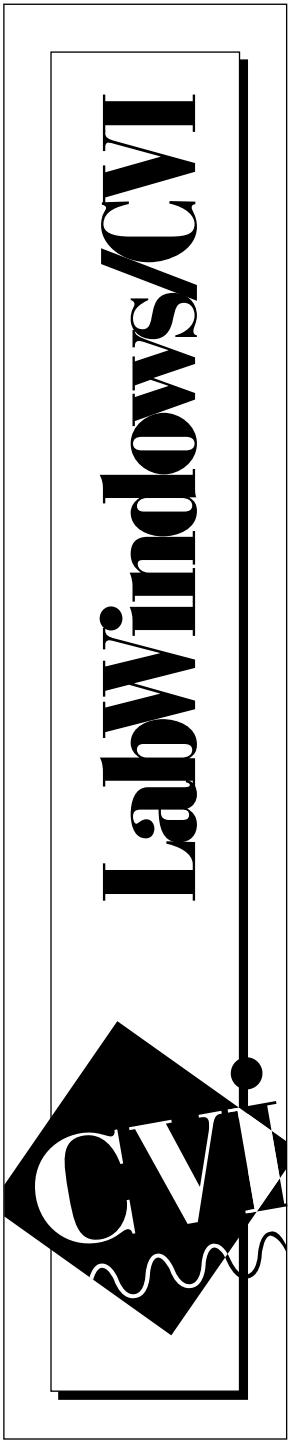


Guide d'évaluation LabWindows®/CVI

Édition de septembre 1997
Référence 350349A-01

© Copyright 1997 National Instruments Corporation. Tous droits réservés.

Copyright

Conformément à la loi sur les droits d'auteurs, ce document ne peut être ni reproduit ni transmis, sous aucune forme que ce soit, informatique ou mécanique, notamment par photocopie, enregistrement, stockage dans un système d'archivage de documentation, ni traduit intégralement ou en partie, sans l'autorisation écrite de National Instruments Corporation.

Marques déposées

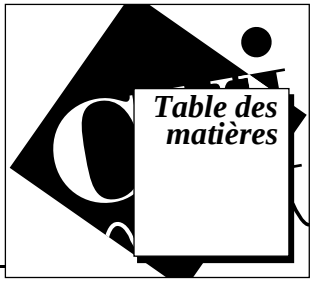
CVI™ et `natinst.com`™ sont des marques déposées par National Instruments Corporation. Les noms de produits et de sociétés cités sont des marques déposées par leurs propriétaires respectifs.

LIMITATIONS DE GARANTIE

CE LOGICIEL DE DÉMONSTRATION AINSI QUE LA DOCUMENTATION QUI L'ACCOMPAGNE (LE MODE D'EMPLOI COMPRIS) SONT FOURNIS EN L'ÉTAT, SANS GARANTIE D'AUCUNE SORTE. EN OUTRE, NATIONAL INSTRUMENTS CORPORATION NE CERTIFIE, NE GARANTIT OU NE FAIT AUCUNE INTERPRÉTATION QUANT À L'USAGE OU LES RÉSULTATS DE L'UTILISATION DU LOGICIEL DE DÉMONSTRATION OU DE LA DOCUMENTATION CORRESPONDANTE EN TERMES D'EXACTITUDE, DE PRÉCISION, DE FIABILITÉ OU AUTRES. C'EST À VOUS QUE LE RISQUE INCOMBE. SI LE LOGICIEL DE DÉMONSTRATION OU LA DOCUMENTATION CORRESPONDANTE SONT DÉFECTUEUX, C'EST À VOUS SEUL, ET NON À NATIONAL INSTRUMENTS, SES PARTENAIRES, DISTRIBUTEURS, AGENTS OU COLLABORATEURS, QU'INCOMBERONT LES FRAIS DE RÉPARATION, SERVICE OU CORRECTION.

NATIONAL INSTRUMENTS CORPORATION NIE TOUTE GARANTIE DE TRANSACTIONS COMMERCIALES, D'APTITUDE À UNE FIN PARTICULIÈRE. AUCUNE INFORMATION OU CONSEIL, ORAL OU ÉCRIT, FOURNI PAR NATIONAL INSTRUMENTS CORPORATION, SES PARTENAIRES, DISTRIBUTEURS, AGENTS OU COLLABORATEURS NE SAURAIT FAIRE OFFICE DE GARANTIE. L'ENSEMBLE DE VOS DROITS PEUT VARIER D'UNE JURIDICTION À L'AUTRE.

NI NATIONAL INSTRUMENTS NI QUICONQUE IMPLIQUÉ DANS LA CRÉATION, LA PRODUCTION OU LA LIVRAISON DE CE PRODUIT NE SAURAIT ÊTRE TENU RESPONSABLE D'AUCUN DOMMAGE, DIRECT, INDIRECT, DÉRIVÉ OU ACCIDENTEL, DONT LES PERTES DE PROFITS, L'INTERRUPTION D'EXPLOITATION, LES PERTES D'INFORMATIONS, LE TOUT RÉSULTANT DE L'UTILISATION OU DE L'INCAPACITÉ À UTILISER UN TEL PRODUIT MÊME SI NATIONAL INSTRUMENTS CORPORATION A ÉTÉ INFORMÉ DE TELS RISQUES. ÉTANT DONNÉ QUE CERTAINS ÉTATS NE PERMETTENT PAS LA LIMITATION OU L'EXCLUSION DE LA RESPONSABILITÉ POUR LES DOMMAGES ACCIDENTELS OU DÉRIVÉS, LA LIMITATION PRÉCÉDEMMENT DÉCRITE PEUT NE PAS S'APPLIQUER À VOTRE CAS.



Chapitre 1
Le C/C++ conçu pour les scientifiques et les ingénieurs

Chapitre 2
Concevoir une interface utilisateur

Chapitre 3
Générer le code du programme avec CodeBuilder

Chapitre 4
Compléter le programme avec un driver d'instrument

Chapitre 5
Ajouter un moniteur de températures au programme

Chapitre 6
LabWindows/CVI : la solution idéale

Chapitre 7
L’engagement de National Instruments

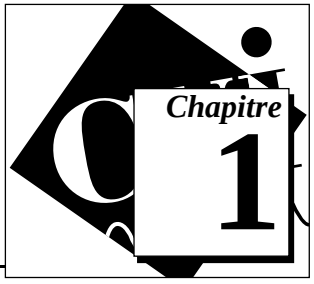
Conventions

Souvenez-vous des conventions d’écriture suivantes en parcourant ce document :

<> Les parenthèses angulaires entourent le nom d’une touche au clavier. Par exemple : <Option>. Elles peuvent également entourer les noms des constantes dans le langage HiQ-Script.

» Le symbol » vous indique le chemin à suivre dans les options de menu et celles des boîtes de dialogue pour procéder à une opération spécifique. Par exemple : la séquence **File»Save As** vous invite à ouvrir le menu **File** puis à sélectionner **Save As**.

Le C/C++ conçu pour les scientifiques et les ingénieurs



Les industries de l'automatisation industrielle et du test et mesure doivent désormais compter sur un nouveau concept : l'instrumentation virtuelle. Bon nombre de scientifiques et d'ingénieurs voient désormais en l'instrumentation virtuelle un moyen de se libérer des limitations propres aux instruments traditionnels. Aujourd'hui plus que jamais, ils reconnaissent tous l'intérêt d'intégrer des éléments matériels et logiciels dans les PC et les stations de travail pour personnaliser des solutions d'instrumentation.

Pour les programmeurs traditionnels qui développent l'instrumentation virtuelle, LabWindows/CVI est devenu l'environnement de développement logiciel le plus répandu. Il permet à tous ceux qui maîtrisent les principes essentiels d'un langage de programmation textuel de construire seuls des instruments virtuels. LabWindows/CVI associe en effet la puissance du langage C ANSI à la souplesse de Windows. Il propose des outils faciles d'utilisation conçus pour la construction de systèmes d'instrumentation virtuelle.

Même les programmeurs inexpérimentés peuvent se servir des utilitaires inédits de génération de code de LabWindows/CVI pour exploiter la puissance du langage C ANSI et limiter le temps de développement en vue de :

- Créer et contrôler des interfaces utilisateur graphiques (GUI) sous Windows.
- Contrôler des instruments et du matériel d'acquisition de données à partir d'un PC.
- Développer et mettre au point des programmes C ANSI pour l'instrumentation.

En outre, LabWindows/CVI pour Windows 95/NT est désormais compatible avec les environnements de développement standards C/C++ 32 bits de Microsoft, Borland, Symantec et WATCOM. Que vous soyez programmeur C professionnel ou développeur occasionnel, vous verrez que les outils contenus dans LabWindows/CVI vous aideront à concevoir et à construire des instruments virtuels vite et facilement.

A propos du logiciel d'évaluation

Configuration système requise

- Processeur 386/25 MHz (486/33 MHz recommandé)
- Co-processeur
- 8 Mo de mémoire vive
- 30 Mo de disque dur
- Adaptateur vidéo VGA ou Super VGA
- Windows 95/NT ou Windows 3.1
- Souris

Fonctionnalités du logiciel d'évaluation LabWindows/CVI

Cette version d'évaluation fonctionne comme le véritable logiciel LabWindows/CVI avec toutefois les restrictions suivantes :

- Vous ne pouvez pas construire de fichiers d'exécutables (`.exe` ou `.dll`)
- Vos programmes ne peuvent fonctionner que dix minutes seulement
- Au-delà de 30 jours, vous ne serez plus en mesure de lancer l'application
- Vous ne pouvez pas imprimer de fichiers source ou de ressource (`.uir`)

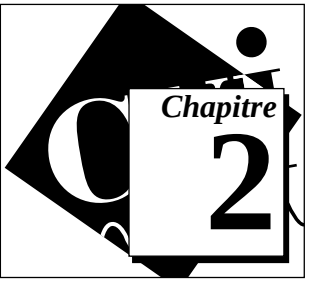
Pour éviter toute perte de travail pendant la période d'évaluation, vous pouvez enregistrer tous les programmes construits, pour ensuite les ouvrir avec une version entièrement fonctionnelle de LabWindows/CVI.

Installer le logiciel

Ce guide d'évaluation est conçu pour fonctionner avec la version 4.0.1 du logiciel d'évaluation LabWindows/CVI. Vous pouvez obtenir ce logiciel dans le CD-ROM Software Showcase ou sur l'InstrumentationWeb à l'adresse suivante : `www.natinst.com/CVI`.

Pour installer LabWindows/CVI à partir du Software Showcase, allez dans la section LabWindows/CVI dans *Virtual Instrumentation Tools* et cliquez sur l'icône de démonstration **demo**.

Concevoir une interface utilisateur



Quoique inspiré du C ANSI, LabWindows/CVI est un outil visuel qui sert à développer des applications d'instrumentation. L'interface utilisateur conçoit votre programme du début jusqu'à la fin. Visuel et intuitif, LabWindows/CVI vous permet de construire un instrument virtuel en un minimum de temps.

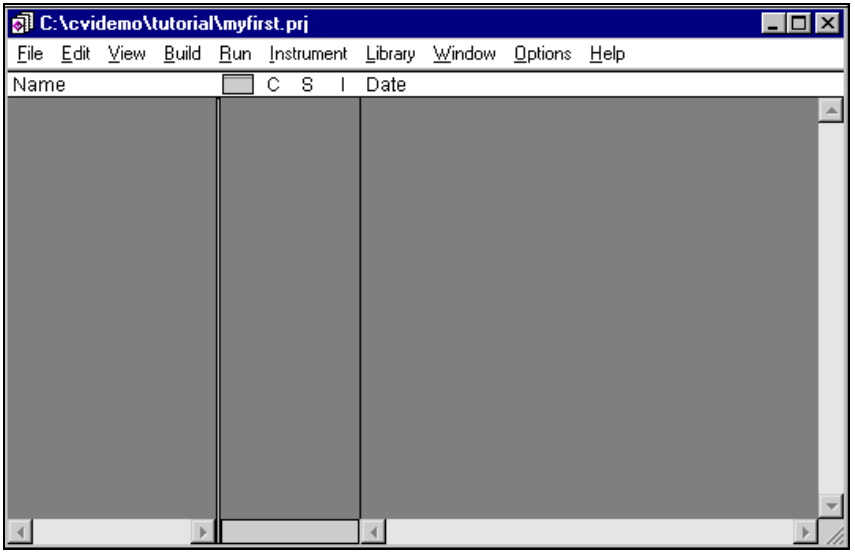
Ce chapitre vous explique comment utiliser l'Éditeur d'interface utilisateur pour construire l'interface utilisateur graphique correspondant à votre instrument. Le chapitre suivant vous explique comment générer du code source directement à partir de l'interface utilisateur créée dans ce chapitre.

Un exemple de projet

L'interface utilisateur graphique (GUI) est l'élément essentiel de n'importe quel projet LabWindows/CVI. Dans un premier temps, vous allez ouvrir un projet et créer une GUI à l'aide de l'Éditeur d'interface utilisateur. Une fois terminé, votre projet va procéder à des opérations simulées d'acquisition de données et de contrôle d'instruments.

1. Sélectionnez **Open Project** dans le menu **File**. Dans la boîte de dialogue **Open File**, sélectionnez `myfirst.prj` dans le répertoire `cvidemo\tutorial`. Puis cliquez sur **Load**.
2. Comme vous pouvez le voir dans le graphique ci-après, le fichier de projet `myfirst.prj` ne devrait contenir absolument aucun fichier pour le moment. Si cette liste de projets est effectivement vide, vous pouvez commencer. Si ce n'est pas le cas, utilisez

l'option **Remove File** dans le menu **Edit** pour effacer tous les fichiers du projet.



L'Éditeur d'interface utilisateur

L'Éditeur d'interface utilisateur est un environnement interactif de type glissé-déposé qui sert à dessiner les interfaces utilisateur. Vous avez le choix entre un grand nombre de contrôles différents (boutons, interrupteurs à bascule, glissières, graphes, LED, etc.) dans le menu **Create** que vous placez ensuite dans vos panneaux. Vous pouvez ensuite vous servir de toute une série de boîtes de dialogue pour régler les attributs des contrôles, comme des étiquettes, des couleurs ou des connexions de raccourcis clavier. Dans l'exemple suivant, vous construisez une interface utilisateur qui acquiert et affiche une courbe.

Étape 1 : ouvrez un nouveau fichier .uir

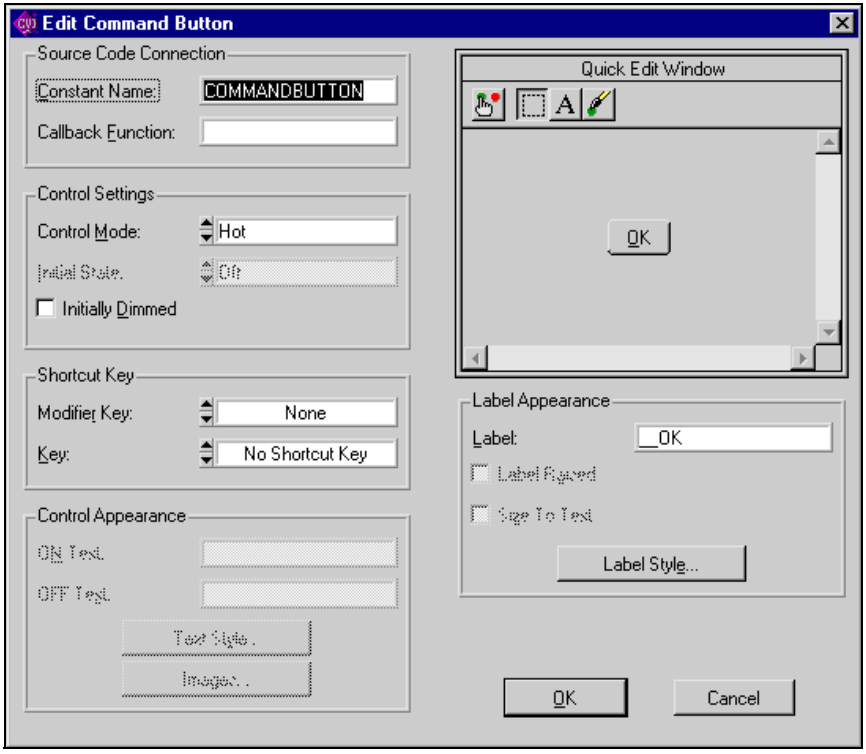
1. Sélectionnez **New»User Interface (.uir)…** dans le menu **File**. L'Éditeur d'interface utilisateur ouvre un fichier sans titre.
2. Sélectionnez **Panel** dans le menu **Create** pour créer le panneau de votre interface utilisateur. Ensuite, vous pouvez y ajouter des contrôles et des indicateurs.

Étape 2 : ajoutez un bouton de commande

1. Sélectionnez **Command Button** dans le menu **Create** et choisissez **Rounded Command Button** dans la palette qui s'affiche. Un

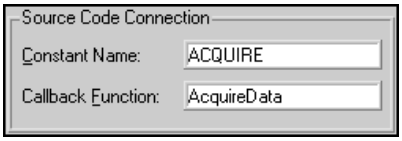
bouton étiqueté **OK** apparaît alors sur le panneau. Servez-vous de votre souris pour y positionner le bouton en haut à gauche.

2. Pour modifier les attributs du bouton, double-cliquez dessus. Vous verrez apparaître une boîte de dialogue intitulée Edit Command Button comme dans l'illustration suivante.



3. Attribuez un nom de constante au bouton. Tapez `ACQUIRE` dans le champ Constant Name à l'intérieur de la section Source Code Connection de la boîte de dialogue.
4. Toujours dans cette section, tapez `AcquireData` dans le champ Callback Function pour attribuer un nom à la fonction qui sera appelée dès que l'utilisateur cliquera sur le bouton.

5. La section Source Code Connection de la boîte de dialogue doit ressembler à ceci :



6. Changez l'étiquette du bouton de commande en effaçant `_OK` dans **Label** et en le remplaçant par `Acquire`.
7. Cliquez sur **OK** en bas de la boîte de dialogue Edit Command Button pour enregistrer vos modifications, puis fermez la boîte de dialogue.

Étape 3 : créez un deuxième bouton de commande

1. Renouvelez l'étape 2 pour créer un deuxième bouton de commande rond et placez-le en bas à droite du panneau. Double-cliquez dessus pour ouvrir la boîte de dialogue Edit Command Button.
2. Dans la boîte de dialogue Edit Command Button, entrez les paramètres suivants :

   Constant Name: `QUIT`

   Callback Function: `Shutdown`

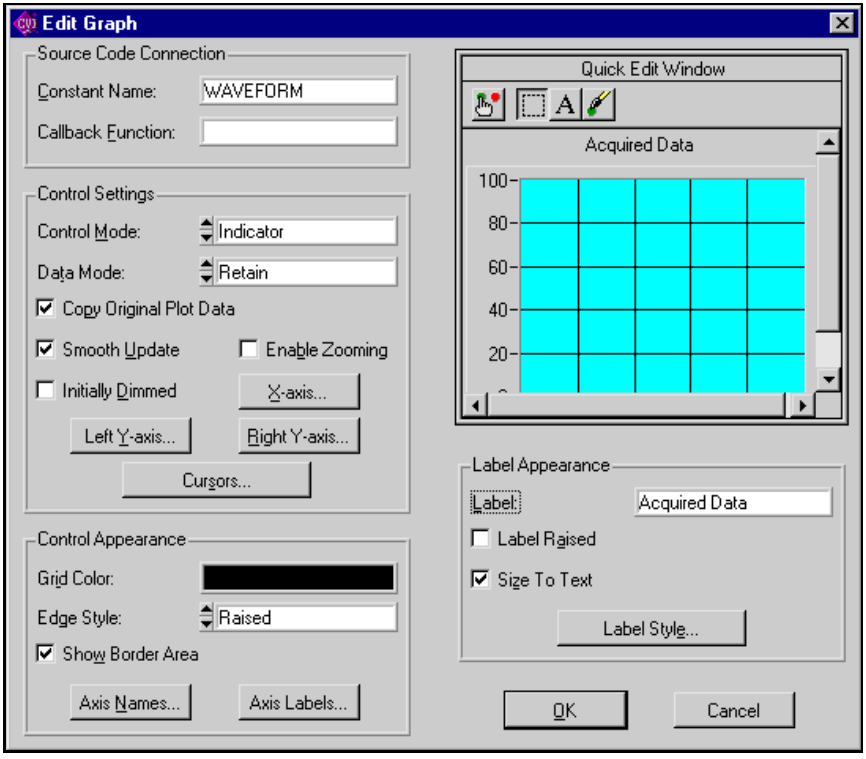
   Label: `Quit`
3. Cliquez sur le bouton **OK** en bas de la boîte de dialogue **Edit Command Button** pour enregistrer vos modifications, puis fermez la boîte de dialogue.

Étape 4 : ajoutez une commande de graphe dans l'interface utilisateur

1. Sélectionnez **Create»Graph** puis cliquez sur la commande de graphe dans la palette qui s'affiche. Une commande de graphe nommée **Untitled Control** devrait alors apparaître sur votre interface utilisateur.
2. Positionnez et dimensionnez le graphe à votre guise avec la souris.
3. Double-cliquez sur la commande de graphe pour afficher la boîte de dialogue Edit Graph.
4. Tapez `WAVEFORM` en lettres majuscules dans le champ Constant Name à l'intérieur de la section Source Code Connection de la boîte de dialogue.

☞ **Remarque** ***Étant donné que ce graphe ne déclenche pas d'action à partir de l'interface utilisateur, inutile d'attribuer une fonction "Callback" à cette commande. Les fonctions "Callback" ne s'imposent que si le fonctionnement de la commande déclenche une opération ou sert d'entrée.***

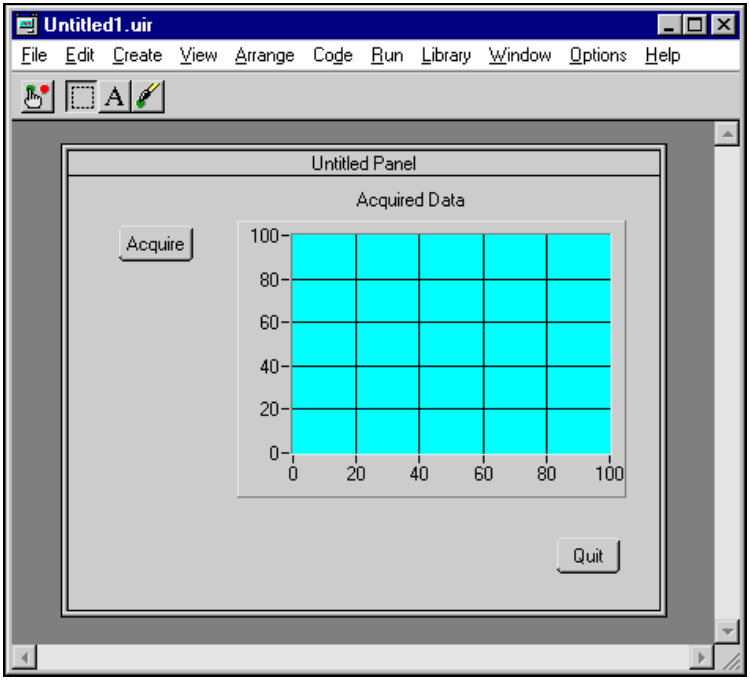
5. Tapez `Acquired Data` dans le cartouche Label à l'intérieur de la section Label Appearance de la boîte de dialogue. Cette boîte de dialogue devrait d'ailleurs ressembler à ce qui suit.



6. Cliquez sur le bouton **OK** en bas de la boîte de dialogue Edit Graph pour la fermer.

Étape 5 : enregistrez le fichier .uir

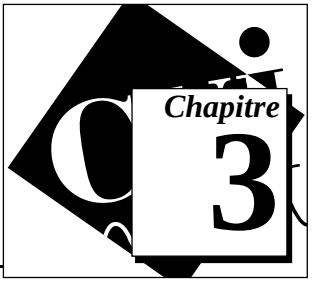
1. Une fois terminée, votre interface utilisateur devrait ressembler à l'illustration suivante.



2. Sélectionnez **File»Save As** pour enregistrer le fichier `.uir` avec les nouvelles commandes ajoutées.
3. Tapez `myfirst` dans le contrôle d'entrée File Name dans la boîte de dialogue **Save File As**. Cliquez sur **Save** pour enregistrer le fichier.

☞ **Remarque** ***Lorsque vous enregistrez un fichier*** `.uir`***, LabWindows/CVI crée automatiquement un fichier d'en-tête (***`.h`***) qui porte le même nom que l'interface utilisateur. Ce fichier d'en-tête est ensuite stocké dans le même répertoire que le fichier*** `.uir`***.***

Générer le code du programme avec CodeBuilder



Dans le chapitre précédent, vous avez conçu une simple interface utilisateur graphique. Dans ce chapitre, vous allez utiliser *CodeBuilder* pour construire automatiquement le code du programme. CodeBuilder crée un code C ANSI pour supporter les objets de votre interface utilisateur. Vous pouvez compiler et lancer votre programme sur le champ. Vous aurez ensuite à ajouter des fonctions au programme généré par CodeBuilder de telle sorte qu'il acquière et affiche une courbe.

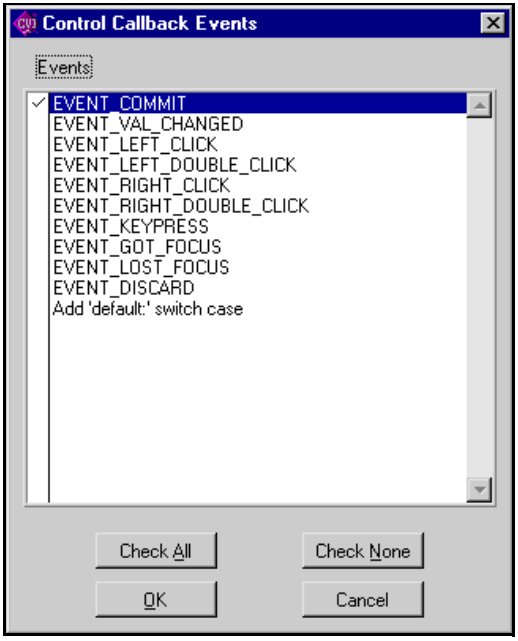
Vous allez suivre toutes les étapes de ce chapitre en utilisant le fichier de l'interface utilisateur, `myfirst.uir`, que vous avez construit au chapitre 2, *Concevoir une interface utilisateur*.

Génération du code automatique avec CodeBuilder

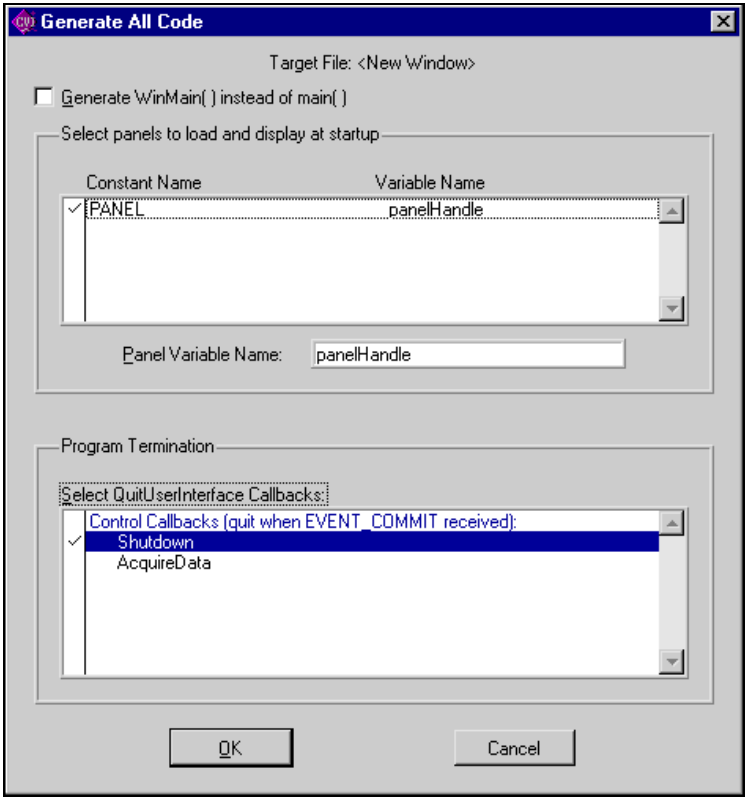
Pour l'instant, vous avez créé une interface utilisateur avec les objets de l'interface utilisateur. Maintenant, il faut que vous vous serviez de CodeBuilder pour créer le code source qui gérera votre interface utilisateur.

1. Précisez le type d'événements auxquels votre programme va réagir. Ouvrez `myfirst.uir` si ce n'est pas déjà fait. Sélectionnez **Code»Preferences»Default Control Events**.

2. Dans la boîte de dialogue Control Callback Events, sélectionnez le type d'événement qu'une commande peut recevoir. Dans ce projet, les commandes doivent réagir à un type particulier d'événement : un événement de type Commit (c'est-à-dire lorsque vous cliquez sur le bouton gauche ou sur <Enter>). Vérifiez que seul `EVENT_COMMIT` est sélectionné, puis cliquez sur **OK**.



3. Sélectionnez **Code»Generate»All Code** pour afficher la boîte de dialogue suivante.



4. En général, vous devez choisir quels panneaux vous souhaitez afficher au démarrage du programme. Vu qu'il n'existe qu'un seul panneau dans cet exemple, il vous suffit de vérifier que Panel Variable Name est bien `panelHandle`.

☞ **Remarque** ***Le nom de variable de votre interface utilisateur doit être*** `panelHandle` ***pour correspondre aux paramètres que vous allez déterminer ultérieurement dans cet exercice.***

5. En bas de la boîte de dialogue, vous verrez que sont répertoriées les fonctions "Callback" de votre fichier `.uir`. Il faut que vous y sélectionniez la fonction `Shutdown` de telle sorte que lorsque vous cliquerez sur le bouton **Quit** dans votre interface utilisateur, le programme s'arrêtera (LabWindows/CVI va insérer la fonction `QuitUserInterface` dans la fonction `Shutdown`).

6. Cliquez sur **OK**. CodeBuilder construit le code source de votre programme. Une nouvelle fenêtre d'édition de Source apparaît avec le nouveau code source.
7. Sélectionnez **File»Save** dans la fenêtre d'édition de Source, puis nommez le fichier source `myfirst.c`.

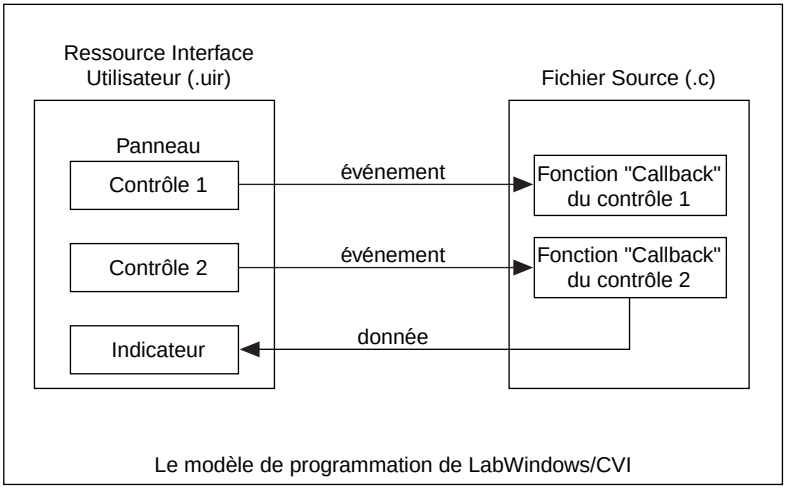
Afficher le code source

Le code suivant devrait s'afficher dans la fenêtre d'édition de Source, comme ci-dessous. Il devrait contenir les trois fonctions principales suivantes : la fonction `main`, la fonction `AcquireData` et la fonction `Shutdown`. Ce sont l'Éditeur d'interface utilisateur et CodeBuilder qui vous ont servi à créer ces fonctions.

```
#include <cvirte.h>/* needed if linking executable in external compiler; harmless oth-
erwise */
#include <userint.h>
#include "myfirst.h"

static int panelHandle;

int main (int argc, char *argv[])
{
if (InitCVIRTE (0, argv, 0) == 0)/* needed if linking executable in external compiler;
harmless otherwise */
return (-1);/* out of memory */
if ((panelHandle = LoadPanel (0, "myfirst.uir", PANEL)) < 0)
return -1;
DisplayPanel (panelHandle);
RunUserInterface ();
return 0;
}

int CVICALLBACK AcquireData (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
switch (event) {
case EVENT_COMMIT:

break;
}
return 0;
}

int CVICALLBACK Shutdown (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
switch (event) {
```

```
case EVENT_COMMIT:
QuitUserInterface (0);
break;

}
return 0;
}
```

Chacune des deux fonctions `CVICALLBACK` figurant dans le code source précédent correspond à un contrôle de votre interface utilisateur. L'illustration suivante présente la structure du modèle de programmation LabWindows/CVI.



Le modèle de programmation de LabWindows/CVI

Les fonctions présentes dans chacune des trois fonctions de ce projet (`main`, `AcquireData` et `Shutdown`) méritent une attention particulière.

La fonction main

La fonction `main` est le premier élément dont vous avez besoin pour construire vos applications. La fonction `main` de notre projet se compose des lignes de code suivantes :

```
int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)   /* Needed if linking in external compiler;
harmless otherwise */
        return -1;  /* out of memory */
    if ((panelHandle = LoadPanel (0, "myfirst.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    return 0;
}
```

Les fonctions contenues dans la fonction `main` doivent préparer votre interface utilisateur à fonctionner de la façon suivante :

- La fonction `LoadPanel` charge le panneau du fichier `.uir` en mémoire.
- La fonction `DisplayPanel` affiche le panneau à l'écran.
- La fonction `RunUserInterface` fait en sorte que LabWindows/CVI envoie les événements de l'interface utilisateur au programme C que vous êtes en train de mettre au point.

La fonction AcquireData

La fonction `AcquireData` présentée dans ce chapitre s'exécute automatiquement à chaque fois que vous cliquez sur le bouton **Acquire**. Pour le moment, elle ne contient aucune fonction à exécuter. Mais dans le prochain chapitre, votre mission consistera à y ajouter des fonctions d'acquisition de données.

```
int CVICALLBACK AcquireData (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
switch (event) {
case EVENT_COMMIT:

break;
}
return 0;
}
```

La fonction Shutdown

La fonction `Shutdown` présentée dans ce chapitre s'exécute automatiquement chaque fois que vous cliquez sur le bouton **Quit** dans votre interface utilisateur. Cette fonction empêche l'interface utilisateur d'envoyer des informations aux fonctions "Callback", et interrompt l'exécution du programme.

```
int CVICALLBACK Shutdown (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
switch (event) {
case EVENT_COMMIT:
QuitUserInterface (0);
break;
}
return 0;
}
```

Construire le fichier projet LabWindows/CVI

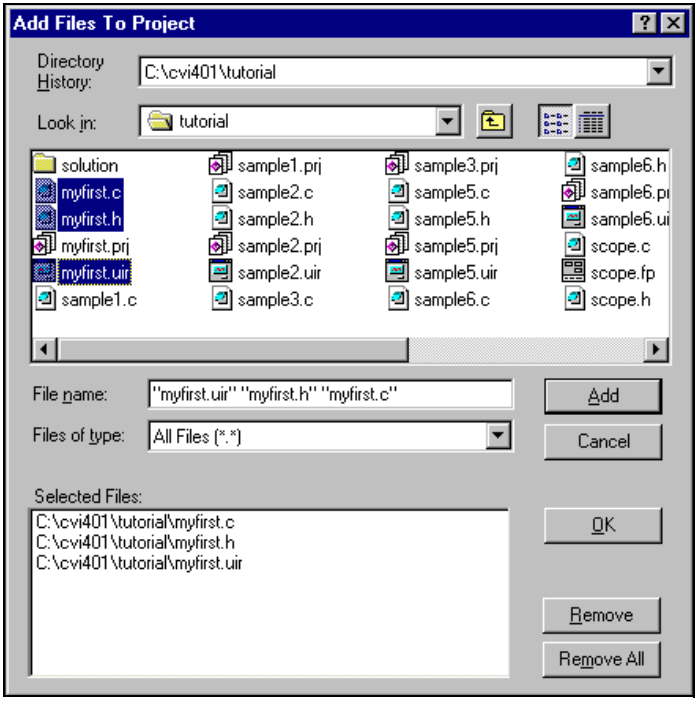
Maintenant que vous avez créé les fichiers source pour votre programme, vous pouvez construire un projet LabWindows/CVI. Dans le cas présent, vous devez stocker les trois noms de fichiers suivants dans le projet de fichier.

- `myfirst.uir` : il s'agit du fichier d'interface utilisateur que vous avez construit dans le chapitre précédent.
- `myfirst.h` : il s'agit du fichier inclus qui a été généré automatiquement au moment où vous avez enregistré le fichier `.uir` au chapitre précédent. Ce fichier inclus déclare tous les noms de constantes et les fonctions "Callback" que vous avez attribués dans l'Éditeur d'interface utilisateur (étapes 2, 3 et 4 au chapitre 2, *Concevoir une interface utilisateur*).
- `myfirst.c` : il s'agit du fichier source créé dans ce chapitre.

Pour construire un projet de fichier, suivez les étapes suivantes :

1. Fermez toutes les fenêtres LabWindows/CVI sauf la fenêtre Projet.

2. Dans la barre de menus, sélectionnez **Edit»Add Files to Project»All Files (*.*)**. Vous verrez apparaître une boîte de dialogue contenant tous les fichiers du répertoire actif.

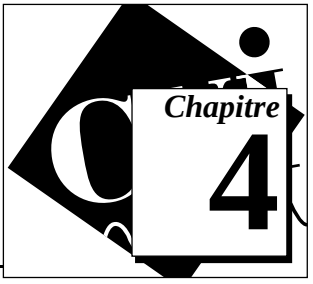


3. Tout en appuyant sur la touche <Ctrl>, cliquez sur `myfirst.c`, `myfirst.h` et `myfirst.uir` dans la boîte de dialogue pour sélectionner les trois fichiers. Cliquez sur **Add** pour placer ces fichiers dans la section Selected Files de la boîte de dialogue.
4. Cliquez sur **OK** pour ajouter les fichiers dans le projet.

Exécuter le projet

Vous venez de terminer un projet LabWindows/CVI. Vous pouvez lancer le projet en sélectionnant **Run Project** dans le menu **Run**. Pour le moment, la seule commande active qui existe dans votre interface utilisateur est le bouton **Quit**. Dans le chapitre suivant, vous apprendrez à ajouter des fonctions à ce programme pour faire en sorte que le bouton **Acquire** acquière des données à afficher. Ensuite, vous obtiendrez une interface utilisateur complète capable d'acquérir et d'afficher des courbes.

Compléter le programme avec un driver d'instrument



Dans le chapitre précédent, vous avez créé votre premier projet `myfirst.prj` dans l'environnement LabWindows/CVI. Sachez toutefois qu'il ne s'agit que d'un squelette de programme. Dans ce chapitre, vous allez apprendre à utiliser un simple driver d'instrument pour acquérir des données, et ensuite tracer les données acquises dans la commande de graphe de votre interface utilisateur.

Un driver d'instrument se compose d'un ensemble de fonctions qui servent à programmer un instrument ou un groupe d'instruments associés. Les fonctions de haut niveau figurant dans un driver d'instrument englobent un grand nombre de fonctions de bas niveau, dont des opérations de lecture et d'écriture GPIB, VXI ou RS-232, de conversion de données ou de mise à l'échelle. L'exemple de module présenté dans ce chapitre ne communique pas avec un véritable instrument, mais illustre la façon dont on s'en sert pour acquérir une courbe depuis un oscilloscope.

Acquérir des courbes avec un driver d'instrument

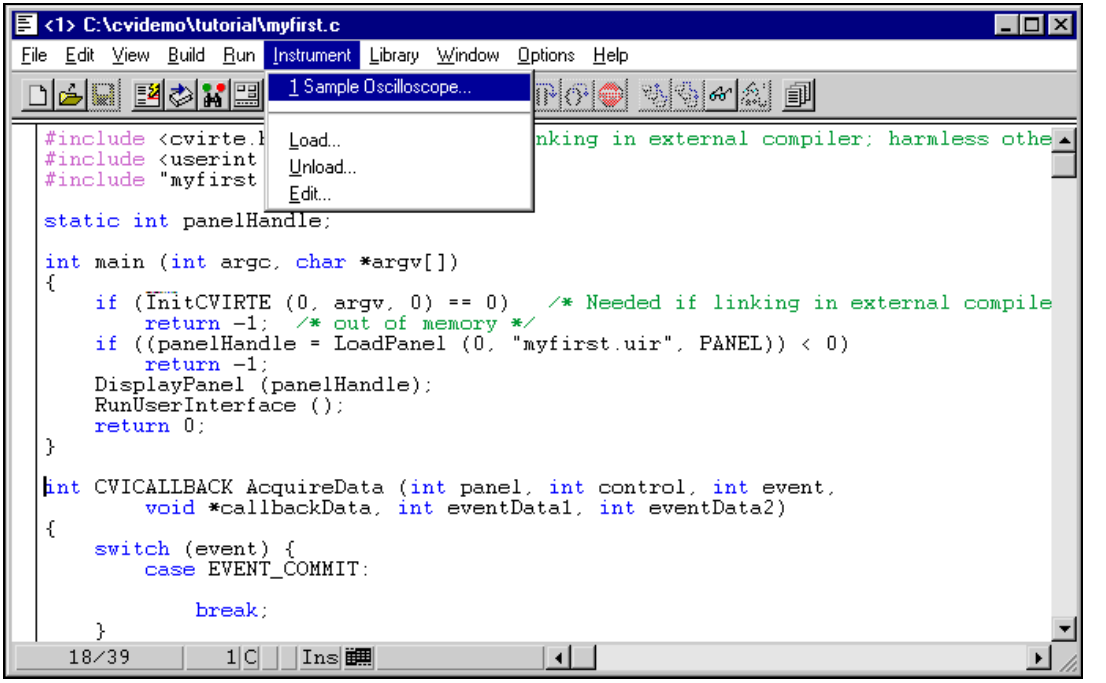
Pour être complet, il faut que votre programme lise un tableau de données générées à partir d'un driver d'instrument, puis qu'il les trace sur le graphe. Pour ce faire, vous devez modifier la fonction `AcquireData` dans le fichier source `myfirst.c`.

Charger le driver d'instrument

Le driver d'instrument dont vous avez besoin se compose de plusieurs fichiers, tous présents sur le disque dur. Vous y avez accès en chargeant le fichier (`.fp`) de l'instrument.

1. Sélectionnez **Add Files To Project»Instrument (*.fp)** dans le menu **Edit** de la fenêtre Projet.
2. Sélectionnez le fichier `scope.fp` dans le répertoire du tutorial. Cliquez sur **Add** puis sur **OK** pour ajouter le driver au projet.

3. Dans la fenêtre Projet, double-cliquez sur `myfirst.c` pour ouvrir la fenêtre d'édition de Source.



4. Placez le curseur sur la ligne vierge qui se trouve juste au-dessous de l'instruction `Event_Commit:` dans la fonction `AcquireData`, comme dans l'illustration précédente.

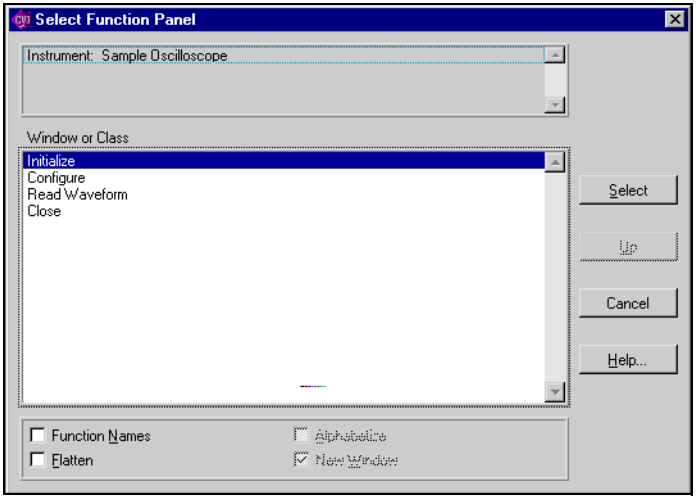
Initialiser l'instrument

Vous allez maintenant utiliser le driver d'instrument d'oscilloscope pour générer du code en vue d'acquérir une courbe. Vous verrez ensuite apparaître le code généré dans la fonction `AcquireData`. Rappelons que cette fonction s'exécute quand on clique sur le bouton **Acquire**.

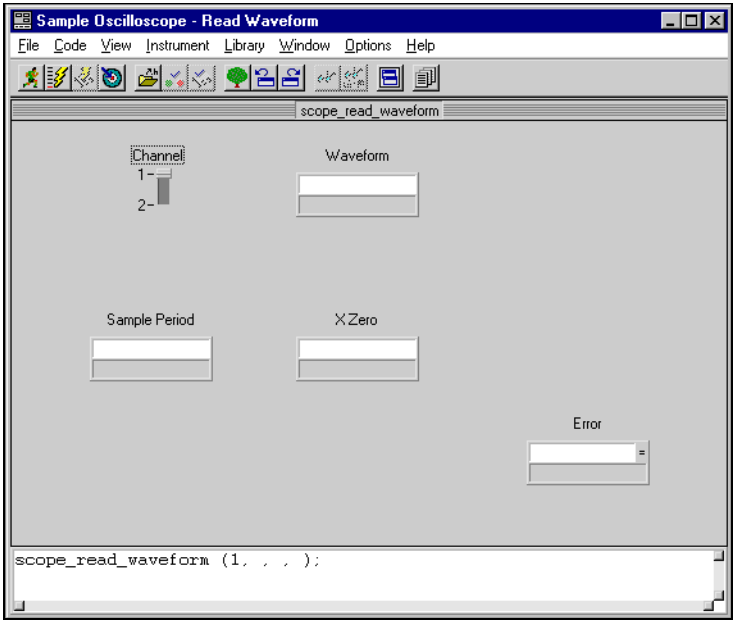
Étape 1 : ouvrez le panneau de fonction

1. Sélectionnez l'option **Sample Oscilloscope** dans le menu **Instrument**. L'exemple de driver d'oscilloscope contient quatre

fonctions simples qui servent à communiquer avec un oscilloscope, comme dans l'illustration suivante.



2. Mettez en surbrillance la fonction `Initialize` et cliquez sur le bouton **Select**. Vous verrez alors apparaître le panneau de fonction d'initialisation de l'oscilloscope.



Les panneaux de fonction sont avant tout des représentations graphiques des fonctions. Ce sont des outils interactifs qui servent à construire et à tester les appels de fonction de votre bibliothèque. Ces panneaux de fonction proposent une aide en ligne complète. Cliquez avec le bouton droit de la souris sur le panneau de fonction pour afficher l'aide en ligne de la fonction Initialize. Les panneaux de fonction génèrent aussi du code. Lorsque vous entrez des valeurs dans les champs du panneau de fonction, l'appel de fonction se crée automatiquement en bas de l'écran.

Étape 2 : initialisez la fonction du driver d'instrument

1. Entrez `1` dans le champ Address.
2. Entrez `err` dans le champ Error.
3. Déclarez la variable `err` en sélectionnant **Declare Variable** dans le menu **Code**.
4. Cliquez sur les options suivantes : Execute declaration et Add declaration to the top of target file. Puis cliquez sur **OK**.
5. Sélectionnez **Run Function Panel** dans le menu **Code**. Cliquez sur **Yes** lorsqu'une boîte de dialogue vous propose d'enregistrer toute modification avant l'exécution. Cette boîte de dialogue apparaît chaque fois que vous procédez à une modification.

Si aucune erreur n'est détectée en cours d'exécution, le paramètre Error vaut `0`. Sinon, vous pouvez cliquer avec le bouton droit de la souris sur le champ Error pour visualiser l'aide en ligne correspondante.

Une fois que vous avez rempli les paramètres et validé la ligne de code, suivez les étapes suivantes pour insérer l'appel de fonction dans le code et effacer le panneau de fonction à l'écran.

Étape 3 : insérez la fonction dans le fichier source

1. Sélectionnez **Insert Function Call** dans le menu **Code** pour recopier le code généré dans la fenêtre d'édition de Source.
2. Sélectionnez **Close** dans le menu **File** pour fermer le panneau de fonction.

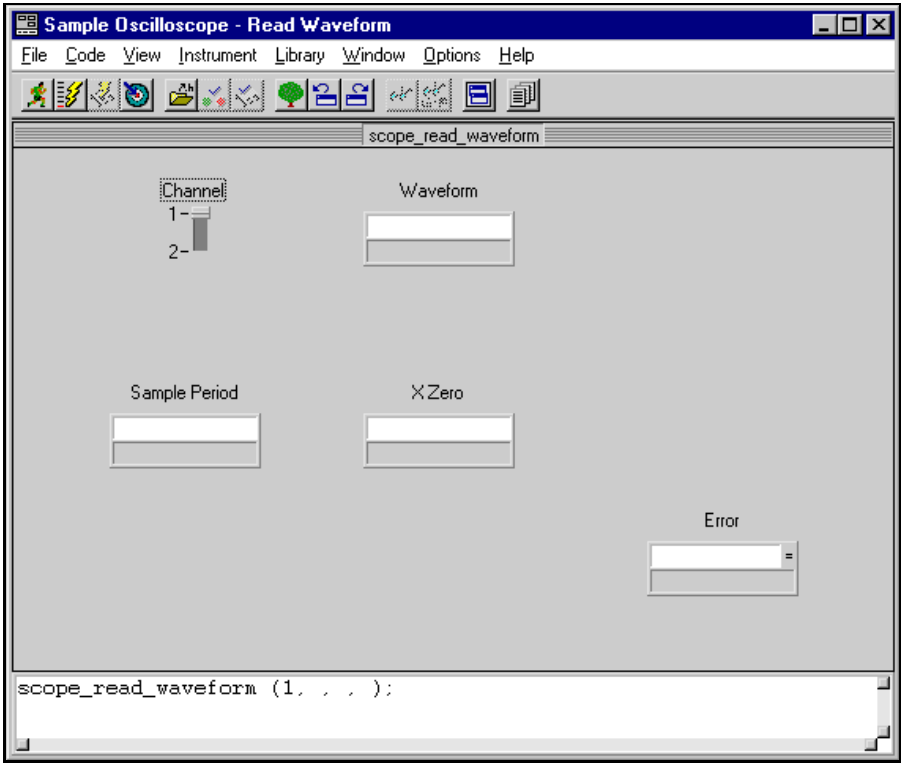
L'appel de fonction pour initialiser le driver d'instrument devrait maintenant s'afficher dans votre code source, au-dessous du code `Event_Commit` :

```
err = scope_init (1);
```

Lire les données de l'instrument

L'une des fonctions les plus importantes de tout driver d'instrument consiste à lire les données émises par un instrument et à convertir les données brutes obtenues en un format directement utilisable par votre programme. Par exemple, un oscilloscope numérique renvoie une courbe sous forme de chaîne de nombres ASCII séparés par une virgule. Ensuite, le driver d'instrument analyse la chaîne de caractères, met à l'échelle les données en tension, puis les place dans un tableau en mémoire à votre place.

1. Sélectionnez **Read Waveform** dans le driver d'instrument **Sample Oscilloscope** du menu **Instrument**. Le panneau de fonction Read Waveform s'affiche alors de la façon suivante.

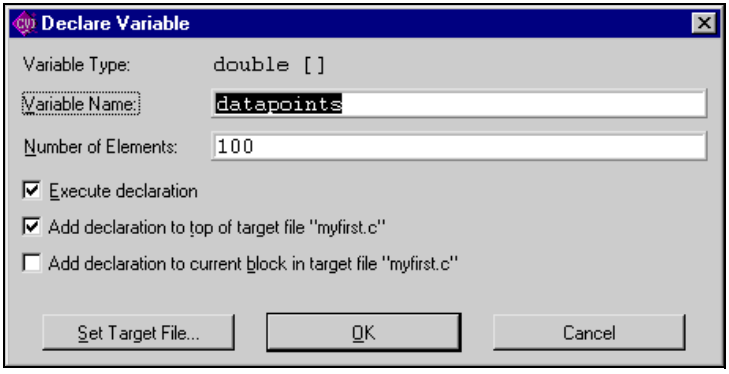


2. Réglez le champ Channel sur 2. Channel 1 est une onde sinusoïdale et Channel 2 un ensemble de données aléatoires.

Déclarer les variables dans les panneaux de fonction

La fonction `Read Waveform` sert à présenter les données sous forme de tableau. Avant de pouvoir exécuter cette fonction, il faut déclarer un tableau. Vous pouvez déclarer des variables, comme par exemple un tableau, à partir d'un panneau de fonction. Pour ce faire, suivez les étapes suivantes :

1. Cliquez sur le champ Waveform et tapez `datapoints`.
2. Sélectionnez **Declare Variable** dans le menu **Code** pour déclarer la variable `datapoints` en mémoire. Vous voyez alors apparaître une boîte de dialogue avec la variable `datapoints` automatiquement insérée dans le champ Variable Name.
3. Tapez `100` dans le champ Number of Elements.
4. Vérifiez que la première case Add declaration est bien cochée. La boîte de dialogue devrait ressembler à l'illustration suivante.



5. Cliquez sur **OK** pour déclarer le tableau de points de données.

Compléter le panneau de fonction Read Waveform

Terminez la configuration du panneau de fonction avant de l'exécuter de la façon suivante :

1. Cliquez sur le champ Sample Period pour le mettre en surbrillance.
2. Sélectionnez **Declare Variable** dans le menu **Code**.
3. Tapez `delta_t` dans le champ Variable Name. Vérifiez que la première case Add declaration est bien cochée, puis cliquez sur **OK**.
4. Cliquez sur le champ X Zero pour le mettre en surbrillance.
5. Sélectionnez **Declare Variable** dans le menu **Code**.

6. Tapez `x_zero` dans le champ Variable Name. Vérifiez que la case "Execute declaration" et que la première des deux cases "Add declaration..." sont bien toutes les deux cochées, puis cliquez sur **OK**.
7. Tapez `err` dans le champ Error.
8. Sélectionnez **Run Function Panel** dans le menu **Code** pour exécuter le panneau de fonction. Enregistrez les changements avant de lancer l'exécution. Si vous avez correctement exécuté la fonction initialize, le champ Read Waveform Error devrait afficher `0`. Une fois la fonction exécutée, vous voyez apparaître une rangée de cases dans le cadre Waveform, ce qui signifie que les données figurent bien dans le tableau.
9. [Optionnel] Vous pouvez double-cliquer sur les tirets qui se trouvent dans le champ Waveform. Cela activera l'éditeur de tableau et vous permettra d'inspecter les valeurs récupérées.
10. Sélectionnez **Insert Function Call** dans le menu **Code** pour recopier le code généré dans la fenêtre d'édition de Source.
11. Fermez le panneau de fonction. La ligne de code suivante existe désormais dans la fonction "Callback" `AcquireData` :

```
err = scope_read_waveform (2, datapoints, &delta_t, &x_zero);
```

Afficher la courbe sur le graphe

Jusqu'à présent, vous avez suivi les étapes nécessaires à l'acquisition des données. L'étape suivante consiste à utiliser la fonction Plot Y, qui sert à tracer le tableau des données acquises dans le graphe de l'interface utilisateur.

Pour afficher et sélectionner la fonction Plot Y, suivez les étapes suivantes :

1. Ouvrez le panneau de fonction Plot Y en sélectionnant les options suivantes. Dans le menu **Library**, sélectionnez **User Interface**. Dans la boîte de dialogue qui s'affiche, sélectionnez **Controls/Graphs/Strip Charts»Graph and Strip Charts»Graph Plotting and Deleting»Plot Y**.

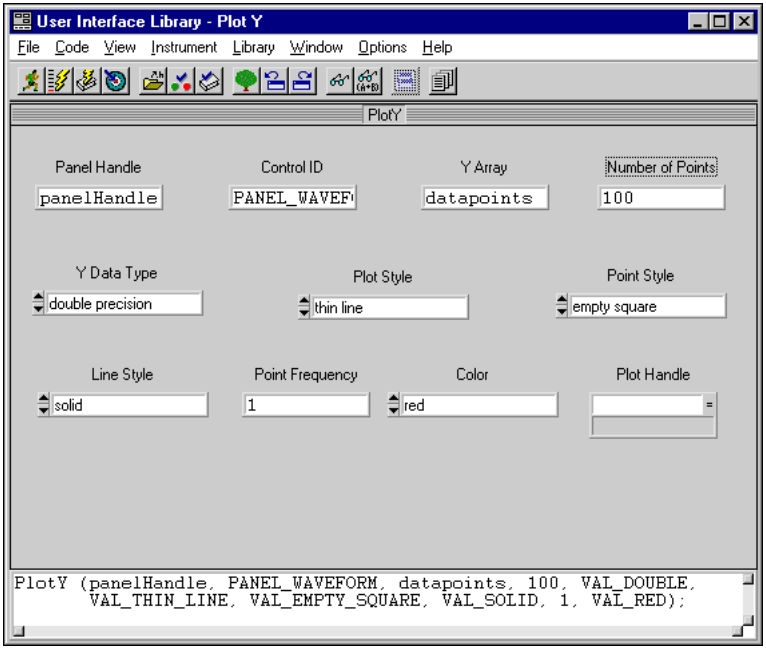
2. Remplissez les champs du panneau de fonction de la façon suivante :

   Panel Handle: `panelHandle`

   Control ID: `PANEL_WAVEFORM`

   Y Array: `datapoints`

   Number of Points: `100`



3. Dans le panneau de fonction `PlotY`, sélectionnez **Insert Function Call** dans le menu **Code** pour insérer la fonction `PlotY` dans votre code source.
4. Fermez le panneau de fonction. Les modifications que vous avez apportées dans la fonction "Callback" `AcquireData` devraient correspondre au code source présenté dans la section suivante.
5. Dans la fenêtre d'édition de Source, sélectionnez **Save** dans le menu **File** pour enregistrer les modifications. Votre programme est maintenant terminé.

Afficher le programme

Une fois terminé, votre fichier source devrait ressembler au code suivant :

```
static double x_zero;
static double delta_t;
static double datapoints[100];
static int err;
#include <cvirte.h> /* Needed if linking in external compiler; harmless otherwise */
#include <userint.h>
#include "myfirst.h"

static int panelHandle;

int main (int argc, char *argv[])
{
if (InitCVIRTE (0, argv, 0) == 0)/* Needed if linking in external compiler; harmless
otherwise */
return -1;/* out of memory */
if ((panelHandle = LoadPanel (0, "myfirst.uir", PANEL)) < 0)
return -1;
DisplayPanel (panelHandle);
RunUserInterface ();
return 0;
}

int CVICALLBACK AcquireData (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
switch (event) {
case EVENT_COMMIT:
err = scope_init (1);
err = scope_read_waveform (2, datapoints, &delta_t, &x_zero);
PlotY (panelHandle, PANEL_WAVEFORM, datapoints, 100, VAL_DOUBLE,
VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_RED);

break;
}
return 0;
}

int CVICALLBACK Shutdown (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
```

```
switch (event) {
case EVENT_COMMIT:
QuitUserInterface (0);
break;
}
return 0;
}
```

Exécuter le projet

Vous disposez désormais d'un projet complet, nommé `myfirst.prj`. Vous pouvez visualiser le statut de chaque fichier associé au projet de la fenêtre Projet et modifier chacun d'entre eux en double-cliquant sur le nom du fichier choisi.

Sélectionnez **Run Project** dans le menu **Run** pour exécuter le code. Au cours du processus de compilation, LabWindows/CVI va reconnaître que votre programme nécessite le fichier d'en-tête `scope.h`. Il va donc proposer de l'insérer dans votre code source. Cliquez sur **Yes** pour ajouter ce fichier inclus dans votre programme. Pendant que votre programme se compile et s'exécute, voici ce qui se passe :

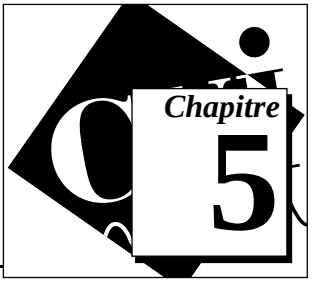
- LabWindows/CVI compile le code source `myfirst.c` qu'il lie ensuite avec les bibliothèques correspondantes dans LabWindows/CVI.
- L'interface utilisateur s'affiche, prête à recevoir des informations du clavier ou de la souris.
- Lorsque vous cliquez sur le bouton **Acquire**, LabWindows/CVI branche l'exécution du programme sur la fonction "Callback" `AcquireData`.
- La fonction `AcquireData` lit les données en provenance de l'instrument simulé avant de les tracer sur la commande de graphe de l'interface utilisateur.

☞ **Remarque** ***Lorsque vous cliquez sur* Acquire *plusieurs fois de suite, les données des acquisitions précédentes restent affichées. Vous pouvez modifier la fonction*** `AcquireData` ***pour y inclure la fonction*** `DeleteGraphPlot` ***comme dans l'extrait de code suivant. Cela permet de n'afficher qu'une seule acquisition à chaque fois que vous lancez le programme.***

Note ***Vous trouverez le panneau de fonction DeleteGraphPlot dans le menu Library sous User Interface»Controls/Graphs/Strip Charts»Graphs et Strip Charts»Graph Plotting et Deleting.***

```
int CVICALLBACK AcquireData (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
switch (event) {
case EVENT_COMMIT:
DeleteGraphPlot (panelHandle, PANEL_WAVEFORM, -1, VAL_IMMEDIATE_DRAW);
err = scope_read_waveform (chan_num+1, datapoints, &delta_t, &x_zero);
PlotY (panelHandle, PANEL_WAVEFORM, datapoints, 100, VAL_DOUBLE,
VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_RED);
break;
}
return 0;
}
```

Ajouter un moniteur de températures au programme



Dans le chapitre précédent, vous vous êtes servi d'un driver d'instrument pour ajouter du code à votre programme afin de rassembler des données sur la courbe. Dans ce chapitre, vous allez modifier votre programme pour qu'il rassemble des données de température en utilisant une fonction "Callback" du Timer afin de procéder à l'acquisition de données synchrone. Pour ce faire, vous allez utiliser un driver d'acquisition de données simulées pour acquérir et afficher une température sur un thermomètre.

☞ **Remarque** ***Vous devez avoir construit le projet*** `myfirst.prj` ***pour pouvoir faire les exercices de ce chapitre. Pour ce faire, il faut que vous ayez terminé de lire les chapitres 2, 3 et 4 de ce guide d'évaluation.***

Les contrôles Timer

Dans le dernier exemple, vous avez utilisé un bouton **Acquire** pour générer un événement. Les contrôles Timer servent à générer automatiquement des événements à des intervalles spécifiques. Dans notre exemple, vous allez utiliser un contrôle Timer pour déclencher une opération d'acquisition de données toutes les secondes afin de lire la température. Vous devez ajouter un contrôle Timer ainsi qu'un indicateur thermomètre pour afficher la température acquise sur votre interface utilisateur.

1. Ouvrez le projet `myfirst.prj`.
2. Double-cliquez sur `myfirst.uir` dans la fenêtre Projet. Dans le menu **Create**, sélectionnez **Timer**. Vous verrez alors apparaître un contrôle Timer sur votre interface utilisateur. Vous pouvez placer le contrôle n'importe où dans votre panneau. Il n'apparaîtra pas sur l'interface utilisateur graphique lorsque vous lancerez le programme.

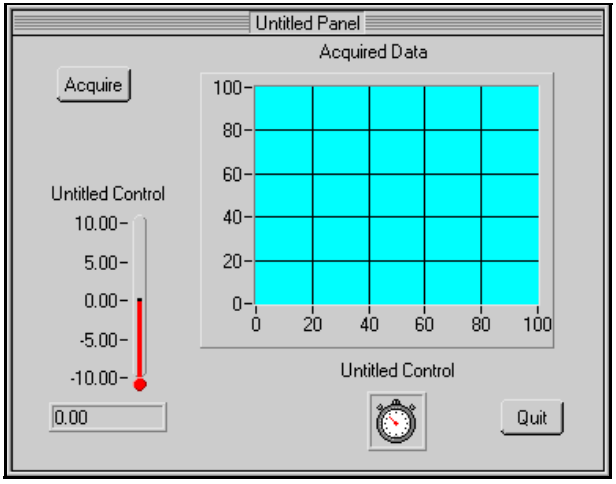
3. Double-cliquez sur le contrôle Timer pour afficher la boîte de dialogue **Edit Timer**. Modifiez ensuite les options suivantes de la boîte de dialogue, puis cliquez sur **OK** pour enregistrer ces paramètres :

   Constant Name: `ACQUIRE_SYNC`

   Callback Function: `SyncAcquire`

   Interval: `0.100`

4. Dans le menu **Create**, sélectionnez **Numeric»Thermometer** pour ajouter un thermomètre dans votre interface. Placez le thermomètre dans une zone libre comme dans l'illustration suivante.



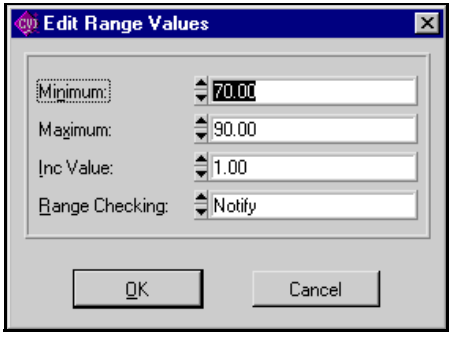
5. Double-cliquez sur le thermomètre pour afficher la boîte de dialogue **Edit Thermometer**. Modifiez ensuite les options suivantes de la boîte de dialogue :

   Constant Name: `THERMOMETER`

   Default Value: `70` (vous pouvez cliquer sur **OK** si vous voyez le message hors gamme)

   Label: `Temperature`

6. Dans la boîte de dialogue **Edit Thermometer**, il faut également que vous changiez la gamme du thermomètre. Cliquez sur **Range Values** pour afficher la boîte de dialogue suivante. Réglez-y les valeurs comme dans l'illustration suivante, puis cliquez sur **OK**.



7. Une fois que vous avez réglé Constant Name, Default Value, Range Values et Label, cliquez sur **OK** pour fermer et enregistrer vos paramètres dans la boîte de dialogue **Edit Thermometer**.
8. Enregistrez le fichier de l'interface utilisateur en sélectionnant **Save** dans le menu **File**. Puis fermez la fenêtre de l'Éditeur d'interface utilisateur.

Vous avez terminé d'apporter les modifications nécessaires dans l'interface utilisateur.

Ajouter la fonction “Callback” du Timer au programme

Maintenant que vous avez procédé aux ajouts nécessaires dans votre interface utilisateur, il faut que vous ajoutiez la fonction “Callback” du Timer dans votre code source. Dans un premier temps, ouvrez la fenêtre d’édition de Source `myfirst.c`. Dans l’Éditeur d’interface utilisateur, cliquez sur le contrôle d’horloge pour le mettre en surbrillance. Cliquez dessus avec le bouton droit de la souris, puis sélectionnez **Generate Control Callback.** La fonction “Callback” du contrôle Timer est alors insérée en bas de `myfirst.c`. Cliquez sur la fenêtre d’édition de Source pour visualiser le code. L’insertion devrait figurer ainsi :

```
int CVICALLBACK SyncAquire (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
switch (event) {
case EVENT_TIMER_TICK:

break;
}
return 0;
}
```

☞ **Remarque** ***Souvenez-vous des points suivants en insérant des fonctions dans un fichier de code source :***

- ***Le fichier de code source que vous souhaitez modifier doit être ouvert.***
- ***Lorsque vous insérez un appel de fonction d’un panneau de fonction, LabWindows/CVI insère le nouveau code là où se trouve le curseur dans votre fichier de code source.***
- ***Lorsque vous insérez une fonction “Callback” de l’Éditeur d’interface utilisateur, LabWindows/CVI insère le nouveau code à la fin de votre fichier de code source.***

Ajouter une fonction d'acquisition de données à la fonction "Callback" du Timer

Maintenant que la fonction "Callback" du Timer se trouve insérée dans le programme, il faut que vous y ajoutiez une fonction d'acquisition de données (DAQ) simulées, de la façon suivante :

1. Ce guide d'évaluation propose une carte d'acquisition de données simulées capable de lire les températures. Dans la fenêtre Projet, sélectionnez **Add Files To Project…Instrument (*.fp)** dans le menu **Edit**.
2. Sélectionnez le fichier `simpldaq.fp` dans le répertoire `cvi\tutorial`. Cliquez sur **Add**, puis sur **OK** pour ajouter le driver au projet.
3. Placez votre curseur sur la ligne vierge au-dessous de `EVENT_TIMER_TICK`: dans la fonction "Callback" `SyncAcquire`.
4. Sélectionnez **Instrument** dans la barre de menus pour vérifier que le driver d'instrument a bien été chargé. Cliquez sur **Simple Data Acquisition…** pour l'ouvrir.
5. La boîte de dialogue **Select Function Panel** contient actuellement une seule et unique fonction : `Read_Value`. Cliquez sur **Select** pour afficher la boîte de dialogue de ce panneau de fonction.
6. Lorsque cette boîte de dialogue apparaît, cliquez sur le contrôle **Temperature**, puis tapez `temp_value` comme nom de variable.
7. Vous devez déclarer la variable `temp_value`. Pour ce faire, sélectionnez **Code»Declare Variable…** dans la barre de menus. Vérifiez que les cases Execute declaration et Add declaration to top of target file sont bien toutes les deux cochées. Puis cliquez sur **OK**.
8. Placez votre curseur dans le contrôle **Error** puis tapez `err`. étant donné que vous avez déjà déclaré cette variable, inutile de procéder à une nouvelle déclaration.

9. Dans la barre de menus, sélectionnez **Code»Insert Function Call**, puis fermez la fenêtre du panneau de fonction. Votre fonction “Callback” `SyncAcquire` contient désormais un appel de fonction :

```
int CVICALLBACK SyncAcquire (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
switch (event) {
case EVENT_TIMER_TICK:
err = simpldaq_read_value (1, 1, &temp_value);

break;
}
return 0;
}
```

Ajouter une fonction pour mettre à jour le thermomètre

Votre programme contient maintenant une fonction d’acquisition de données simulées. Le programme a aussi besoin d’une fonction pour mettre à jour le thermomètre en utilisant les nouvelles données acquises. Suivez les étapes suivantes pour ajouter cette fonction et terminer le programme :

1. Dans la fenêtre d’édition de Source, placez le curseur sur la ligne vierge en dessous de la fonction `simpldaq_read_value`.
2. Ouvrez le panneau de fonction Set Control Value. Dans la barre de menus, sélectionnez **Library»User Interface**. Puis dans la boîte de dialogue qui s’affiche, sélectionnez **Controls/Graphs/Strip Charts»General Functions»Set Control Value**.
3. Remplissez les options du panneau de fonction de la façon suivante :

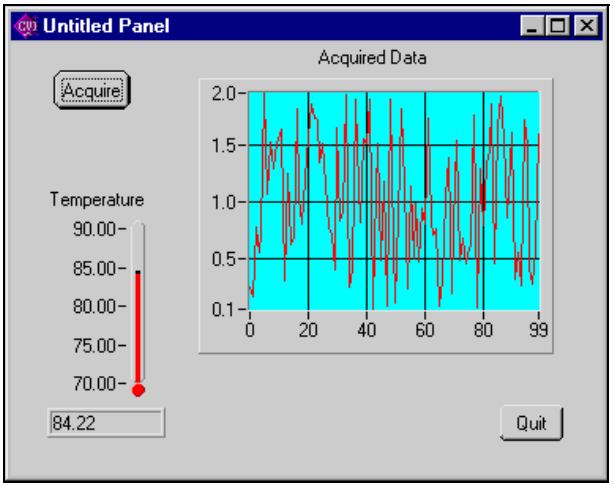
Panel Handle:	`panelHandle`
Control ID:	`PANEL_THERMOMETER`
Value:	`temp_value`

4. Dans la barre de menus, sélectionnez **Code»Insert Function Call** pour insérer l'appel de fonction `SetCtrlVal` dans votre fonction "Callback" du Timer. Fermez la fenêtre du panneau de fonction et retournez dans la fenêtre d'édition de Source. Votre fonction "Callback" devrait correspondre à l'extrait suivant :

```
int CVICALLBACK SyncAcquire (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
switch (event) {
case EVENT_TIMER_TICK:
err = simpldaq_read_value (1, 1, &temp_value);
SetCtrlVal (panelHandle, PANEL_THERMOMETER, temp_value);

break;
}
return 0;
}
```

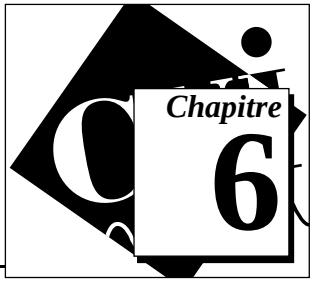
Exécuter le projet

Sélectionnez **Run Project** dans le menu **Run**. Pendant le processus de compilation, LabWindows/CVI va reconnaître que votre programme manque l'instruction du fichier d'en-tête `simpldaq.h`. Choisissez **Yes** pour ajouter le fichier inclus dans votre programme, puis enregistrez le fichier quand on vous le demande.



Lorsque le programme s’exécute, le Timer génère des événements à intervalle régulier. Une fois l’événement généré, la fonction “Callback” du Timer est appelée, et le thermomètre affiche la nouvelle température. La fonction d’acquisition de données s’exécute en arrière-plan, permettant ainsi aux utilisateurs d’acquérir et de tracer une courbe à partir de d’instrument oscilloscope à chaque fois qu’ils cliquent sur **Acquire**.

LabWindows/CVI : la solution idéale



Bien que vous ayez terminé de lire ce guide d'évaluation, rappelez-vous toutefois que vous n'avez fait qu'effleurer la programmation avec LabWindows/CVI. En effet, ses capacités vont bien au-delà de ce que vous avez abordé jusqu'à présent. LabWindows/CVI offre en effet une solution idéale pour vos besoins de programmation en matière d'instrumentation, avec des caractéristiques en perpétuelle amélioration et un grand nombre de boîtes à outils complémentaires.

Des caractéristiques supplémentaires pour répondre à vos besoins

National Instruments ne cesse d'améliorer et d'étendre les possibilités de LabWindows/CVI en ajoutant des boîtes à outils et des options de support matériel.

Les caractéristiques

LabWindows/CVI garantit une entière compatibilité avec les outils de programmation standards. Toutes les bibliothèques d'instrumentation, d'analyse et d'interface utilisateur de LabWindows/CVI sont disponibles sous forme de DLL standards 32 bits pour les environnements de programmation C/C++ de Microsoft, Borland, Symantec et WATCOM. En plus des fichiers (`.exe`) standards exécutables, LabWindows/CVI vous permet également de construire des DLL 32 bits compatibles non seulement avec vos outils C/C++ à usage universel, mais aussi avec Visual BASIC et LabVIEW.

Comme toujours, LabWindows/CVI vous offre la solution de driver d'instrument la plus complète du marché : soit quelque 600 drivers pour des instruments proposés par plus de 65 instrumentiers différents. La bibliothèque de drivers d'instruments inclut des drivers pour les instruments GPIB, VXI, série et CAMAC. En outre, l'Alliance Systems VXI*plug&play* a adopté LabWindows/CVI comme technologie essentielle et fondement à la normalisation industrielle.

Les plates-formes

LabWindows/CVI supporte les systèmes d'exploitation et les plates-formes informatiques suivants :

Plate-forme supportée	Configuration système requise	Compilateurs compatibles
Windows 95/NT/3.1	486/33 avec capacité FPU recommandée, 25 Mo de disque dur, écran VGA	Microsoft Visual C++, Borland C++, WATCOM C, Symantec C (sous les systèmes d'exploitation Windows 32 bits uniquement)
Sun Solaris	Station de travail SPARCstation 1, 24 Mo de mémoire principale, 32 Mo d'espace disque de mémoire swap, 12 Mo d'espace disque pour les fichiers d'application	Compilateur Sun C ANSI (acc), compilateur GNU C (gcc)
HP-UX (bibliothèques Run-time uniquement)	Station de travail HP-UX série 700, 24 Mo de mémoire principale, 8 Mo d'espace disque dur disponibles avant installation, système d'exploitation HP-UX 9.05 ou plus	Compilateur HP-UX C, compilateur GNU C (gcc)

Des boîtes à outils complémentaires

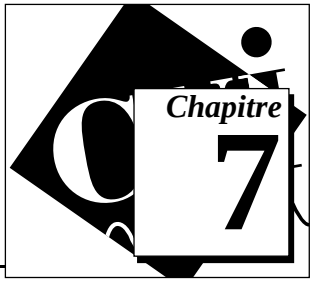
National Instruments propose toute une série de boîtes à outils complémentaires à LabWindows/CVI qui vous serviront dans des domaines d'application bien spécifiques. Ces boîtes à outils sont des bibliothèques ou des utilitaires qui répondent aux besoins des marchés, industries ou domaines d'application très spécialisés.

- **Test Executive** : il s'agit d'un système de gestion de tests qui gère le séquencement de tests, le bouclage et l'enregistrement de données pour des situations de test en production. Vos efforts peuvent s'orienter sur le développement de modules de tests individuels en laissant le Test Executive gérer et contrôler les tests à votre place. Voir le SPC Toolkit.
- **SQL Toolkit (Windows uniquement)** : il s'agit d'un ensemble de bibliothèques que vous pouvez utiliser pour vous connecter à plus de 30 bases de données locales ou déportées. Avec ces bibliothèques, vous pouvez connecter vos programmes de test ou le Test Executive directement à une base de données pour enregistrer les résultats ou télécharger les paramètres de test.
- **SPC Toolkit** : il s'agit d'une bibliothèque de fonctions de maîtrise statistique de processus qui servent à analyser la qualité de la production en utilisant des opérations de statistique de processus, de graphes déroulants de contrôle, et d'analyse Pareto. Voir le Test Executive toolkit.
- **PID Control Toolkit** : il s'agit d'un ensemble d'algorithmes pour le contrôle PID que vous pouvez intégrer dans vos applications de gestion et de contrôle de processus.
- **Internet Developers Toolkit (Windows uniquement)** : il s'agit d'une bibliothèque de fonctions qui vous aident à créer un serveur Web en vue d'afficher vos panneaux d'interface utilisateur LabWindows/CVI sur Internet. Les utilisateurs du Web peuvent ainsi cliquer sur ces panneaux pour interagir avec vos applications LabWindows/CVI. Vous pouvez également envoyer des messages E-mail à partir de vos applications et de vos fichiers de transfert vers et depuis les serveurs FTP.
- **Digital Filter Design Toolkit** : il s'agit d'un outil de conception polyvalent destiné au conditionnement de signaux, aux systèmes de commande ainsi qu'au traitement de signaux numériques.
- **Third-Octave Analysis Toolkit (Windows uniquement)** : il s'agit d'un analyseur basé PC en tiers d'octave prêt-à-l'emploi.

Quelques exemples supplémentaires

Une fois que vous avez fini de lire ce guide d'évaluation, vous pouvez utiliser le *Manuel d'initiation à LabWindows/CVI* pour approfondir le fonctionnement de LabWindows/CVI. Ce tutorial est consultable sur le CD-ROM Software Showcase. Le site Web de National Instruments offre des exemples supplémentaires qui présentent les concepts fondamentaux de LabWindows/CVI à l'adresse suivante : `www.natinst.com/cvi`.

L’engagement de National Instruments



LabWindows/CVI est le fruit d’un engagement de longue date pris par National Instruments : offrir des outils qui simplifient le développement de systèmes d’instrumentation en utilisant des langages standards. En choisissant LabWindows/CVI comme environnement de développement, vous enrichissez le nombre de scientifiques et d’ingénieurs qui profitent déjà de la puissance de la programmation C ANSI et de la souplesse de Windows pour leur système d’instrumentation.

Une compatibilité à long terme

LabWindows/CVI est l’aboutissement d’années de développement, de support permanent et d’efforts pour améliorer les outils de programmation. Les utilisateurs de LabWindows dans l’environnement DOS peuvent immédiatement traduire leurs programmes pour fonctionner dans Windows 3.1, Windows 95 et Windows NT en utilisant LabWindows/CVI. Ce souci de supporter et de mettre à jour ne s’arrête pas à Windows. En effet, la possibilité vous est désormais offerte de développer des programmes dans LabWindows/CVI sous Microsoft Windows et de les porter sur des SPARCstations Sun sous Solaris ou sur des stations de travail HP série 700 sous HP-UX. National Instruments continuera aussi à supporter et à mettre à jour LabWindows/CVI sur les futurs systèmes d’exploitation. Lorsque LabWindows/CVI sera disponible sur ces futurs systèmes d’exploitation, vous verrez que tous vos efforts actuels de développement seront récompensés.

La formation des clients

National Instruments propose des cours de formation approfondis de trois jours sur LabWindows/CVI. Vous y apprendrez à construire rapidement vos applications. Les informations et les astuces de développement qui vous y sont présentées peuvent accroître votre productivité dans LabWindows/CVI. Ces cours sont proposés tous les mois dans nos locaux. Sachez en outre que vous sont également

proposées des formations de deux jours sur le GPIB, l'acquisition de données et le VXI qui devraient vous renseigner sur les systèmes dans leur totalité.

Le programme Alliance

Le programme Alliance est un réseau de conseillers et de développeurs tierce partie tous spécialisés dans LabWindows/CVI. Le manuel National Instruments *Solutions* répertorie les bibliothèques et les utilitaires supplémentaires développés par les membres de notre programme Alliance pour vous aider à utiliser LabWindows/CVI. Il vous fournit par ailleurs les coordonnées des spécialistes LabWindows/CVI que vous pouvez solliciter pour vous aider à personnaliser vos applications.

Le support technique

National Instruments vous fournit une multitude de précieuses informations qui vous aideront à compléter vos applications. Vous pouvez utiliser nos sites Internet (Web et FTP), BBS ou des systèmes Fax-on-Demand pour télécharger des informations importantes ou des exemples de produits, des documents informatifs ou des astuces relatives au développement technique. Un forum technique sur LabWindows/CVI existe sur Internet pour que vous puissiez discuter de vos problèmes avec d'autres utilisateurs de LabWindows/CVI. Par ailleurs, National Instruments vous propose les services d'ingénieurs hautement qualifiés qui vous assurent une assistance technique rapide et efficace.

Les clients de National Instruments ont le choix entre les différentes options de support techniques suivantes :

Support Web

Site Web : `www.natinst.com`

Support FTP

Site FTP : `ftp.natinst.com`

Support par courrier électronique (E-mail)

`support@natinst.com`
`lw.support@natinst.com`

Support Bulletin Board

BBS États-Unis : (512) 794-5422
BBS Grande-Bretagne : 01635 551422
BBS France : 01 48 65 15 59

Support Fax-on-Demand aux États-Unis

(512) 418-1111

Support téléphonique aux États-Unis

(512) 795-8248
(512) 794-5678 (Fax)

Les filiales

Allemagne 089 741 31 30, Australie 03 9879 5166,
Autriche 0662 45 79 90 0, Belgique 02 757 00 20,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521,
Corée 02 596 7456, Danemark 45 76 26 00, Espagne 91 640 0085,
États-Unis 512 794 0100, Finlande 09 725 725 11,
France 01 48 14 24 24, Grande-Bretagne 01635 523545,
Hong Kong 2645 3186, Israël 03 573 4815, Italie 02 413091,
Japon 03 5472 2970, Mexique 5 520 2635, Norvège 32 84 84 00,
Pays-Bas 0348 433466, Singapour 2265886, Suède 08 730 49 70,
Suisse 056 200 51 51, Taïwan 02 377 1200

National Instruments Corporate Headquarters

6504 Bridge Point Parkway, Austin, TX 78730-5039
Tel. : (512) 794-0100

National Instruments en France

National Instruments France
Centre d'Affaires Paris-Nord
Immeuble "Le Continental"
BP 217
93153 Le Blanc-Mesnil Cedex

Formulaire de documentation

National Instruments vous invite à apporter vos commentaires sur la documentation fournie avec nos produits. Ces informations permettront de garantir des produits de qualité adaptés à vos besoins.

Titre : *Guide d'évaluation LabWindows®/CVI*

Date d'édition : Septembre 1997

Référence : 350349A-01

Veuillez apporter vos commentaires sur l'exhaustivité, la clarté et l'organisation de ce manuel.

_____

_____

_____

_____

_____

Si vous avez rencontré des erreurs, veuillez noter le(s) numéro(s) de pages correspondants et décrire leur nature.

_____

_____

_____

_____

_____

Nous vous remercions de votre collaboration.

Nom _____

Fonction _____

Société _____

Adresse _____

Téléphone (_____)_____ Fax (_____)_____

Adressez votre courrier à :

National Instruments
Centre d'Affaires Paris-Nord
Immeuble "Le Continental"
BP 217
93153 Le Blanc-Mesnil Cedex
Tél. : 01 48 14 24 24
Fax : 01 48 14 24 14