un séquencement adéquat de ces étapes, trois solutions sont possibles :

- Une dépendance naturelle : un nœud ne peut s'exécuter que si toutes les entrées sont disponibles.
  Dans ce cas, c'est le flux de données qui cheminent via les fils de liaison qui détermine l'ordre d'exécution des éléments du diagramme.
- Une dépendance artificielle : même si un flux de données n'est pas nécessaire à une opération, il peut être souhaitable d'en faire un flux des données requises afin de s'assurer qu'elle ne s'exécute que lorsque l'exécution du nœud précédent est complétée.
- Une structure Séquence : dans une structure Séquence (déroulée ou empilée), un programme est divisé en sous-diagrammes (étapes) qui s'exécutent un par un, suivant l'ordre imposé par la Séquence.

3 Application à la lecture de fichiers

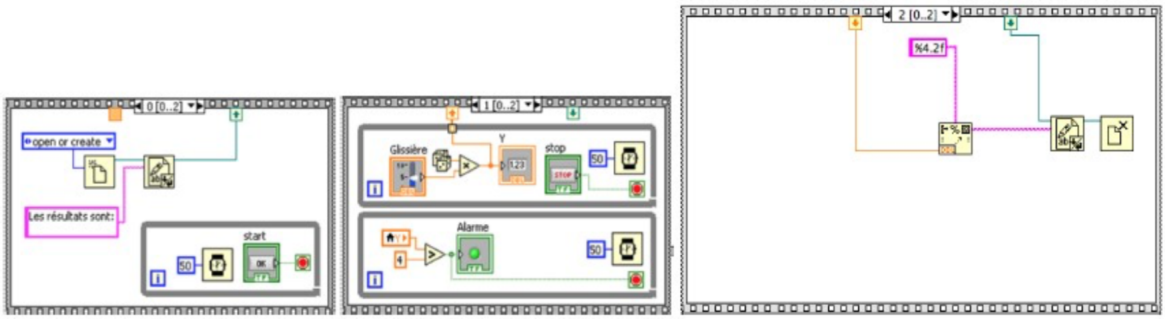
1. Créer une séquence déroulée de trois étapes.
2. Dans la première étape, ajouter le VI d'ouverture de fichier. "Open or Create"
3. Dans la commande de ce bloc, créer une constante et sélectionner "Open or Create".
4. Dans la seconde étape, on souhaite écrire dans le fichier une chaine de caractère donnée par l'utilisateur.
5. Dans la dernière étape de la séquence, on va fermer le fichier.
6. Écrire quelque chose dans la variable de texte et vérifier le bon comportement de ce VI.

4 Calculs parallèle et enregistrement de données

Lorsque deux ou plusieurs tâches peuvent être effectuées de façon indépendante (sans que l'une ait un impact sur l'autre ou que l'ordre de leur exécution ait un impact sur les résultats du programme), on parlera de tâches parallèles. Dans LabVIEW, les tâches parallèles sont souvent insérées dans des boucles While indépendantes (sans aucun fil de liaison entre elles).
On va créer un VI avec une structure séquence de trois étapes superposées :

- L'étape 1 doit créer un fichier et écrire l'entête suivante : "Les résultats sont :"
- L'étape 2 va simuler une mesure qui fluctue dans le temps et gérer un signal d'alarme.
- L'étape 3 va enregistrer les données.

1. Créer un VI avec une séquence empilée de trois étapes.
2. Les différentes étapes doivent ressembler aux images suivantes :



Le diagramme de ce VI comporte certains éléments qui vous sont inconnus :

- Carrés orange et vert avec une flèche : il représente une variable locale de séquence. Cette variable permet de faire passer des données d'une étape à l'autre de la séquence. Elle est ajoutée à l'aide d'un clic droit sur le rebord de la séquence.
- Rectangles avec une petite maison : ce sont des variables locales. Variable qui renferme les mêmes informations que la variable d'origine. Une variable locale est créée en cliquant, avec le bouton droit de la souris, sur l'indicateur d'origine. Par défaut, cette variable est en mode écriture ; avec le bouton droit de la souris, il est possible de changer ce mode en lecture.
- Fonction "Formater en chaîne" de l'étape 3.
- Fonction générateur aléatoire de l'étape 2.

3. Vérifier le fonctionnement du VI décrit par la figure. Le VI précédent a une condition d'arrêt étrange. Dans l'étape 2 si l'alarme n'a pas lieu ou que le bouton stop n'est pas pressé, le programme va rester coincé.
4. Faire en sorte que les deux boucles s'arrêtent si le bouton stop OU l'alarme est déclenchée.
5. Pourquoi n'a-t-on qu'un résultat dans le fichier ? Observer le déroulement du programme pour l'expliquer.
6. Modifier le VI pour qu'on puisse enregistrer au maximum 100 valeurs. (Arrêt du programme après 100 mesures ou arrêt précoce par le bouton ou l'alarme).

TD6 – Affichages et graphes

1 Objectif du TD

Acquérir les notions de manipulation des graphes et affichages.

2 Convention temporelle

LabVIEW comprend les types de graphes et de graphes déroulants suivants :

- **Graphes et graphes déroulants :** Affichent les données acquises à une vitesse constante.
- **Graphes XY :** Affichent les données acquises à une fréquence variable et les données pour les fonctions à valeurs multiples.
- **Graphes et graphes déroulants d'intensité :** Affichent des données 3D sur un tracé 2D en utilisant des couleurs pour afficher les valeurs de la troisième dimension.
- **Graphes numériques :** Affichent les données en tant qu'impulsions ou groupes de lignes numériques.
- **Graphes de signaux mixtes :** Affichent les types de données qu'acceptent les graphes, les graphes numériques et les graphes XY. Acceptent aussi les clusters qui contiennent des combinaisons de ces types de données.
- **Graphes 2D :** Affichent des données 2D sur un tracé 2D de face-avant.
- **Graphes 3D :** Affichent des données 3D sur un tracé 3D de face-avant.
- **Graphes 3D ActiveX :** Affichent des données 3D sur un tracé 3D dans un objet ActiveX de la face-avant.

Nous nous consacrerons ici que sur les affichages : graphXY, Waveform Chart et Waveform Graph.

3 Affichage des données contenues dans un fichier Excel

On va créer un fichier Excel (.csv) et afficher les données contenues dans celui-ci. Les données de la première colonne seront les abscisses tandis que celles de la seconde sont les ordonnées correspondantes.
La première colonne correspond au temps, la seconde à l'amplitude du phénomène mesuré.

1. Créer un fichier Excel avec des valeurs quelconques.
2. Enregistrer ce fichier au format csv, vérifier le contenu du fichier ainsi sauvegardé.
3. Charger le fichier et lire les données contenues dans celui-ci de telle sorte que les données soient contenues dans un tableau sous LabVIEW.
4. Tenter d'afficher les données avec les différents outils disponibles de graphe. Expliquer les affichages.
   Tenter de modifier les données d'entrée en les transposant, observer les résultats.
   On souhaite utiliser le graphe XY avec en X les données de la première colonne et en Y les valeurs de la seconde colonne. Néanmoins, l'affichage XY ne prend en entrée un cluster et pas une matrice/tableau.
   Il faut donc décomposer les lignes du tableau et les refusionner dans un cluster.
5. Découper le tableau en deux colonnes (ou lignes si vous travaillez sur la transposée).
6. Afficher les deux nouveaux tableaux.
7. Regrouper ces deux tableaux dans un cluster.
8. Afficher ce cluster dans le graphe XY.

4 Simulation de signal et affichage en continu des données

Dans cette partie, on va générer un signal simulé et l'observer sur différents affichages.

1. Créer un VI, insérer un module de simulation de signaux.
2. Entrer dans les propriétés du simulateur et choisir un signal sinusoïdal d'amplitude 1V, de fréquence 2 Hz avec 1000 échantillons par secondes et avec un bruit d'amplitude 0.1V.
3. Créer un afficheur numérique en sortie du générateur.
4. Afficher ce signal sur un graphe déroulant et un graphe classique.
5. Agrandir la case du simulateur et moduler l'amplitude du sinus par un signal carré de fréquence 0.5 Hz et d'amplitude 0.5 V et de valeur continue 0.5 V.
6. On souhaite afficher sur le même graphique le signal carré et le signal sinusoïdal modulé. Modifier le VI pour rendre cet affichage possible. Attention, le graphe déroulant doit prendre des doubles en entrée, il faut transtyper la sortie des simulateurs.

TD7 – Gestion des structures

1 Objectif du TD

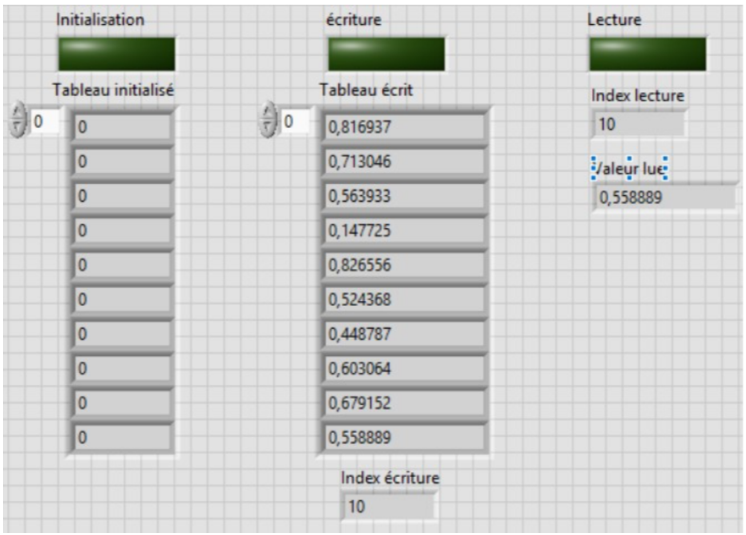
Séquencer un diagramme pour améliorer la lisibilité du programme.

2 Mise en place du programme

On veut remplir un tableau de 10 lignes avec des nombres aléatoires.
Á cet effet, utiliser la structure Séquence Déroulée pour séparer le programme en trois étapes explicites :
– Initialisation du tableau – Ecriture du tableau – Lecture du tableau

Créer une séquence déroulée à 3 étapes.

- **Séquence 1 :** Initialisation du tableau
  1. Sur la face avant (Front Panel), créer un tableau (nommé Tableau initialisé) de 10 lignes.
  2. Sur le bloc diagramme, initialiser toutes les lignes de ce tableau à 0. Ce tableau sera stocké dans une variable locale.
- **Séquence 2 :** Ecriture du tableau
  1. Sur la face avant (Front Panel), créer un tableau (nommé Tableau écrit) de 10 lignes et un indicateur Index écriture.
  2. Placer une boucle For en câblant N avec le nombre de lignes du tableau et placer un registre à décalage sur cette boucle.
  3. Placer la variable locale Tableau initialisé. Cette variable doit être lue.
  4. Initialiser ce registre par la variable locale Tableau initialisé.
  5. Réaliser l'écriture de nombres aléatoires dans ce tableau.
  6. Ecrire le résultat dans la variable locale Tableau écrit.
- **Séquence 3 :** Lecture du tableau
  1. Sur la face avant (Front Panel), créer un indicateur Index Lecture pour la ligne lue et Valeur lue pour la lecture.
  2. Créer une boucle For en câblant le nombre d'itérations et en insérant un temporisateur de 750 ms.
  3. Pour lire les valeurs dans le Tableau écrit, utiliser la fonction indexer un tableau.
  4. Vérifier le bon fonctionnement du programme.
- Ajouter des LEDs pour vérifier quelle séquence est en cours d'exécution (voir la figure ci-dessous).


Initialisation
écriture
Lecture
Tableau initialisé
Tableau écrit
Index lecture
Valeur lue
0
0
0
0
0
0
0
0
0
0
0
0,816937
0,713046
0,563933
0,147725
0,826556
0,524368
0,448787
0,603064
0,679152
0,558889
10
0,558889
Index écriture
10

TD8 – Traitement de données

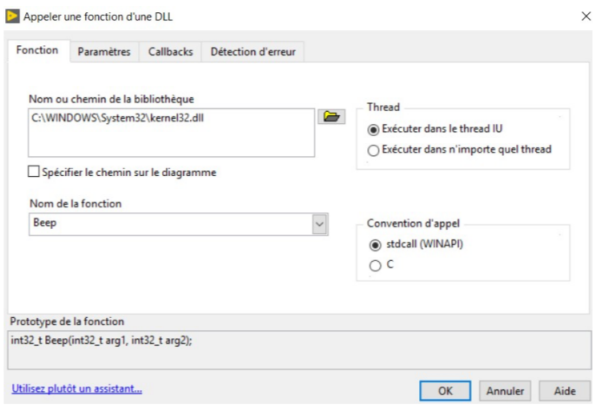
1 Objectif du TD

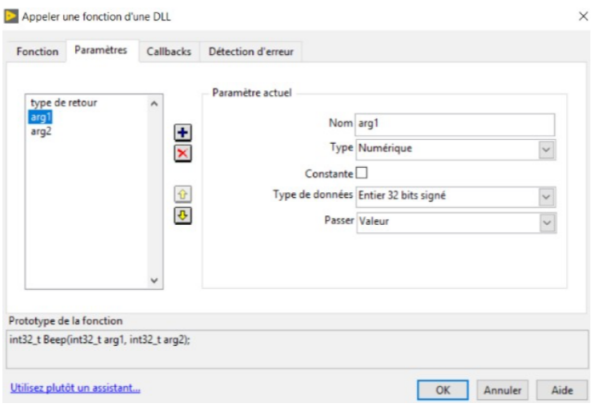
Acquérir des notions en traitement de données sous LabVIEW.

2 Mise en place du programme

On doit simuler des données acquises chaque demi-seconde à l'aide d'un capteur, pour cela on utilise un générateur de variable aléatoire qui varie entre 20 et 30.

1. Afficher ces données dans un tableau, graphe et indicateur.
2. Ecrire ces données dans un fichier de mesures. Choisir format Texte (LVM). Ne pas oublier de mentionner le nom et l'emplacement de ce fichier.
3. Ecrire ces données dans un fichier de mesures. Choisir format Microsoft Excel (.xlsx). Ne pas oublier de mentionner le nom et l'emplacement de ce fichier.
4. Calculer le minimum, la moyenne et le maximum des valeurs simulées.
5. Allumer une LED rouge et déclencher une alarme dans le cas où les données dépassent un certain seuil (exemple : seuil=28).
   Indice : Pour l'alarme, utiliser la fonction "Appeler une fonction d'une DLL". Connectivité ⟶ Biblio et exécutables ⟶ Appel fonction DDL. Voir les images ci-dessous.





6. Ouvrir les fichiers enregistrés (Questions 3 et 4) et visualiser-les pour constater les différences.

TD9 – Compteur et afficheur 7 segments

1 Objectif du TD

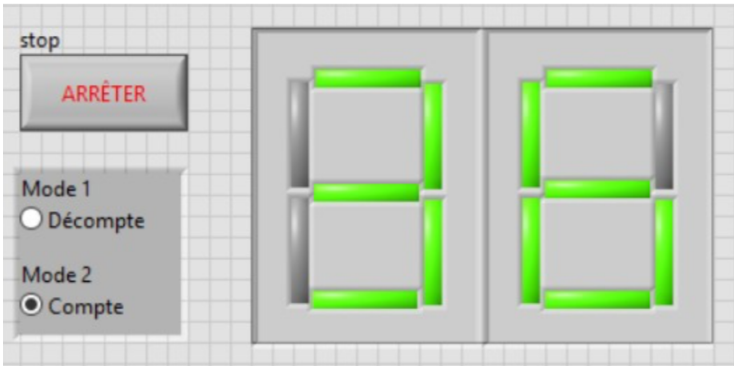
Réaliser un compteur – décompteur.

2 Mise en place du programme

- On va réaliser un afficheur 7 segments à 2 digits (2 chiffres).
- Utiliser les "LED carrées" pour réaliser cet afficheur.

Rappel : Chaque segment est identifié par des lettres (a ... g).
Chaque digit doit être inséré à l'intérieur d'un cluster.

1. Le compteur doit compter de 0 à 99 avec une fréquence de 1 Hz. Initialiser chaque digit de l'afficheur à l'aide d'un tableau de constantes logiques (True, False).
2. Coder votre programme pour réaliser ce compteur.
3. On veut que notre programme réalise aussi l'opération de décomptage (de 99 à 00). C'est à l'utilisateur de choisir quelle opération à réaliser.
   La face avant (Front panel) doit ressembler (peut être différente selon les affichages utilisés) à :



TD10 – Programmation avancée

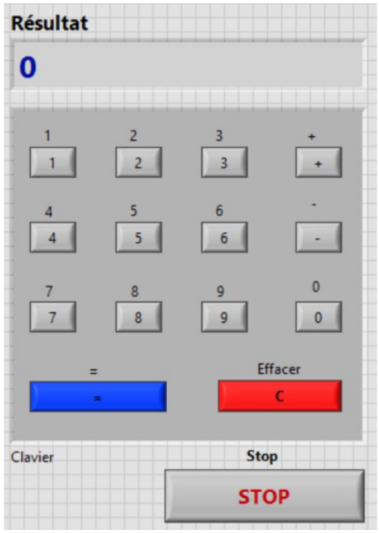
1 Codage d'une calculatrice

Nous allons réaliser en plusieurs étapes la programmation d'une calculatrice. Veillez à sauvegarder les différents VI avant chaque modification.

1. Créer un VI qui, à partir de deux valeurs numériques calculera la somme, la différence, le produit, la division et la moyenne de deux nombres.
2. Modifier ce VI pour l'inclure dans une boucle "While" de telle façon que l'on puisse changer les entrées sans avoir à relancer le programme.
3. Modifier le VI pour y ajouter un bouton d'arrêt.
4. Reprendre le VI précédent. Cette fois-ci, l'utilisateur pourra choisir le type de calcul qu'il souhaite utiliser à l'aide d'un menu déroulant (il vous faudra utiliser une structure "Case").

Á présent que vous savez utiliser les structures "Case", on va essayer de coder une calculatrice où toutes les touches sont des boutons (à la façon de la calculatrice Windows).

1. Créer un VI, et réaliser la face avant du VI comprenant les boutons des chiffres de 0 à 9 et les opérations de base : + et –.
2. Renommer les boutons et leur textes booléens de façon à ce que cela corresponde au rôle du bouton (exemple le bouton + avec le texte +).
3. Regrouper les différents boutons dans un "Cluster". Pour cela, sur la face avant créer un "Cluster" et glisser à l'intérieur les différents boutons.
   Á ce stade vous devriez avoir quelque chose comme :

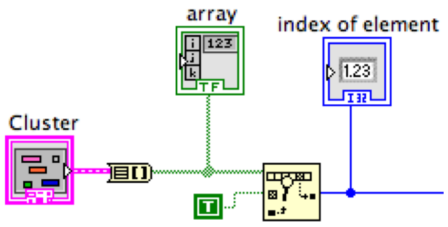


4. Il faut ordonner ces boutons, clic droit sur le cluster, choisissez ordonner et ordonner les boutons de façon intelligente.
5. Convertir la sortie du cluster en tableau, créer alors un afficheur tableau. Agrandissez le tableau horizontalement, la valeur du curseur indique la première case représentée, laissez là à zéro.

6. Lancer le programme et tester selon le bouton appuyé l'effet sur le tableau.
   Á ce stade on peut savoir quel bouton a été appuyé grâce à la case du tableau mise à 1 ou "vrai".
   Pour trouver cette case, nous allons utiliser les outils de tableau et principalement celui "recherche dans un tableau 1D".

7. Ajouter la brique de recherche dans le tableau 1D de la valeur booléenne "vraie" et créer un indicateur pour vérifier la sortie.

8. La réaction des boutons est trop rapide, ajouter une temporisation.
   Au final la lecture du cluster doit ressembler (peut différer selon les affichages utilisés) à :



   Il va falloir créer une structure "Case" adaptée à cette sortie. Pour faire cela de façon simple nous allons réaliser une version de la calculette non optimale (certains cas ne seront pas gérés).

9. Faire en sorte que lorsqu'on appuie à la suite sur des chiffres de 0 à 9, la valeur numérique obtenue soit bonne (exemple : si on appuie sur "1" puis "7" puis "0", un afficheur indique 1 puis 17 puis 170).
   Chaque chiffre doit ainsi passer à une puissance de 10 supérieure. Pour cela utiliser un registre à décalage et des opérateurs mathématiques simples (+ et x). Pour définir un cas multiple de 0 à 9, on peut écrire 0 .. 9.
   Vérifier le bon comportement de cette étape.
   Á ce stade la valeur qui est en train d'être entrée est gérée. Il faut dès lors s'occuper de la variable qui va contenir le résultat et les appuis sur les boutons d'opération +, –, = et C (tout effacer).

10. Ajouter un registre à décalage pour gérer la variable de résultat.

11. Coder le cas "C" (effacement) qui remet la valeur courante et le résultat à 0. Tester le bon fonctionnement de ce bouton.
    Lorsque l'on va appuyer sur + ou –, l'opération ne devra se faire qu'après la fin de l'entrée du nombre à ajouter (ou à soustraire). Par exemple si l'on tape :

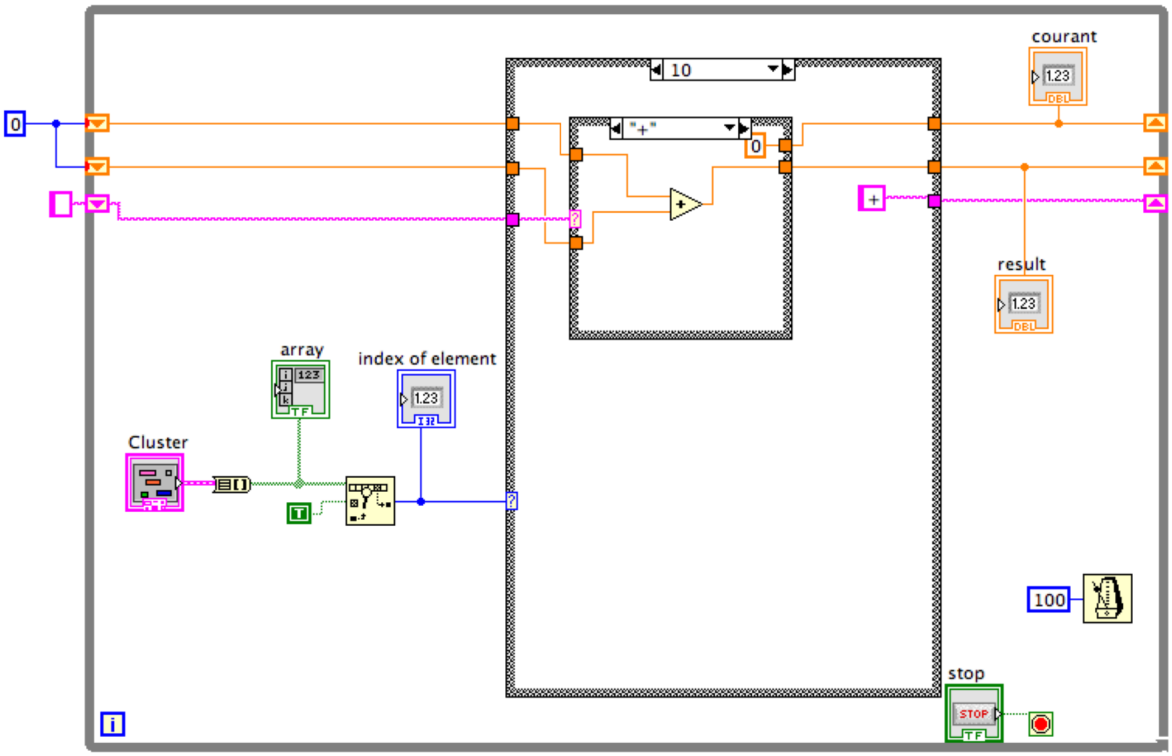
    - "2 +", il va falloir attendre le prochain chiffre/nombre.
    - "2 + 3", on ne sait pas si après le 3 il va y avoir un autre chiffre...
    - "2 + 32 =" à ce stade seulement on sait que l'opération 2+32 peut être faite !
      Il faut donc mémoriser l'opération jusqu'à ce que l'on rencontre une nouvelle opération pour pouvoir l'appliquer.

12. Il va falloir mémoriser l'opération, pour cela il faut à nouveau un registre à décalage qui stockera le "signe de l'opération" + / – / =.

13. A chaque appui sur un bouton de calcul, il va donc falloir exécuter l'opération mémorisée si elle existe.
    Créer donc une structure "Case" à l'intérieur de la structure précédente pour gérer les différents cas d'opération.

14. Lister sur papier les différents cas possibles et les différents calculs à réaliser.

15. Coder ces différents cas. N'oubliez pas de gérer les cas par défaut !
    Pour vous aider, voici la vue générale d'un des cas (cas 10, addition (appuie sur +), cas + (l'opération précédente est un +)) :



16. Ajouter les opérations "multiplier", "diviser" et "racine carrée".
17. Chercher les cas non gérés par cette calculatrice. Proposer des solutions et tester les.

2 Réponse indicielle, diagrammes de Bode, Black et Nyquist

Dans cette partie on va simuler des systèmes linéaires de 1er et de 2ème ordre et afficher leurs diagrammes de Bode, lieux de Black, Nyquist et leur réponse indicielle.

Remarque 2. Tous les modules sont disponibles sur la "face avant" (front panel) dans Contrôle et Simulation $\longrightarrow$ Control Design.

2.1 Système de 1er ordre

1. Créer un nouveau VI, insérer dans la face avant (front panel) un module de création de modèle de 1er ordre.
2. Insérer dans la face avant (front panel) un module pour tracer le diagramme de Bode de ce système et câbler les deux modules entre eux.
3. Modifier les paramètres de votre système (Gain, Constante de temps et retard) et observer les changements.
4. Refaire la même chose pour la réponse indicielle et les lieux de Nyquist et Black (Nichols).

2.2 Système de 2ème ordre

1. Créer un nouveau VI, insérer dans la face avant (front panel) un module de création de modèle à partir de sa fonction de transfert. Remplir le numérateur et le dénominateur de votre fonction de transfert pour avoir un système de 2ème ordre.
2. Insérer dans la face avant (front panel) un module pour tracer le diagramme de Bode de ce système et câbler les deux modules entre eux.
3. Modifier la fonction de transfert de votre système (amortissement, pulsation, ...) et observer les changements.