

LabVIEW Debugging

Bonnes pratiques de développeurs

Secrets d'architectes



Amaury LAURENT
(CLA, CLED)
NERYS



Luc DESRUELLE
LabVIEW Champion, CLA, CTD
MESULOG (Director)



Nerys Group companies

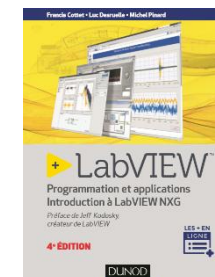


Luc DESRUELLE

Président - MESULOG & Nerys

- Chef de projet Test & Mesure (architecture, déploiement, industrialisation)
- Analyse et valorisation des données mesurées (Prédiction, statistique, IA)
- Cybersécurité et gouvernance des projets numériques (OS DR)
- LabVIEW Architecte & Champion, TestStand Développeur

ld@mesulog.fr



“Nous accompagnons les équipes industrielles et R&D dans la conception, la sécurisation et l’optimisation de leurs systèmes de test, en combinant expertise logicielle, analyse des données et intelligence artificielle.”



Amaury LAURENT

Responsable systèmes embarqués – NERYS

- Chef de projet Test & Mesure
- Développeur LabVIEW depuis 2014
- Certifié Développeur Systèmes Embarqués et Architecte LabVIEW

ARCHITECT



EMBEDDED
DEVELOPER



Nerys Group



CONCEPTION BANCS D'ESSAIS ET DE SYSTÈMES DE MESURE
Métrologie, R&D, caractérisation, endurance, Contrôle

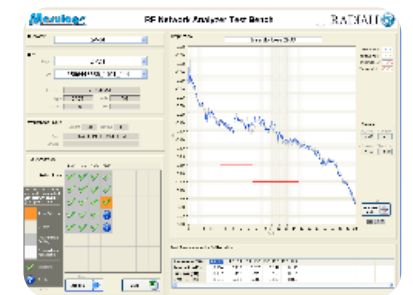
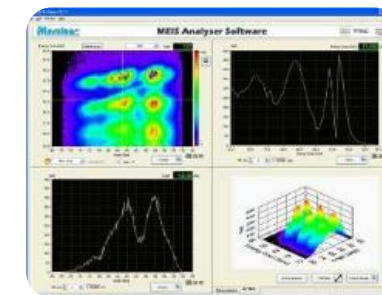
CONCEPTION DE BANCS D'ESSAIS

- Banc de test
- Baies de contrôle commande
- Développement cartes électroniques
- Modélisation et conception mécanique
- Systèmes de mesure
- Développement logiciel



LOGICIELS DÉDIÉS AU TEST ET À LA MESURE

- Expert en logiciel test et mesure
- Expert en logiciel de tests automatisés
- NI LabVIEW et TestStand
- Partenaire NI Emerson
- Optimisation, fiabilisation de code
- Architecture projets complexes
- Analyse de données



“Nous accompagnons les équipes industrielles et R&D dans la conception de leurs systèmes de test, en combinant expertise logicielle, sélection de matériels, analyse des données et intelligence artificielle.”

Nerys Group : 4 pôles d'activités

1. Réalisation de système Test & Measure (complet, en partie ou rénovation) :

- Banc essai (armoie électrotechnique, tiroir, valise d'essai)
- Machine spéciale (BE mécanique, BE électrotechnique et BE logiciel)

2. Développement logiciel

- Personnalisé sous NI LabVIEW & Python : spécialisé test et mesure, analyse données (IA)
- Test automatisé sous NI TestStand
- Système embarqué NI sous LabVIEW RT / FPGA
- VHDL, Visual Basic, C#, .NET, C, C++, Python

3. Editeur de solutions « clé en main »

- Suite Logiciels « Vasco » pour pilotage de banc d'essai - notamment système NI RT/FPGA
- Produits pour le séquenceur NI TestStand

4. Service & Formation Qualiopi

Gérer des projets dans une **démarche qualité** et **cybersécurité** (OS DRSF)

Nerys Group : notre équipe

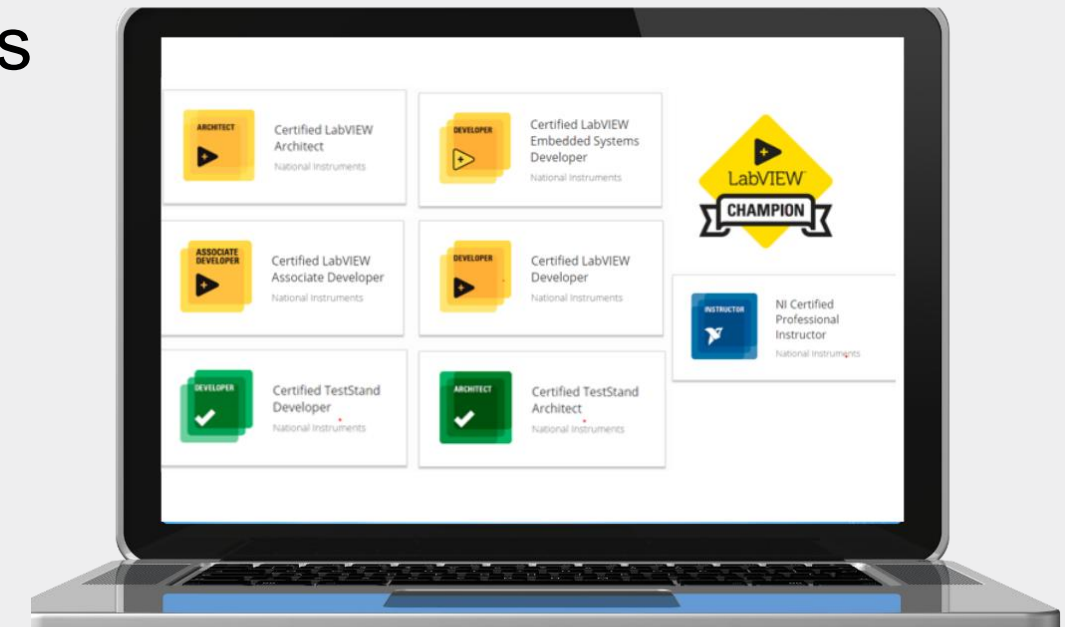
Un pôle logiciel qualifié : 13 développeurs expérimentés

Des certifications :

- Certified LabVIEW Architect & Developer
- Certified TestStand Architect & Developer
- Certified LabVIEW Embedded Developer,
- Certified Professional Instructor
- 1 LabVIEW Champion

Des langages de programmation et de développement

- NI LabVIEW (RT, FPGA)
- NI TestStand,
- NI VeriStand
- NI LabVIEW FPGA, VHDL,
- Visual Basic, C#, .NET, C, C++, Python



Plus d'exemples et présentations techniques

<https://www.mesulog.fr/fr/metiers/realisations>

Mesulog

[ACCUEIL](#) [ENTREPRISE](#) [MÉTIERS](#) [PRODUITS](#) [EXPERTISE](#) [FORMATIONS](#) [ACTUALITÉS](#) [CONTACT](#)

INDUSTRIE

PDF

BANC DE CARACTERISATION VIS A BILLES

Pilotage d'un banc de mesure de l'erreur de pas d'une vis à billes. Acquisition synchronisée entre mesure angulaire (codeur) et mesure linéaire (interféromètre laser).

Télécharger

Prévisualiser

PDF

BANC DE CARACTERISATION VIS A BILLES

Solution utilisateur NI

Télécharger

Prévisualiser

PDF

BANC DE TEST RF POUR PRISE RJ45

Test en production des performances RF d'une prise RJ45. Pilotage d'AWR Microwave Office et d'un analyseur de réseau depuis LabVIEW.

Télécharger

Prévisualiser

PDF

BANC DE TEST CONVERTISSEURS HF

Test de convertisseur optique-électrique haute-fréquence de la gamme de produit D-Lightsys. Validation et caractérisation des produits en émission et en réception.

Télécharger

Prévisualiser

PDF

BANC DE TEST ACTIONNEUR

Test en production du sous-ensemble actionneur de la gamme de commutateurs RAMSES. Mesure, résistance bobine, rigidité diélectrique, force de contact, course.

Télécharger

Prévisualiser

PDF

BANC DE TEST COMMUTATEURS RF

Banc de test en production des produits finis de la gamme RAMSES. Gagnant du concours des meilleures applications 2008, dans la catégorie "Instrumentation/Test électronique" Test continu des caractéristiques des commutateurs RF (jusqu'à 32) lors d'un cycle en température.

Télécharger

Prévisualiser

PDF

BANC DE TEST COMMUTATEURS RF

Solution utilisateur NI

Télécharger

Prévisualiser

MÉTROLOGIE

PDF

BANC D'ETALONNAGE DEBITMETRIE

Logiciel d'étalonnage en débitmétrie liquide sur trois lignes simultanées. Banc référence nationale avec accréditation COFRAC. Liaison automate. Rapport de test évolutif.

Télécharger

Prévisualiser

PDF

BANC D'ETALONNAGE THERMOMETRIE

Étalonnage de 40 sondes de température en simultané. Mesures de précision métrologique avec accréditation COFRAC. Génération automatique du rapport d'étalonnage.

Télécharger

Prévisualiser

EXPERTISE

FORMATIONS

PRÉSENTATIONS TECHNIQUES

VIDÉOS LUGE

VIDÉOS VASCO

LIENS

BLOG

MÉTIERS

PRODUITS

EXPE

RÉALISATIONS

COMPÉTENCES

PRESTATIONS

SYSTÈME DE MESURE

EXPERTISE LABVIEW

EXPERTISE TESTSTAND

Nerys

Mesulog

EMERSON

Nerys Group companies

7

NI Days 2026 : Retrouvez-nous au stand 19

Défiiez l'IA. Parlons cybersécurité



Venez nous voir

Pour récupérer votre souvenir NI Days Paris 2026 – 40 de LabVIEW – 50 de NI

Pour jouer à notre jeu : Défiiez l'IA.

Pour échanger autour :

- Des architectures logicielles de mesure autour NI LabVIEW
- Des bancs de test automatisés autour de NI TestStand
- De la cybersécurité des systèmes de test
- De l'intégration concrète de l'IA dans les outils d'ingénierie.
- De l'analyse des données de tests

Dans un contexte de transformation industrielle, nos architectes vous accompagnent également sur :

- Migration Windows 10 → Windows 11
- Mise en conformité Cyber Resilience Act (CRA)
- Modernisation de bancs existants


Nerys Group companies

HUMAIN VS IA

SEREZ-VOUS PLUS RAPIDE QUE L'INTELLIGENCE ARTIFICIELLE ?



Une règle simple

- 1 Choisissez 2 dés
- 2 Lancez les dés
- 3 Annoncez la somme sur la tablette
- 4 L'IA donne son résultat

Le vainqueur est celui qui donne la somme en premier !

Astuce

Utilisez le dé "NI Days" pour simplifier votre calcul et prendre l'avantage !



Résultat Humain

Temps : **0.0s**
Somme : **0**

Résultat IA

Temps : **0.0s**
Somme : **0**

En attente du lancer



Choisissez 2 dés et lancez-les quand vous êtes prêt. Que le meilleur gagne !

Statistiques visiteurs

Victoires humaines	Participants	Victoires IA
0 %	 0	0 %

Sources

Merci :

- LabVIEW Debugging Tips and Tricks 2026 par Elijah Kerry, pour les Gifs animés sur les nouveautés LabVIEW et illustration outils NI
- Mes collègues, Mes clients, Les forums d'aide pour la source d'inspiration

Qu'est-ce qu'un bug ?

C'est un écart entre la réalité « constatée » et :

- ce que le client attend,
- ce que le cahier des charges décrit,
- ce que le développeur pense avoir codé,

Vous avez déjà entendu les anecdotes suivantes - lors des retours clients :

- “Chez moi ça marche.”
- “C'est normal, c'est codé comme ça.”
- “C'est bon j'ai corrigé”

Ces phrases **déplacent le problème** vers une **explication instinctive** au lieu de **créer une démarche d'analyse méthodique**.

Elles vont conduire à un correctif, mais pas toujours à la solution

Nous défendons une idée simple : passer de la réaction instinctive à l'enquête méthodique

« Mettre au point, ce n'est pas corriger à l'instinct, c'est appliquer une démarche méthodique pour comprendre pourquoi un système ne se comporte pas comme prévu »

Luc DESRUELLE, MESULOG

La démarche : prévenir → observer → analyser → comprendre → corriger → capitaliser.

Debugging LabVIEW Applications

Sommaire

1. La meilleure mise au point, c'est de l'éviter : coder dans les règles de l'art
2. Quand le bug arrive : Une stratégie pour mener l'enquête
3. Analyser les fichiers traces et journaux : de l'analyse manuelle à l'IA
4. Mettre au point dans LabVIEW 2026 : nouveautés de l'EDI
5. La spécificité des fuites mémoire
6. Aller plus loin avec les outils professionnels NI (logiciel & matériel)

Un planning chargé!..... 😊

La meilleure mise au point, c'est de l'éviter

Écrire un code source dans les « règles de l'art »

Est-il possible de faire un code maintenable sans méthode ?

A l'instinct ?

- Sans méthode
- Sans outils
- Sans règle

Evidement non

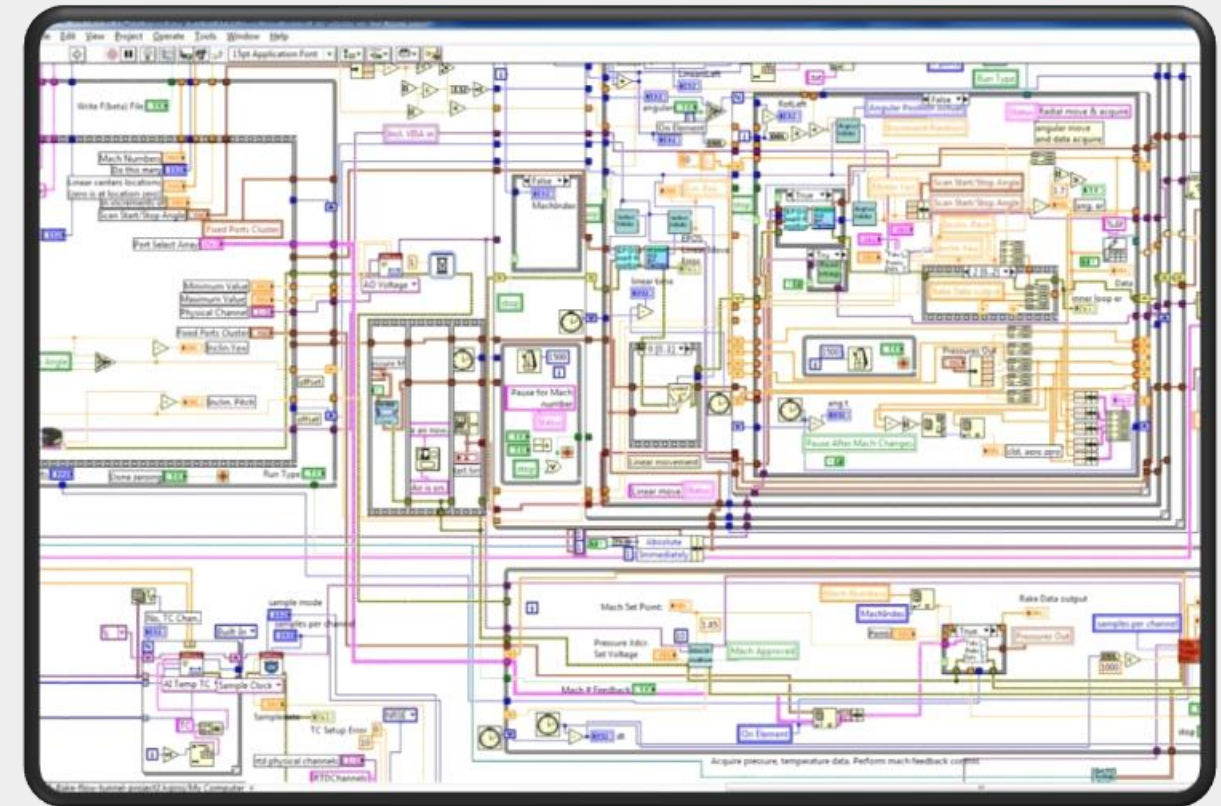
Comme cette image générée par mon IA préférée



La réalité constatée

L'absence de méthode est à l'origine des bugs

- Code complexe
- Non documenté
- Non maintenable
- Imprévisible
- Pas de mise au point possible
- Problèmes de performance
- Problème de fuite mémoire
- ...



Méthodes et outils NI pour la gestion d'un projet logiciel

Etapes du développement



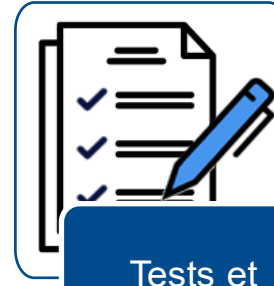
Spécifications



Architecture d'application



Développement



Tests et validation



Déploiement et correctifs

- NI Package Manager
- Failles de Sécurités
 - CVE
 - CWE

- Matrice des exigences
- Requirements Gateway

- Framework (DQMH...)
- Certifiés CLA + CLED

- SAST : **VI Analyser Toolkit**
- DAST : **Unit Test Framework**
- CI : Intégration continue

- Sécurité des produits numériques
- CRA : Cyber Resilience Act
- Exigences minimales cybersécurité tout au long du cycle de vie

- Les 10 commandements LabVIEW
- **LabVIEW Style Checklist**
- Techniques de développement
- Livres sur NI LabVIEW de Luc ☺
- Développeurs Certifiés CLD + CLAD

Les catégories d'erreurs qui conduisent aux bugs

2 types d'erreurs

La qualité du code source
(la forme)



L'architecture du projet
(le fond)



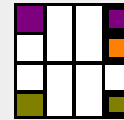
Les 10 commandements : Les bases de la qualité



1. Sur un seul écran ton code tiendra
2. De gauche à droite tu coderas
3. Tes fils tu ne croiseras pas
4. La gestion d'erreurs tu ne transgresseras pas
5. Le connecteur de type 4x2x2x4 toujours tu utiliseras
6. Tes objets judicieusement tu nommeras
7. Le flux de données tu respecteras
8. Ton code tu documenteras
9. Des sous-VI tu feras
10. La lisibilité du code tu garantiras

Exemples de base : style et documentation

- Connecteur 4x4 (Par défaut : 4x2x2x4)
- Icône « explicite »
- Documentation Aide Contextuelle *Ctrl + H*
- Câbler toutes les erreurs
- Dans les boucles = registre à décalage pour mémoriser l'erreur
- Dans une boucle avoir une condition d'arrêt si erreur
- Commentaires développeur – fils droit – Gestion erreur – gauche vers la droite

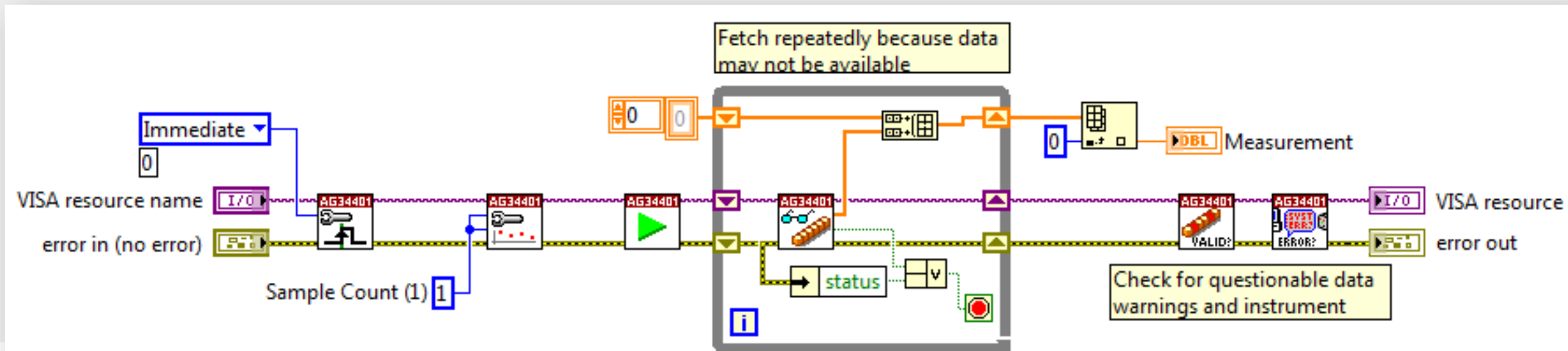


Aide contextuelle

C:\...ents\LabVIEW 2012\instr.lib\Agilent 34401\Public\Utility\Error Query.vi

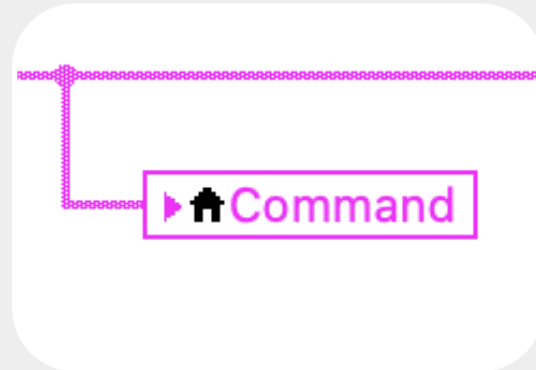
VISA resource name **AG34401** VISA resource name out
 error in (no error) **ERROR?** Error Code
 Error Message
 error out

This VI queries the instrument for any errors in the instrument's error queue. It will continue to read errors until all errors have been read from the instrument. Any errors will be returned in the Error Out cluster.

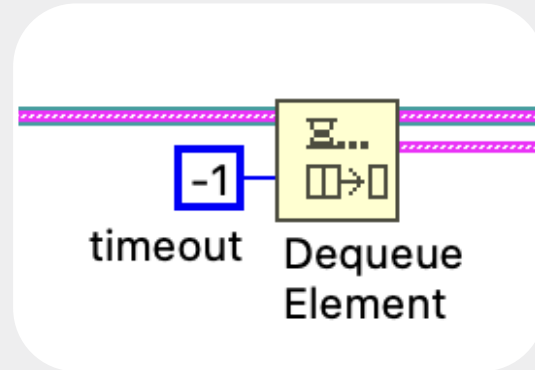


10 astuces pour éviter les bugs

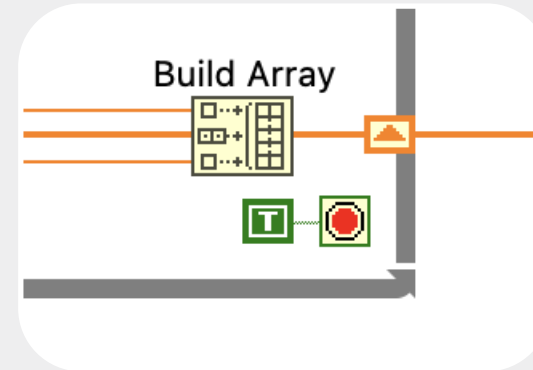
Inspirées des règles "Power of Ten" de la NASA et de JPL pour les logiciels critiques



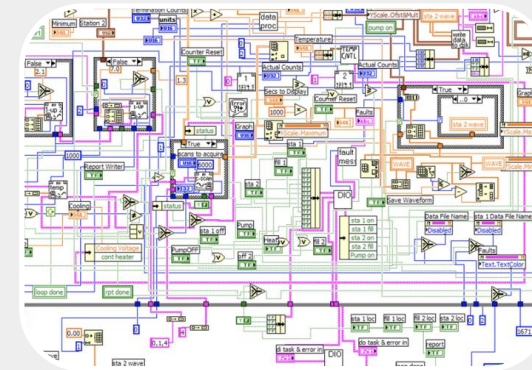
Éviter les locales et globales. Attention aux écritures en parallèle.



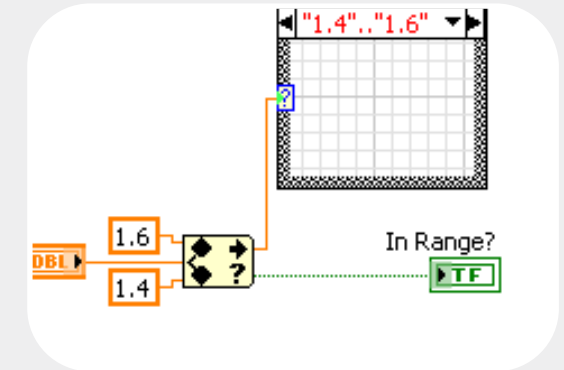
Définir des temps d'attentes finies et imposer un nombre maximal de tentatives.



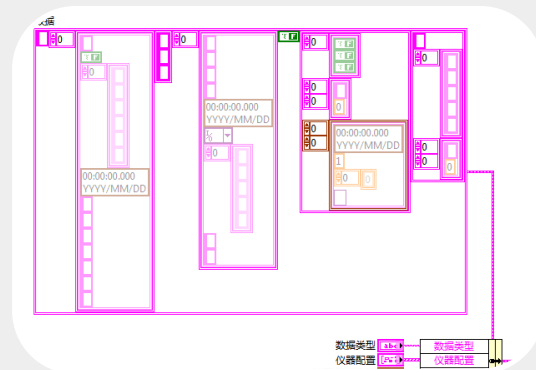
Pre-allouer les tableaux et éviter la construction de tableau dans les boucles.



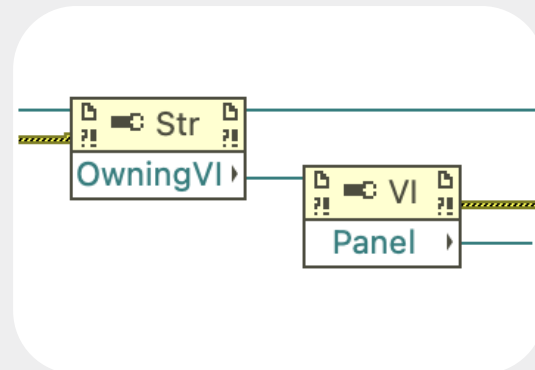
Le code doit être lisible en entier sur un écran.



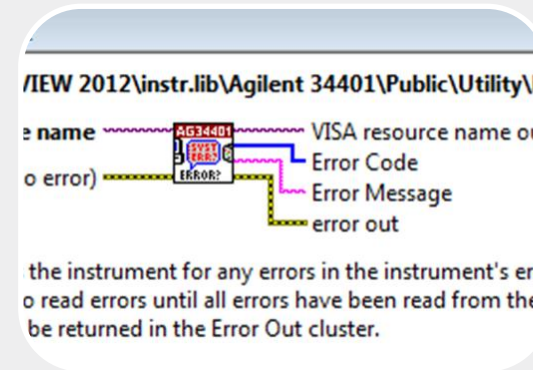
Rendre les conditions des structures explicites



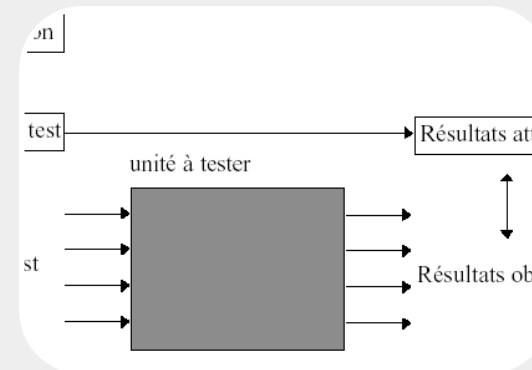
Limiter les structures de données trop complexes



Gérer toutes les erreurs. Ne jamais penser qu'une erreur ne peut pas se produire.



Documenter son code



Tester son code lors de l'écriture, Penser à toutes les possibilités

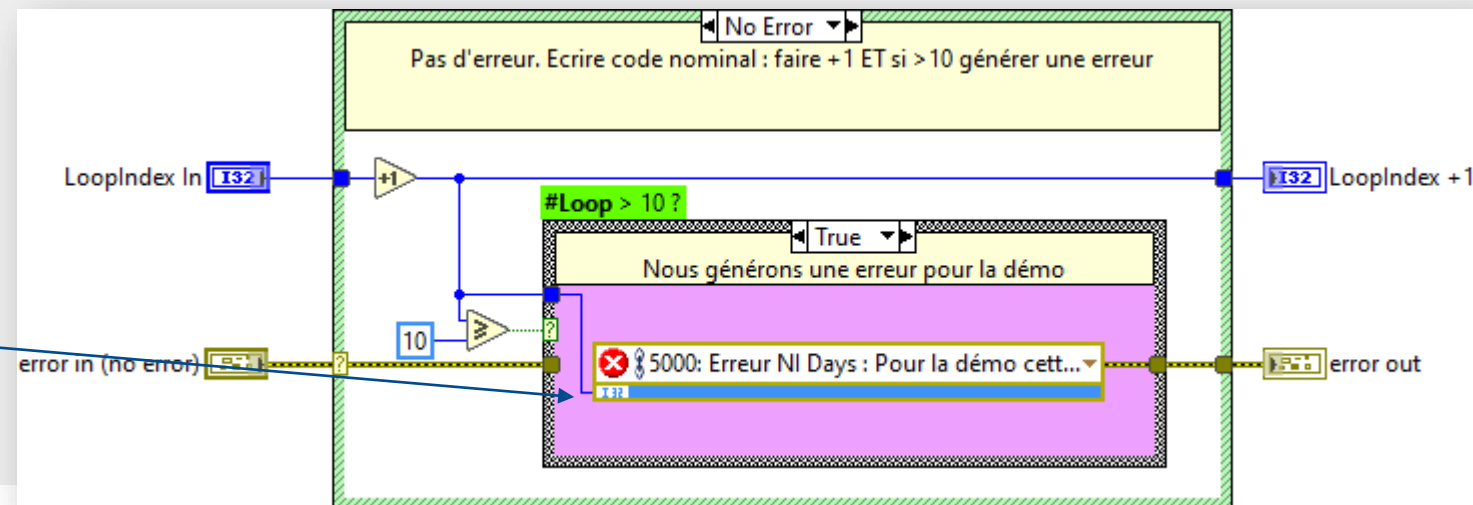
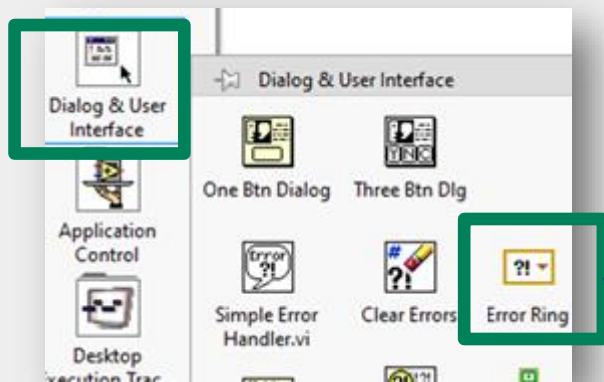
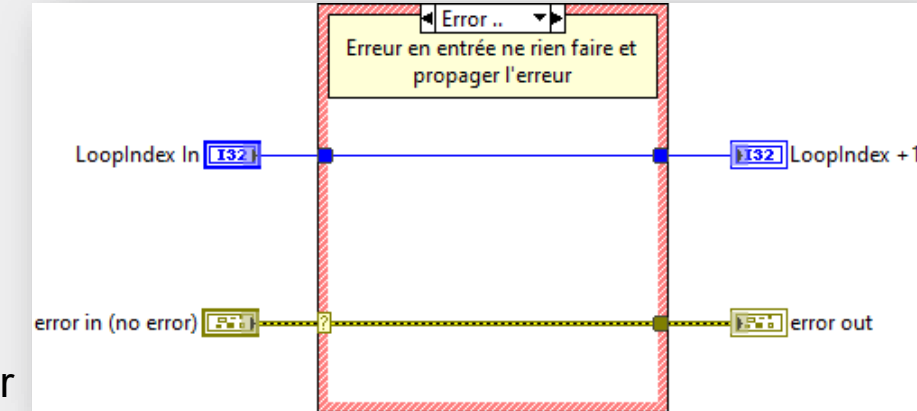


Lancer régulièrement le VI Analyzer.

Concevoir en pensant à la mise au point!

Les bases de la gestion d'erreur sous LabVIEW

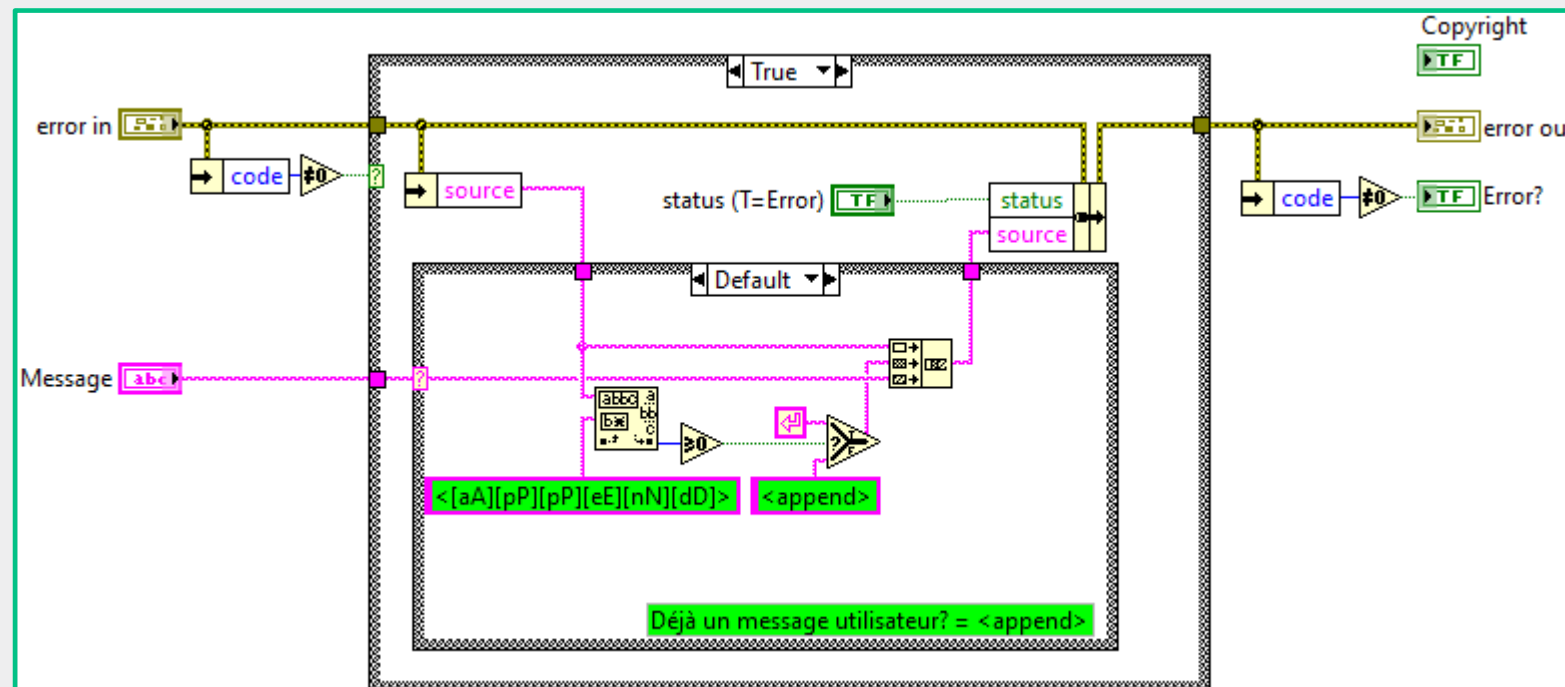
- Coder les fonctionnalités attendues et protégez-vous des comportements non désirés
- Chaque VI doit avoir - en paramètre d'entrée et de sortie - un cluster d'erreur,
- L'erreur d'entrée doit être câblée sur une structure condition afin de transmettre l'erreur sans la modifier
- Utiliser un registre à décalage dans les boucles pour ne pas perdre l'erreur
- Savoir générer une erreur utilisateur (code erreur > 5000, **message, chaîne d'appel et informations de diagnostic**)



Conseil : La gestion d'erreur pour les architectes

Ajouter des indices dans votre enquête : personnaliser les messages d'erreur

- En cas d'erreur, **penser à ajouter un message utilisateur dans le message d'erreur** afin de faciliter la mise au point.
- Ajouter une balise **<append>** dans la « source » du cluster d'erreur
- En faire un VI facilement utilisable



Exemple avec un fichier qui n'existe pas

Message

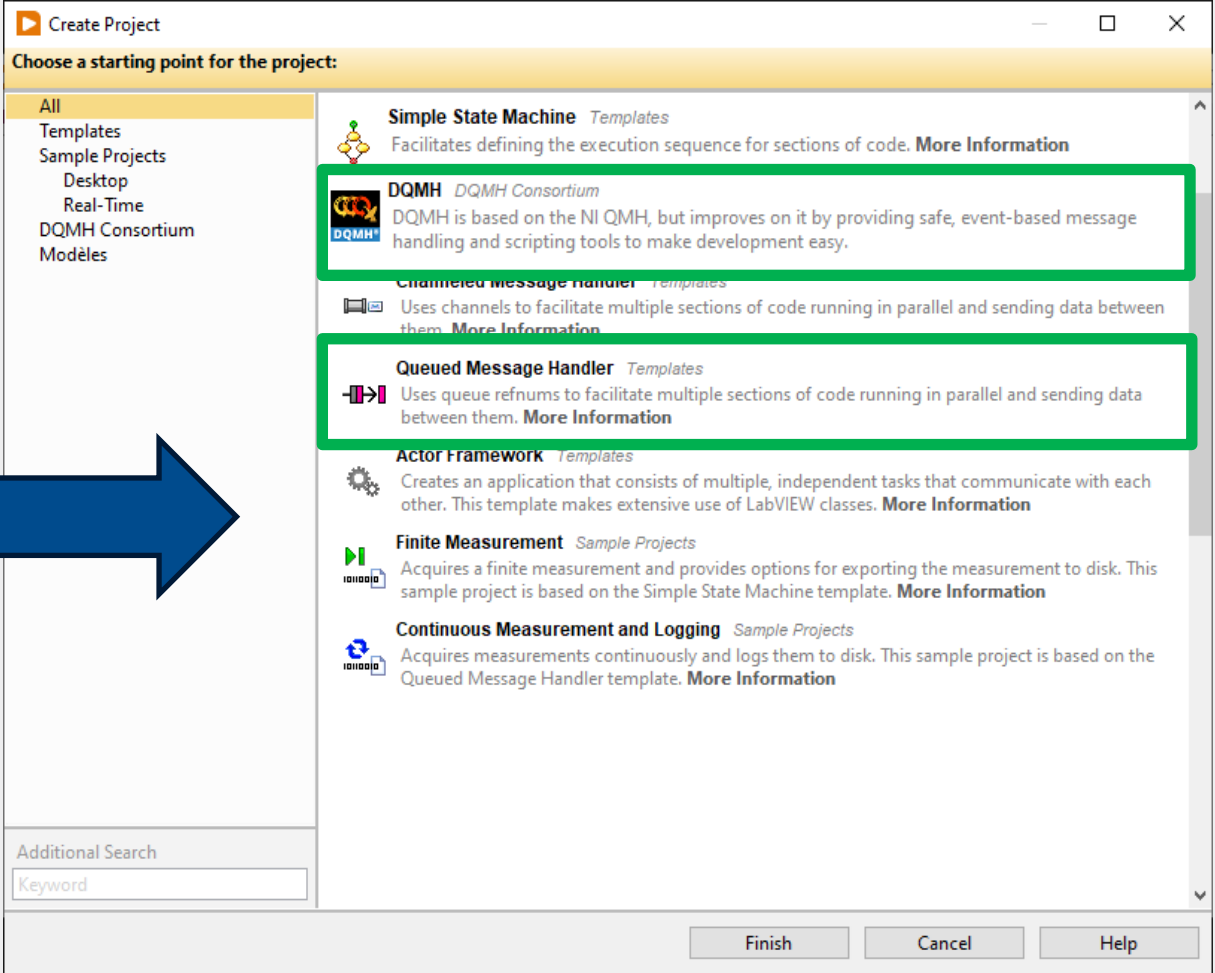
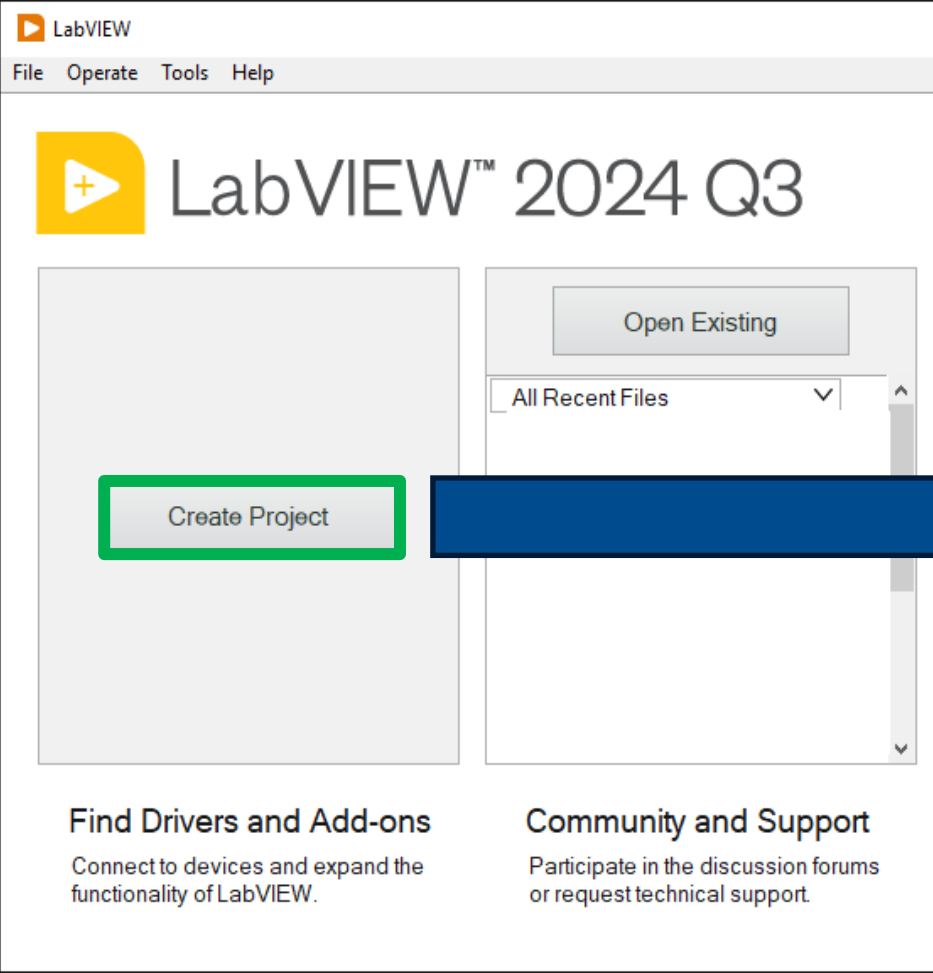
Le fichier c:\NIDays.log n'existe pas

Explain Error

Error Cluster	Explanation
Status: Code: 7 Hex Code: 0x 7 Source: <append> Le fichier c:\NIDays.log n'existe pas	Error 7 occurred at an unidentified location Possible reason(s): LabVIEW: File not found. The file might be in a different location or deleted. Use the command prompt or the file explorer to verify that the path is correct. NI-488: Nonexistent GPIB interface. Le fichier c:\NIDays.log n'existe pas

Architecture projet : Démarrer avec un modèle de projet

Les incontournables à maîtriser



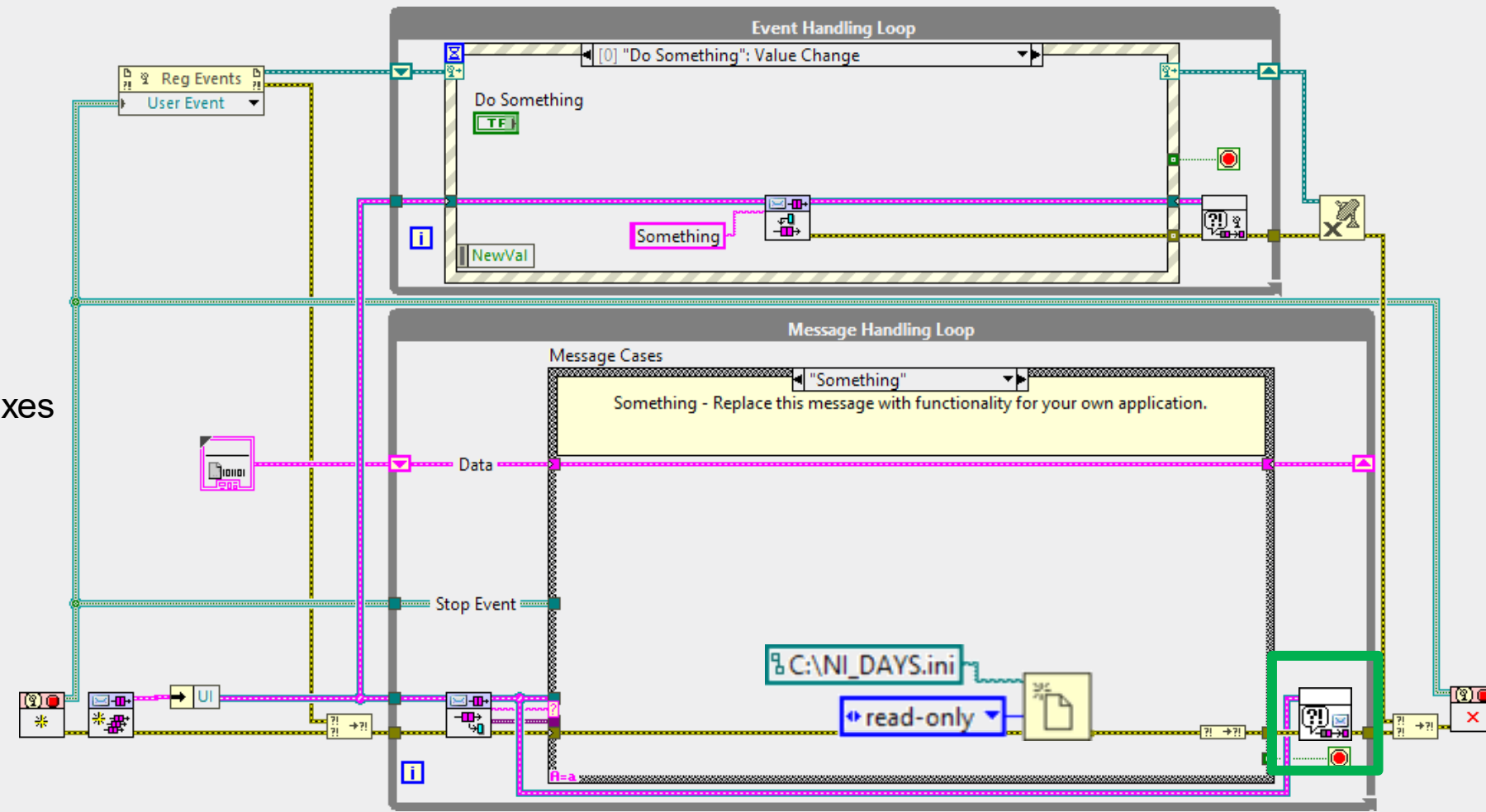
DQMH
Framework tiers largement adopté par la communauté

QMH
La base par NI

Exemple du QMH : le modèle de projet de NI

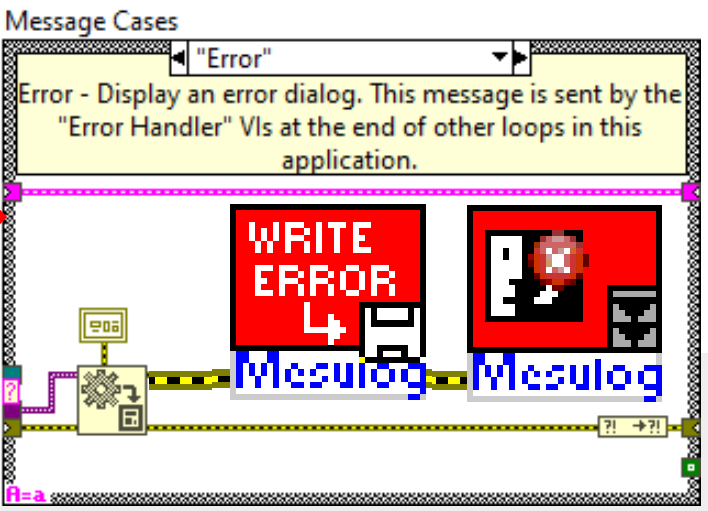
Modèle nativement proposé par NI dans LabVIEW

- **Utilisable pour les certifications**
- Documenté
- Grande souplesse d'évolution
- Facile à mettre au point
- Facile à maîtriser
- Brique de base pour construire des applications complexes
- Standard approuvé
- **Gestion d'erreur intégrée**

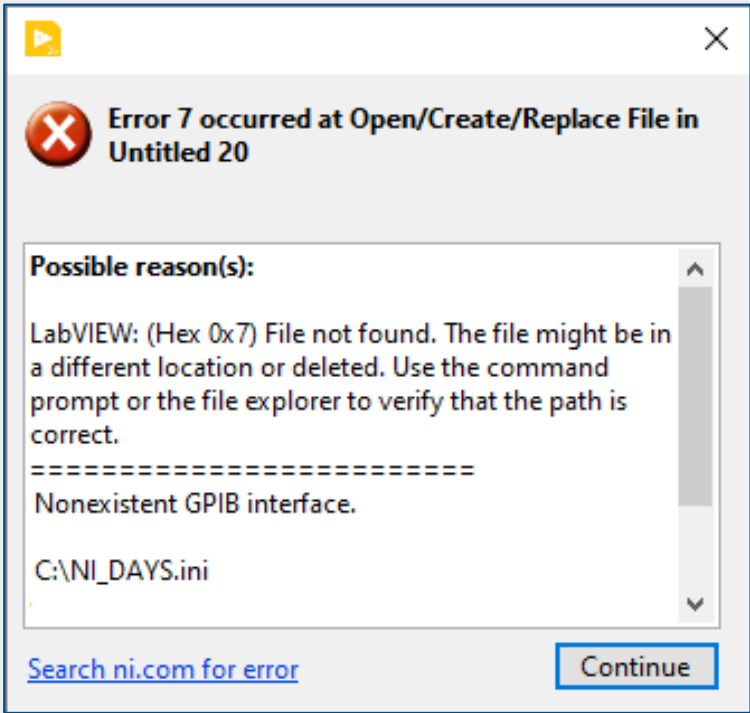


La gestion d'erreur dans le QMH

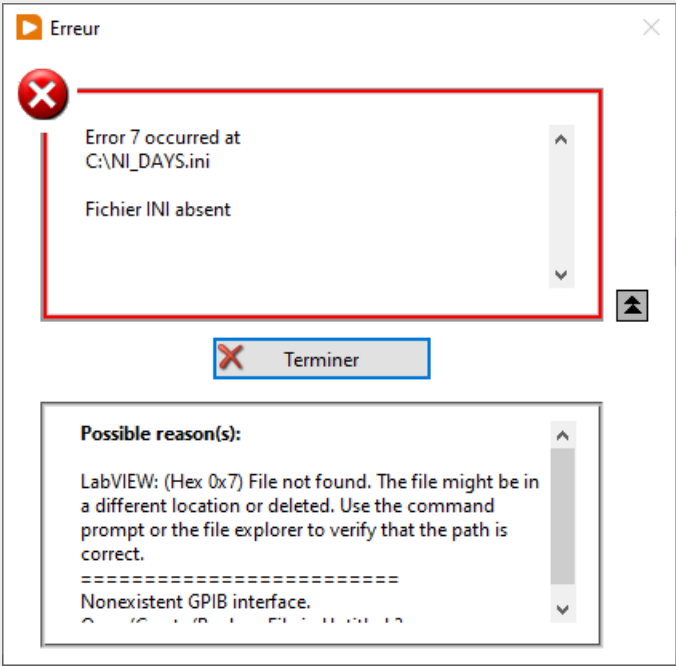
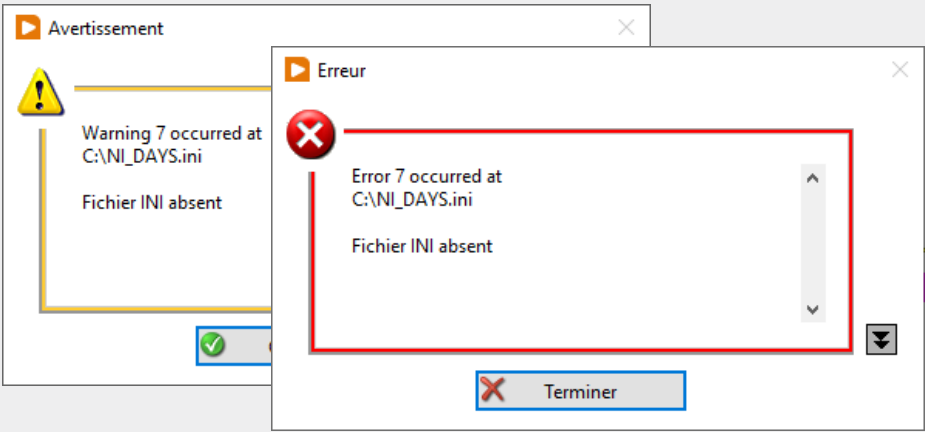
Toutes les erreurs sont envoyées dans l'état « ERROR »



Boite dialogue erreur standard



Boite de dialogue erreur personnalisée



Visualiser les informations dans l'application



Conseil : Ajouter un log d'erreur



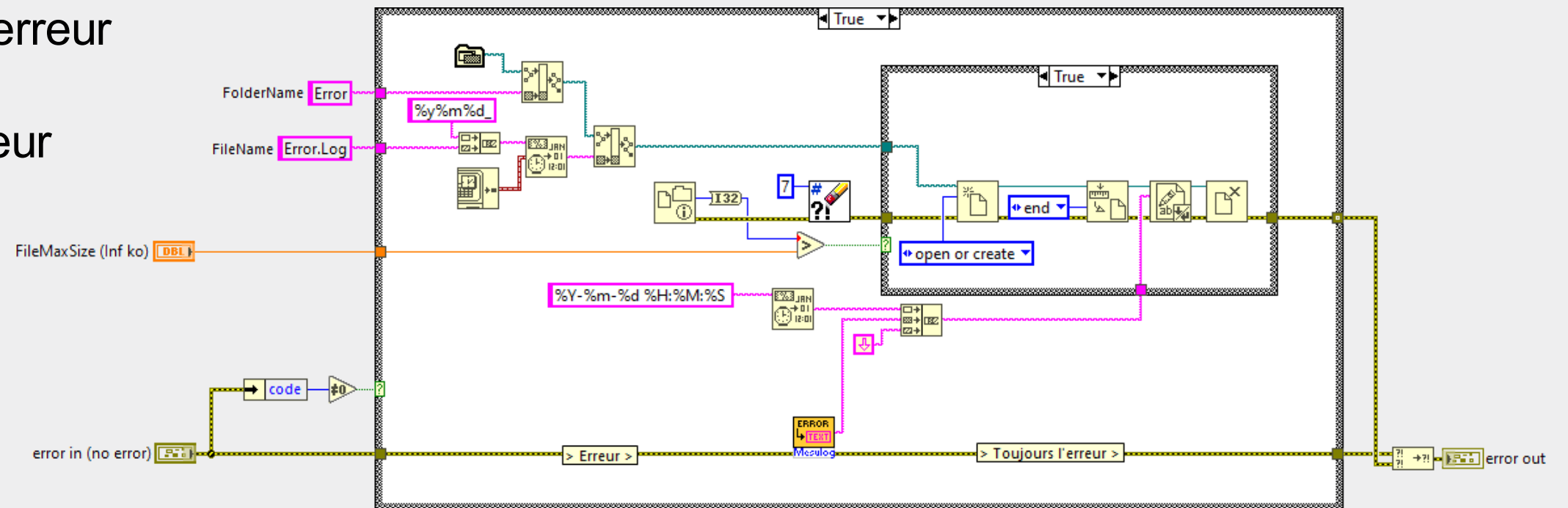
La sauvegarde des erreurs

Dans un fichier LOG

- ➔ Date de l'erreur
- ➔ Heure de l'erreur
- ➔ Localisation de l'erreur
- ➔ Code de l'erreur
- ➔ Contexte de l'erreur

260605_Error.log

```
260602_Error.log x |
1 2026-06-02 9:27:02
2 Warning occurred at Open/Create/Replace File in MAIN.VI/DEMO_NI_Days.vi
3 Possible reason(s):
4 LabVIEW: (Hex 0x7) File not found.
5 The file might be in a different location or deleted.
6 C:\NI_DAYS.ini Fichier INI absent
```



Conseil pour la mise au point : ajouter un fichier de diagnostic

Suivre des informations sur le logiciel

Dans un fichier Diag

- ➔ Date
- ➔ Heure
- ➔ Message
 - Action utilisateur
 - Démarrage logiciel
 - Arrêt logiciel
 - Taskill logiciel
 - ...

260602_Diag.log x

260605_Diag.log

1	2026-06-02 9:25:28	Start NI_DAYS_APPLICATION.EXE
2	2026-06-02 9:25:32	Version 1.0.0 RELEASE
3	2026-06-02 9:26:58	<UserAction> : Clic Button Start DEMO
4	2026-06-02 9:27:02	Error 8

Synthèse : La mise au point commence lors de l'écriture du code source

Pas seulement “bien coder”, mais avoir :

- **un guide et des règles développeur pour la qualité du code source**
- une architecture projet,
- des modes dégradés dans les VIs si erreur
- **une gestion d'erreurs systématique** (dans source et projet),
- **des journaux traces pour le diagnostic** « diag »
- **des journaux de sauvegarde des erreurs** « log »
- des tests unitaires ou scénarios de validation,
- une revue de code,
- ne pas attendre la fin du projet pour générer l'exécutable

Quand le bug arrive, mener l'enquête

Mettre au point, ce n'est pas "essayer des trucs", c'est une méthode d'analyse

Quand le bug arrive : mener l'enquête – une méthodologie

Mettre au point, ce n'est pas “essayer des trucs”. C'est une enquête méthodique.

- Ecouter les retours, reformuler, prendre des notes : l'audition des témoins
- Ne pas chercher une conclusion trop rapidement,
- Catégoriser le problème : en [source, exécutable, matériel], [systématique / aléatoire], IHM, crash, erreur de calcul ...
- Exploiter les fichiers [traces, log] + indicateurs IHM : pour recueillir les indices et orienter vers les causes possibles
- Isoler le code source qui génère le dysfonctionnement : l'audition du suspect
- Reproduire le problème,
- Formuler des hypothèses, les tester, et garder des preuves de vos tests
- Corriger votre code
- Valider la correction, la déployer
- Documenter

C'est trouver une explication.... Rationnelle!

« Quand vous avez éliminé l'impossible, ce qui reste, aussi improbable soit-il, doit être la vérité. »

Sir Arthur Conan Doyle - The Sign of the Four

le code était brisé et j'ai des erreurs 56 les 3/4 du temps c'est normal ?

Hello, vu : erreur 56 = alimentation qui ne génère pas assez de courant

Pour le code brisé, je ne sais pas ce que tu as reçu.

OK. Merci. Pour info le programme chargé n'était pas le bon.

Mise en garde dans cette enquête

Garder en mémoire que:

- Même de bonne fois, dans une enquête, 20 % des indices sont faux,
- Un bug est résolu lorsque l'explication trouvée :
 - explique tous les faits observés
 - explique les faux indices
- Les corrections hâtives sont des rustines : Elles masquent un symptôme sans corriger la cause
- Un message d'erreur peut être une conséquence, pas une cause.
- Si vous fermez les yeux, le dysfonctionnement reviendra un vendredi à 18h00

Analyser les indices, les fichiers traces et journaux

De l'analyse manuelle à l'IA

Collecter les indices : des plus simples aux plus complexes

Collecter et sauvegarder :

1. Depuis votre application

- Vos indicateurs dans l'IHM
- Vos fichiers d'erreur (fichier log)
- Vos journaux de diagnostic (fichier diag),

2. Depuis votre OS

- Les journaux des événements

3. Les crashes dumps et journaux : EDI LabVIEW et les autres applications

- %userprofile%\Documents\LabVIEW Data\LVInternalReports \NomApplication
- %localappdata%\National Instruments\ApplicationUtilise\Logs*.log
 - NIPB log file: %localappdata%\National Instruments\NI Package Builder\Logs*.log
 - NIPM log file: %localappdata%\National Instruments\NI Package Manager\Logs\NIPkgLogs*.log
 - NIPM error file: %localappdata%\National Instruments\NI Package Manager\Logs\errorlog.txt

Erreur 7 : Fichier INI absent

☐ Masquer les erreurs

Réinitialiser les erreurs

0

0

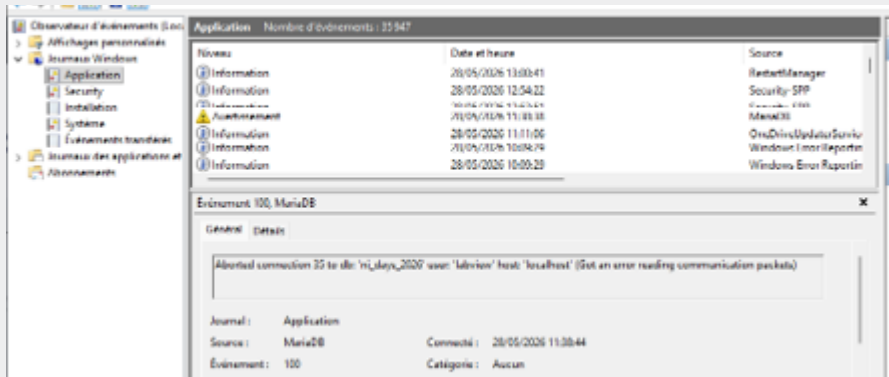
260602_Diag.log

```
1 2026-06-02 9:25:28 Start NI_DAYS_APPLICATION.EXE
2 2026-06-02 9:25:32 Version 1.0.0 RELEASE
3 2026-06-02 9:26:58 <UserAction> : Clic Button Start DEMO
4 2026-06-02 9:27:02 Error 8
```

260602_Error.log

```
1 2026-06-02 9:27:02
2 Warning occurred at Open/Create/Replace File in MAIN.VI/DEMO_NI_Days.vi
3 Possible reason(s):
4 LabVIEW: (Hex 0x7) File not found.
5 The file might be in a different location or deleted.
6 C:\NI_DAYS.ini Fichier INI absent
```

Observateur d'événements sous Windows



« Le programme s'arrête »

-> le PC a redémarré suite mise à jour Windows

Comment les analyser ? : humain versus IA

C'est une équipe : complémentaire - chacun couvre ce que l'autre ne sait pas faire



ANALYSE MANUELLE par humain

CE QU'ELLE APPORTE

- ✓ Connaît le contexte métier : ce que le code doit faire
- ✓ Expérience - Expertise : il connaît son métier
- ✓ Remonte à la cause racine, pas au seul symptôme

SON COÛT

- ⚠ Chronophage dès que le volume explose
- ⚠ Peu reproductible dépend du développeur



ANALYSE AUTOMATIQUE par IA Classique

CE QU'ELLE APPORTE

- ✓ Analyse en quelques secondes des volumes énormes
- ✓ Regroupe et corrèle les signaux faibles
- ✓ Traduit un code d'erreur cryptique en clair

SES LIMITES

- ✗ Pas de notion liée au métier, au matériel NI
- ✗ Confiante même quand elle se trompe

Analyse de vos indicateurs IHM et fichiers de « votre application » LabVIEW : L'analyse manuelle

Le développeur connaît son métier

- C'est lui qui a fait le code, il connaît le contexte métier des fichiers traces
- Lecture du message d'erreur doit orienter vers l'origine et donc la solution
- Vos fichiers d'erreur (fichier log) : nom du VI en erreur + code + raison possible = 95% des résolutions
- Analyse critique avec la possibilité d'ajouter des informations de diagnostic dans le code

Fichier trace d'une application tierce : Analyse par IA

Exemple erreur « Package Builder »

- Constat : Impossible de réaliser le Package
- Information :
 - error file: %localappdata%\National Instruments\NI Package Builder\Logs
 - Fichier log-2026-06-10_09-47-20.txt
- Analyse : fichier erreur par Claude
- Temps attente : 10 secondes
- Proposition : réinstaller NI TestStand
- Verdict : Validé

C'est un démarrage de **NI Package Builder 26.0** (64-bit).
Au lancement, le plugin **TestStand** échoue à se charger.
Le reste se passe bien. L'appli est donc fonctionnelle,
juste amputée du support **TestStand**.

La cause technique

L'erreur racine est un classique COM Windows :

0x80040154 - REGDB_E_CLASSNOTREG (« Classe non enregistrée »).

Package Builder essaie d'instancier le moteur **TestStand**, mais cette classe COM n'est pas enregistrée sur la machine.

Traduit : **TestStand** n'est pas installé ou son enregistrement COM est cassé.

Pistes de résolution (par probabilité)

- **TestStand** n'est pas installé sur cette machine.
- **TestStand** installé en 32-bit alors que **Package Builder** tourne en 64-bit.
- Enregistrement COM corrompu → une réparation de l'install **TestStand** remettrait le CLSID en place.

Fichier crash de mon exécutable : Analyse par IA

Exemple crash application LabVIEW

- Constat : crash Vasco.exe
- Information
 - %USERPROFILE%\Documents\LabVIEW Data \LVInternalReports\Vasco\
- Analyse : 2 fichiers par Claude
- Temps attente : 30 secondes
- Proposition : ajouter classe dans build
- Verdict : Validé

Fichier Ivlog

Ce que dit le log

C'est un crash log de ton appli LabVIEW **Vasco**

Le cœur du problème

La stack d'appel du VI :

Get LV Class Default Value.vi → CONF_CreateDynamicObject.vi →
GetAllEnregTypes.vi → LectConf_v2.vi → Main.vi

Traduit : au démarrage, ton **Main.vi** lit une configuration (**LectConf**) et instancie dynamiquement des objets par type (**CreateDynamicObject**), via **Get LV Class Default Value** à partir d'un nom de classe.

L'interprétation technique

C'est typiquement lié à du chargement dynamique de classes (Factory pattern, plugins) où toutes les classes ne sont pas dans la dépendance statique du build.

Pistes (par probabilité)

- Classes chargées dynamiquement non incluses correctement dans le **build/AppLib**.

Fichier dump (dmp)

Ce que le dump confirme

Toute la stack du log tombe à l'intérieur de **lvrt.dll** — cohérent avec le log.
Le warning naît dans le moteur LabVIEW, déclenché par ton chargement dynamique de classe par **Get LV Class Default Value.vi**.

Ce que le dump ajoute

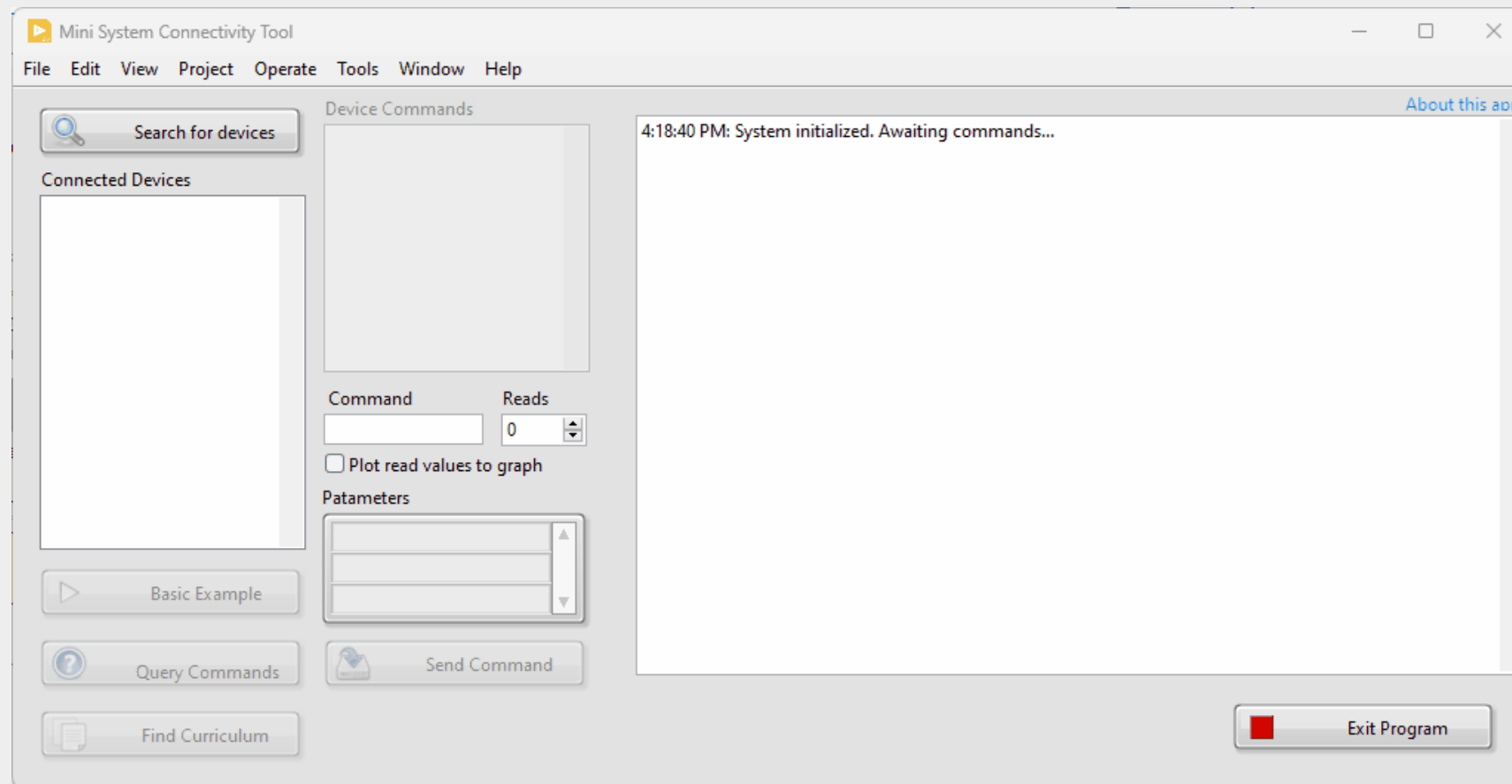
- Aucun **exception stream**.
- L'incident est **purement dans la couche objet/config**, pas dans une I/O matérielle.

Le diagnostic affiné

Au démarrage, **Main.vi** → **LectConf.vi** lit une config et instancie dynamiquement des classes via **Get LV Class Default Value**.
les classes sont **absentes de la spécification de construction**.

Nigel l'IA spécialisée de NI : exemple de mise au point avancée

Nous interrogeons des instruments, et certaines requêtes ne fonctionnent pas



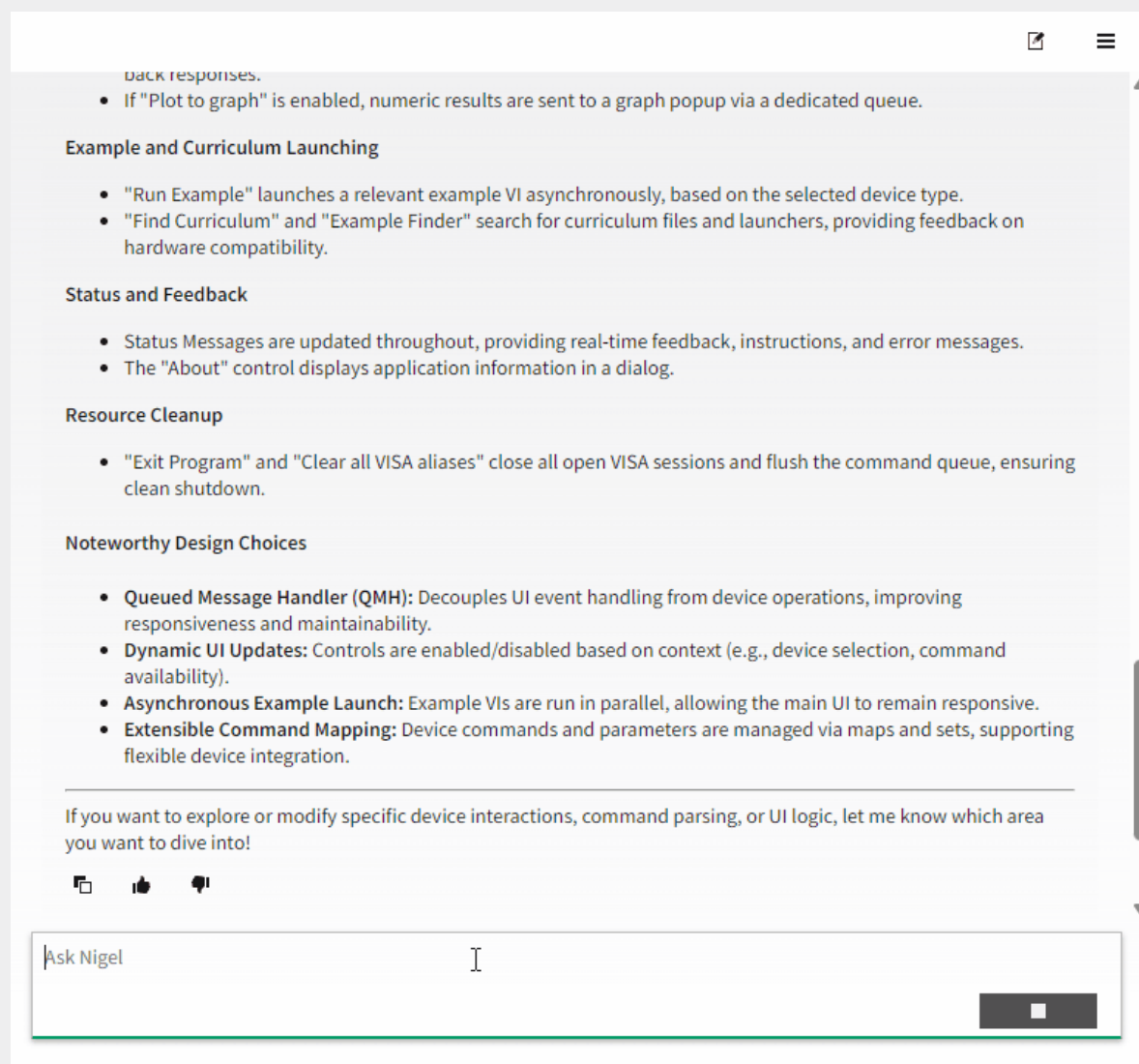
Les attentes

- Sur requête, l'appareil devrait retourner une liste de commandes valides.
- Les requêtes devraient retourner une valeur.
- Un champ vide n'est pas une réponse valide.

La pratique

- Certaines commandes fonctionnent
- D'autres ne semblent rien faire
- Il n'y a pas d'erreur constatée

Utiliser Nigel pour décrire le problème



Où implémenter

- Dans le case « Retrieve Commands » (boucle contrôleur), après avoir analysé la réponse du périphérique et avant de l'insérer dans l'ensemble (set) ou de mettre à jour la listbox.
- Optionnellement, filtrer également dans le case « Display Commands » avant d'écrire dans la propriété Item Names de la listbox.

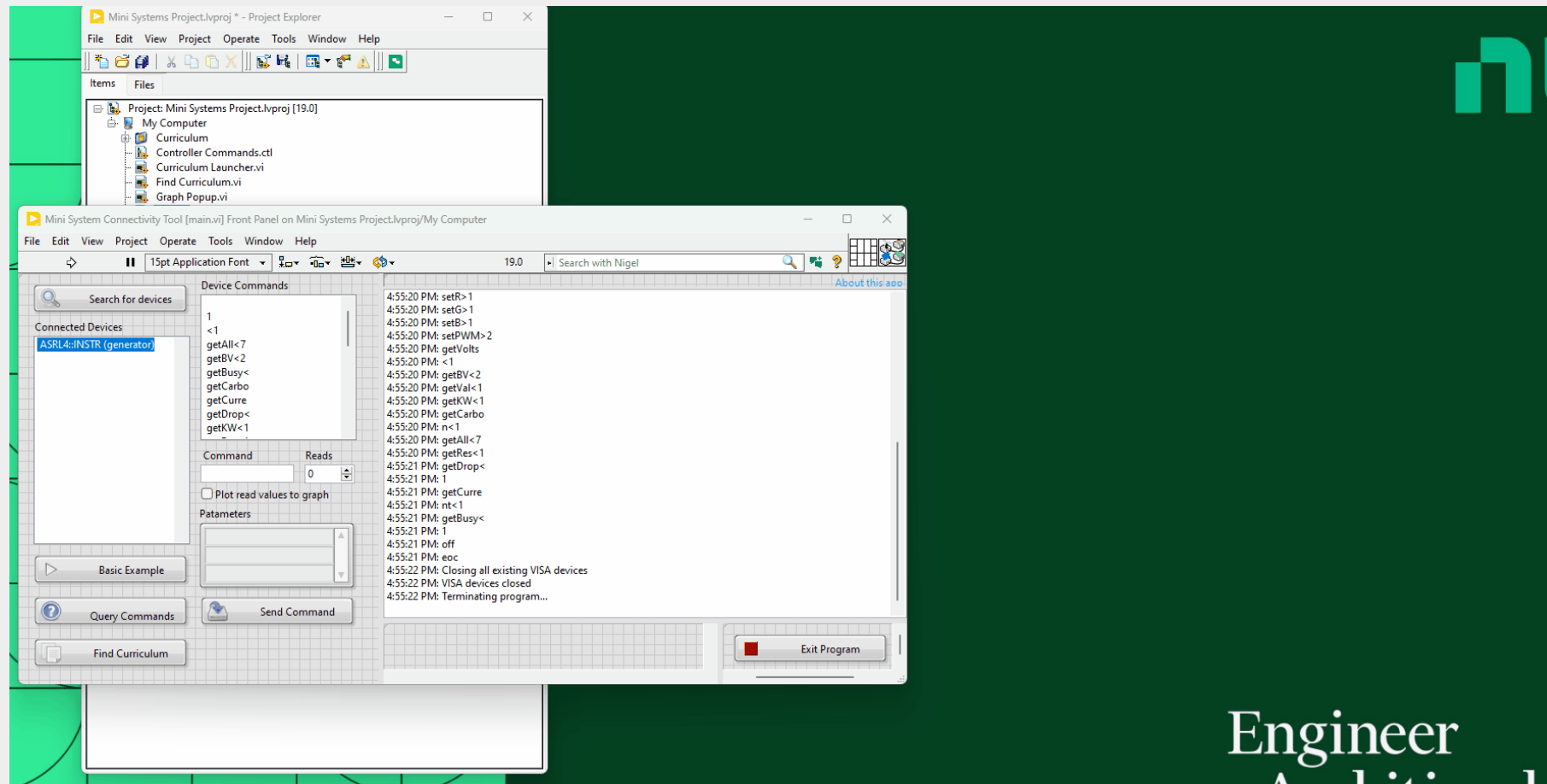
Erreurs courantes à éviter

- Ne pas supprimer les espaces (whitespace) avant de vérifier si une chaîne est vide.
- Ne pas gérer les cas où le périphérique renvoie un format inattendu (par ex. délimiteurs supplémentaires, retours à la ligne).
- Ne pas mettre à jour de manière cohérente à la fois l'ensemble (set) et l'interface utilisateur (UI).

Étapes suivantes

1. Ajouter une logique de filtrage après l'analyse de la chaîne de commande et avant l'insertion dans l'ensemble (set).
2. Optionnellement, ajouter une validation par rapport à une liste connue de commandes supportées.
3. Tester avec des périphériques qui renvoient des délimiteurs de fin ou des listes de commandes mal formées afin de vérifier que les entrées vides ou invalides ne sont pas affichées.

Utiliser Nigel + NI I/O Trace pour comprendre l'erreur



- I/O Trace confirme que l'application reçoit bien les commandes de l'appareil.
- I/O Trace montre que certains appels ont pour statut le code 0x3FFF0006.
- Nigel explique une raison possible et indique la localisation dans le code source.
- Nigel recommande de changer la taille du buffer de lecture à 250 octets

IA et mise au point : révolution ou illusion ?

Ce que les IA – et notamment Nigel - savent bien faire

- Analyser les fichiers traces, dump
- Expliquer ou générer du code
- Interpréter les erreurs
- Documenter

Ce que l'IA fait mal

- Comprendre votre contexte industriel
- Comprendre le timing réel
- Comprendre l'architecture globale d'un projet complexe et les interactions

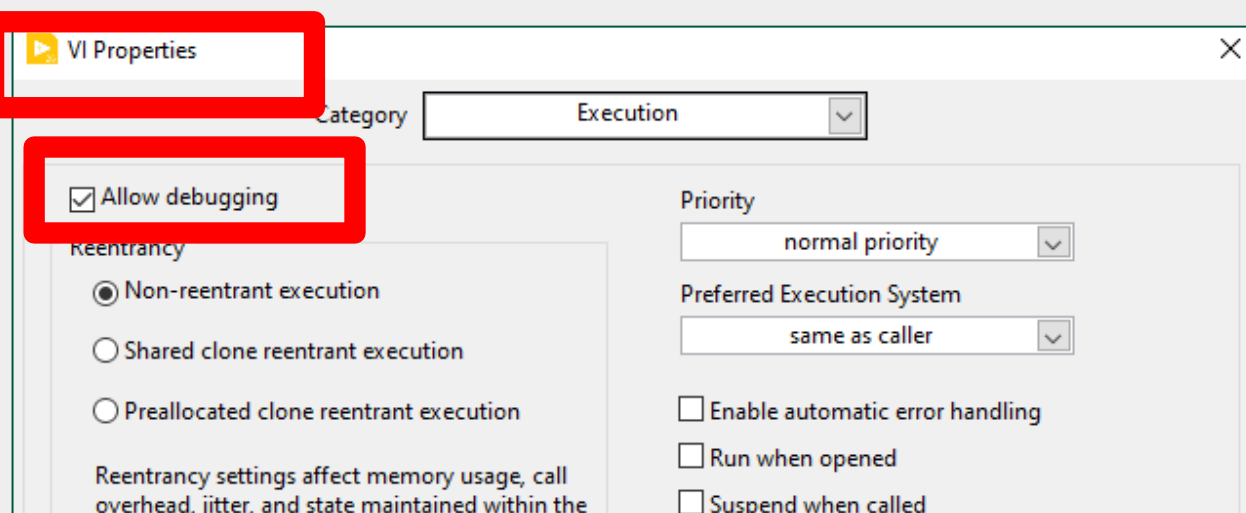
Outils NI pour isoler des problèmes de code source dans NI LabVIEW

Les méthodes et outils dédiés au code source dans NI LabVIEW

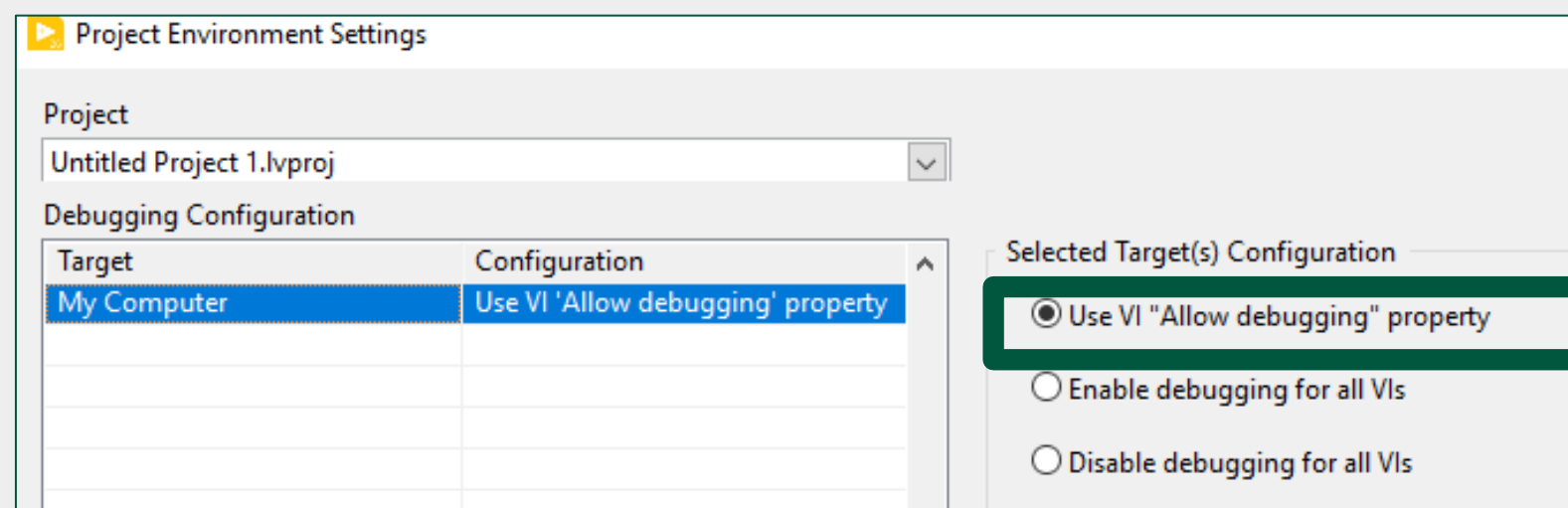
1) Activation du débogage pour un VI ou tous les VIs d'un projet

Permet - d'activer / désactiver - les outils de mise au point dans l'EDI NI LabVIEW

Soit chaque VI

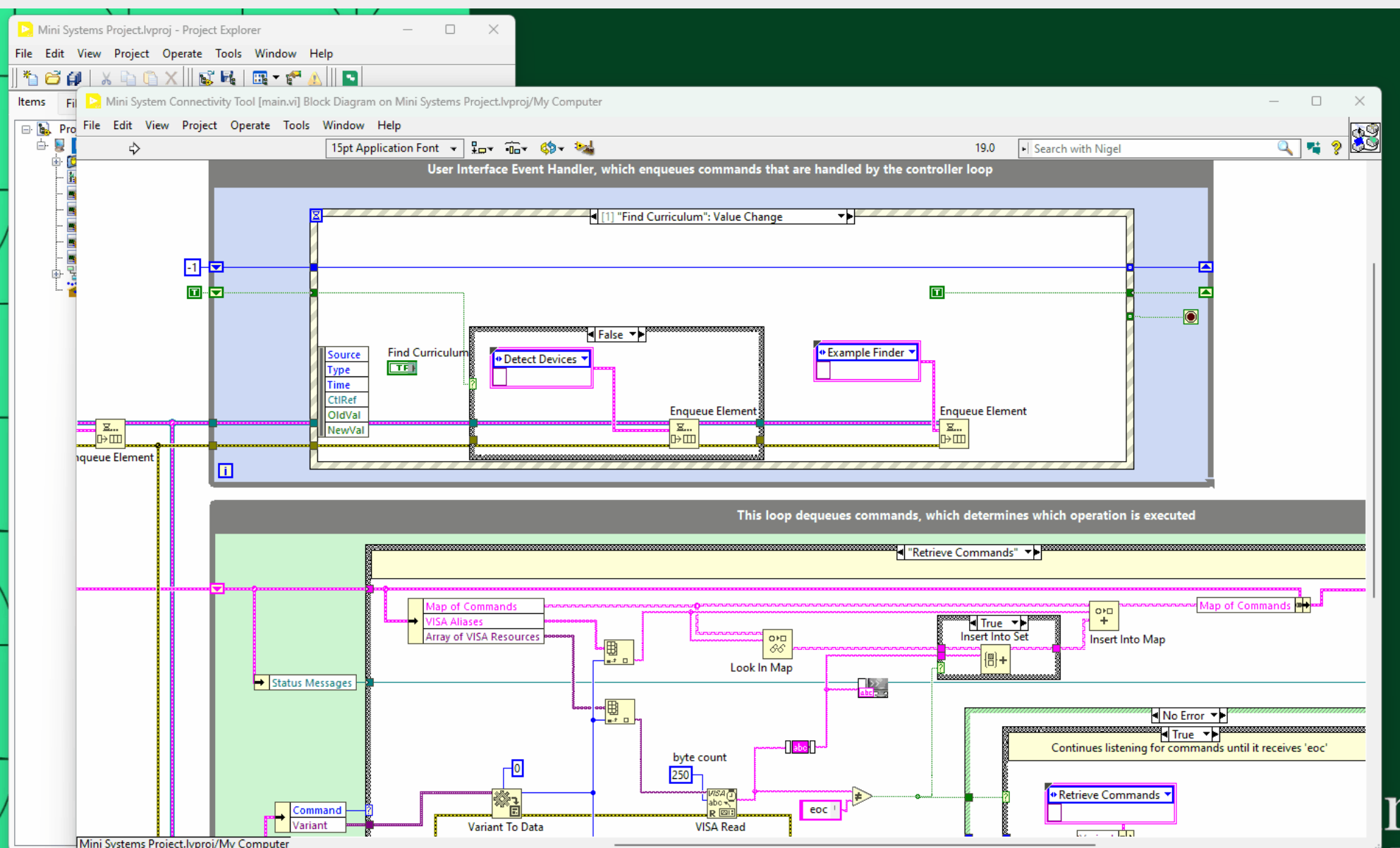


Soit pour l'ensemble des VIs du projet



Activation du débogage pour tous les VIs d'un projet LabVIEW

Menu : **Project > Project Environment Settings** et option « Activer débogage pour tous les VIs ».



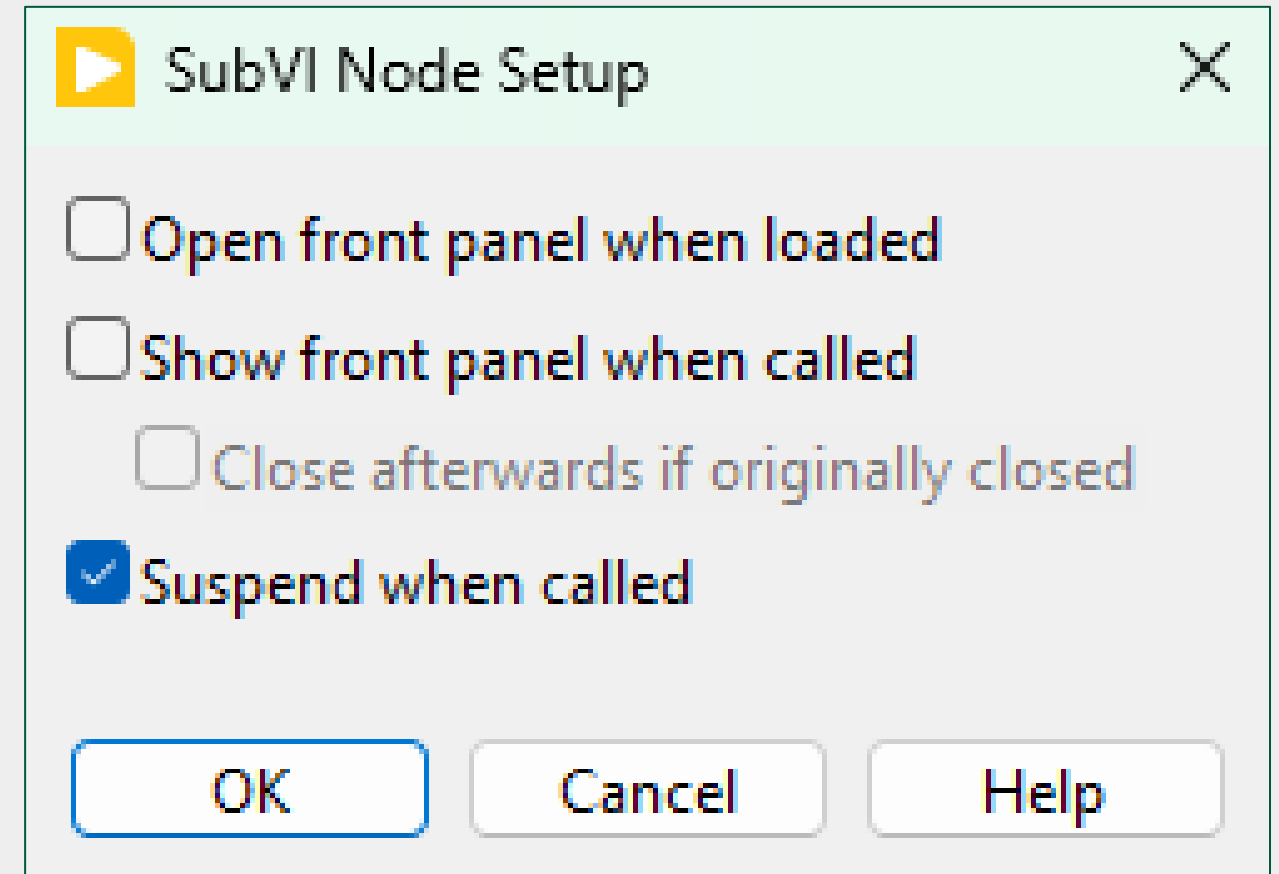
Pourquoi ?

Vous n'avez pas à gérer des sous-VIs dont l'option de débogage est désactivée et donc d'avoir à recommencer toute votre session de mise au point.

2) Suspendre un VI lors de son appel... et modifier les données

Clic droit sur le VI -> SubVI Node Setup.

Sélectionner "Suspend when called"



- La face-avant du VI s'affiche lors de son exécution
- Vous pouvez **modifier les valeurs des commandes et des indicateurs de la face-avant**
- Avant de cliquer sur le bouton pour poursuivre l'exécution avec les valeurs saisies manuellement
- Utile pour modifier manuellement les données d'un appareil absent ou un système simulé

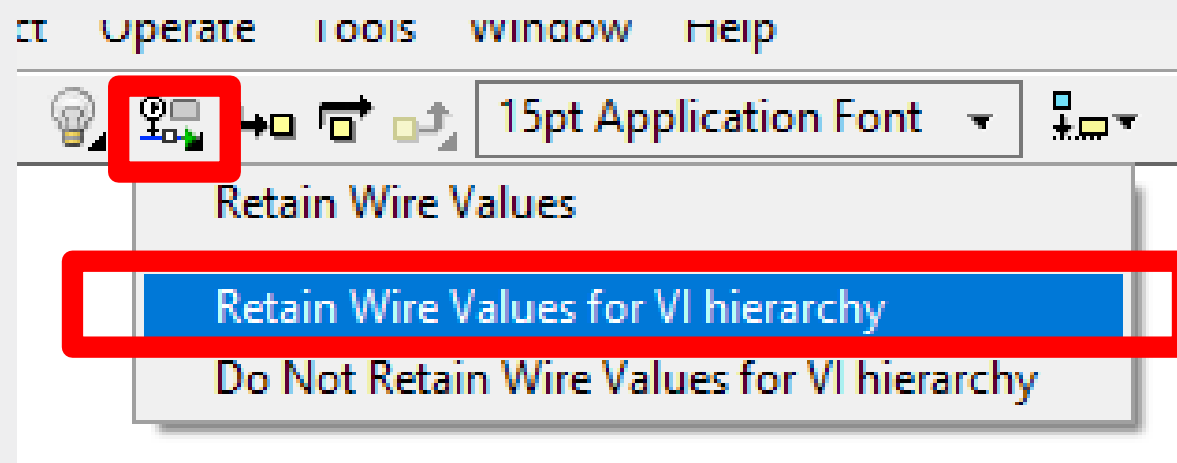
3) Inspecter les valeurs « des fils de données » après l'exécution du code

Activer 'Retain Values' depuis la barre d'outils d'un VI :



sert à **conserver la dernière valeur** entre deux exécutions du VI.

Pour Activer l'option pour toute la hiérarchie des VIs -> maintenir le clic souris sur le bouton menu

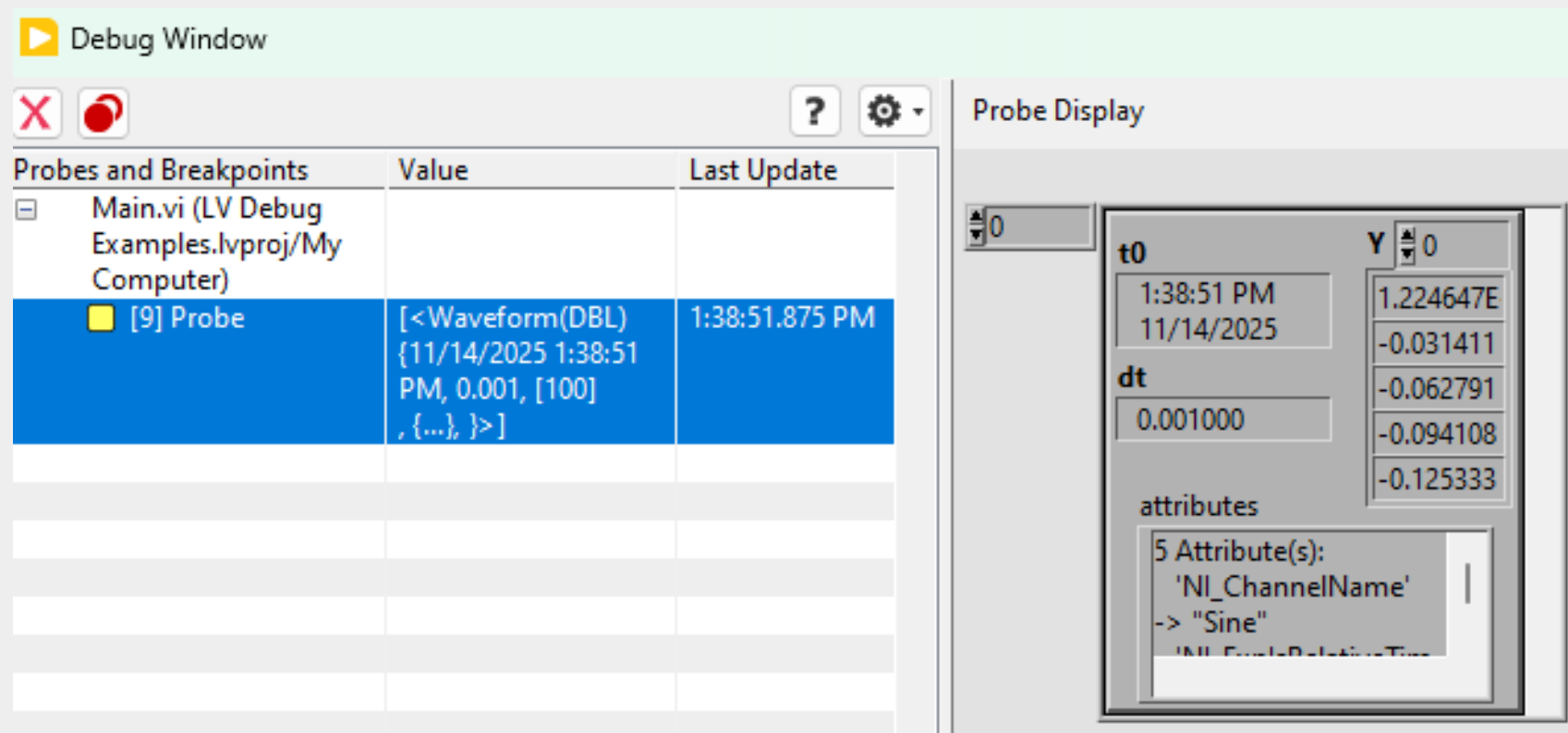


Pourquoi ? Cela vous permettra de placer des sonder les câbles après une exécution dans tous les sous-VIs.

Cela doit être utilisé avec prudence sur tous les VIs allouant une quantité importante de mémoire à l'exécution.

4) Fenêtre de débogage

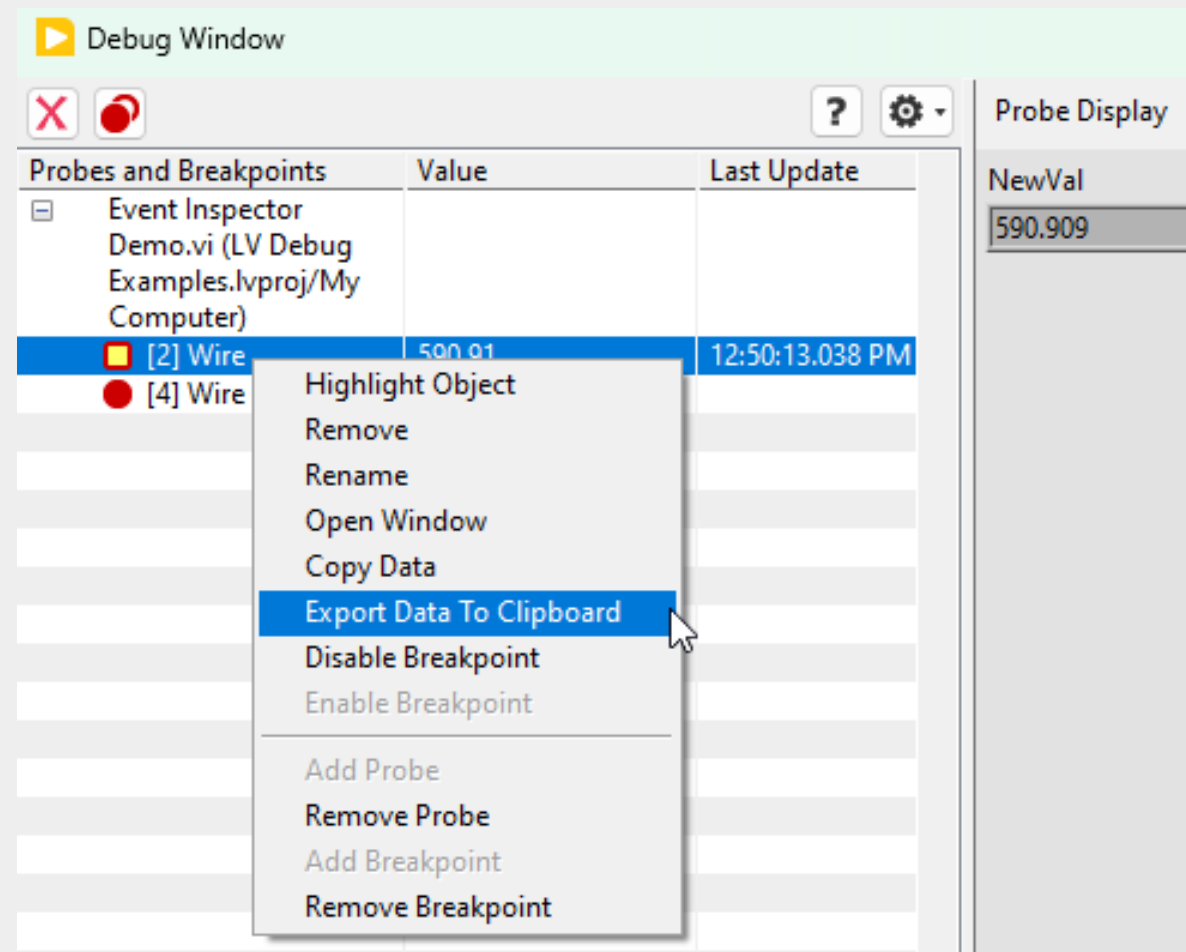
Les sondes et points d'arrêt sont gérés depuis une fenêtre commune



- Permet d'ajouter/supprimer des sondes et des points d'arrêt.
- Les sondes et points d'arrêt sont conservés d'une session à une autre.

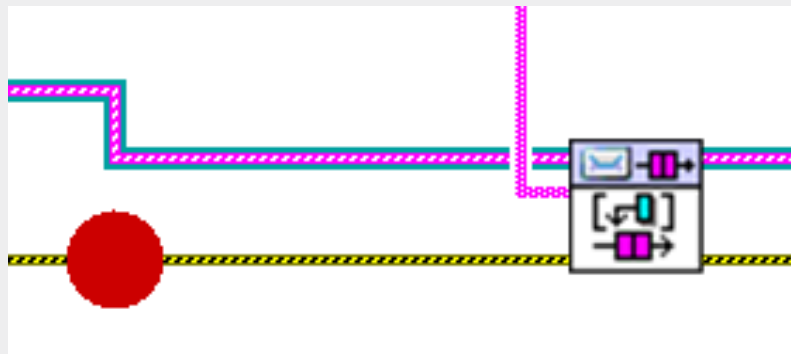
Fenêtre de débogage

Nouvelle fonctionnalité pour exporter des données dans le presse-papier

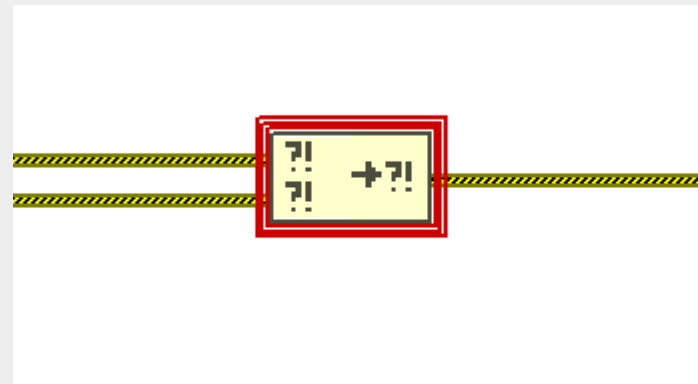


5) Les points d'arrêt

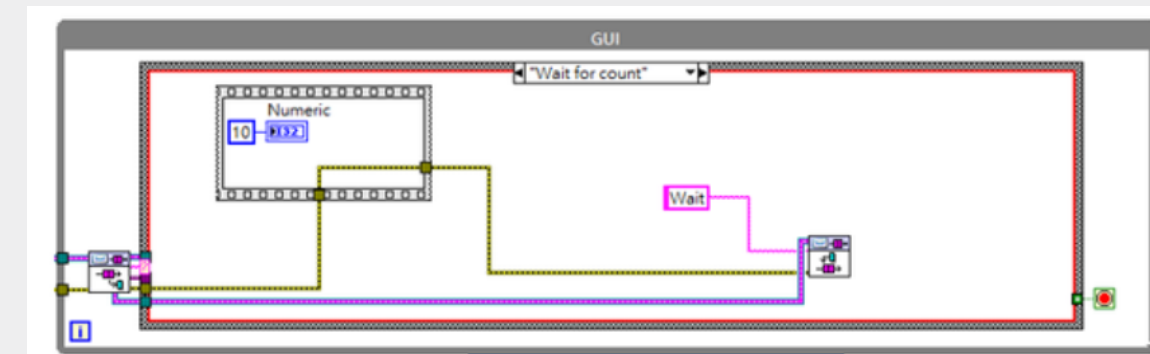
Les points d'arrêt « mettent en pause » le code LabVIEW jusqu'à ce que le développeur réactive l'exécution



Câble



Noeud / sousVI



Diagramme



Exécution détaillée (Ctrl-Flèche bas)

Ouvre un nœud et pause l'exécution du code.
Pause à la prochaine action du sous-VI ou de la structure.



Exécution sans détail (Ctrl-Flèche droite)

Exécute un nœud et pause au prochain nœud.



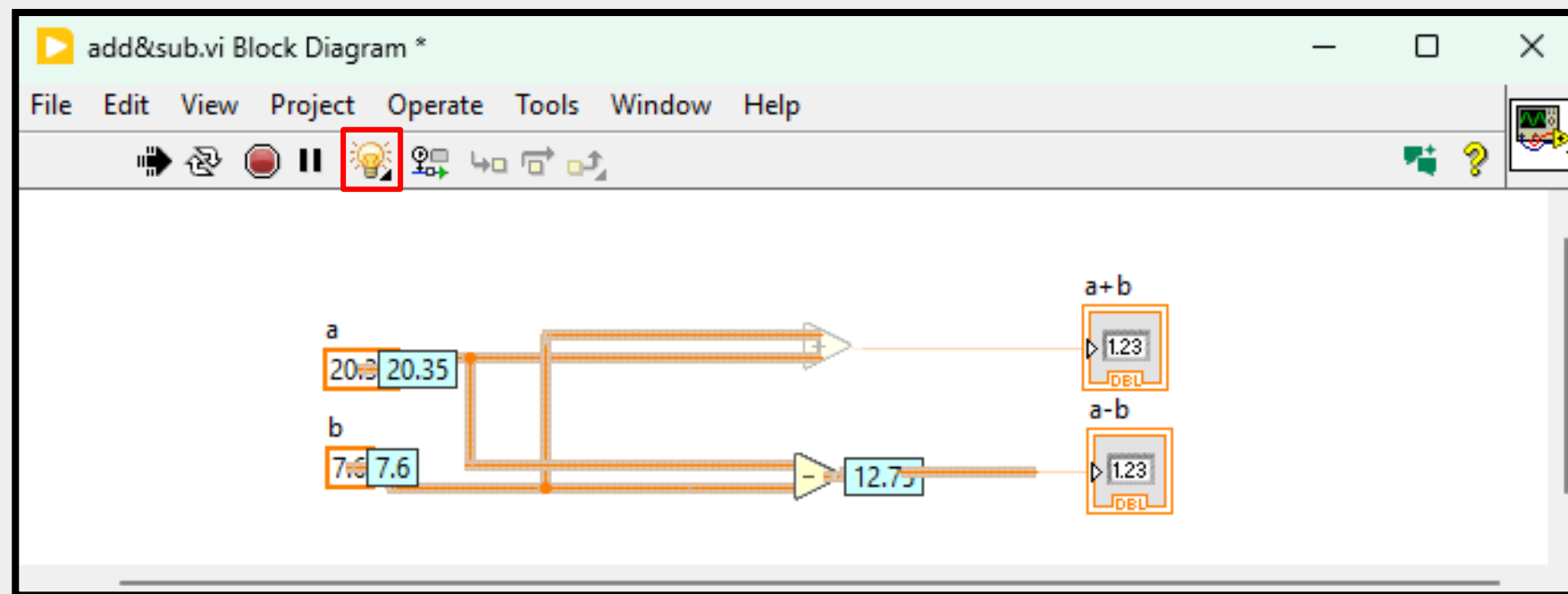
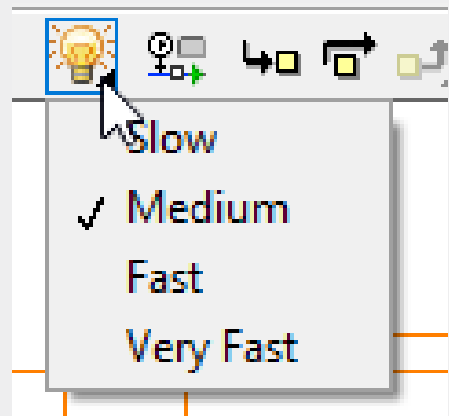
Quitter (Ctrl-Flèche haut)

Arrête l'exécution du nœud actuel et pause. Lorsque le VI a fini son exécution, le bouton « Quitter » est grisé.

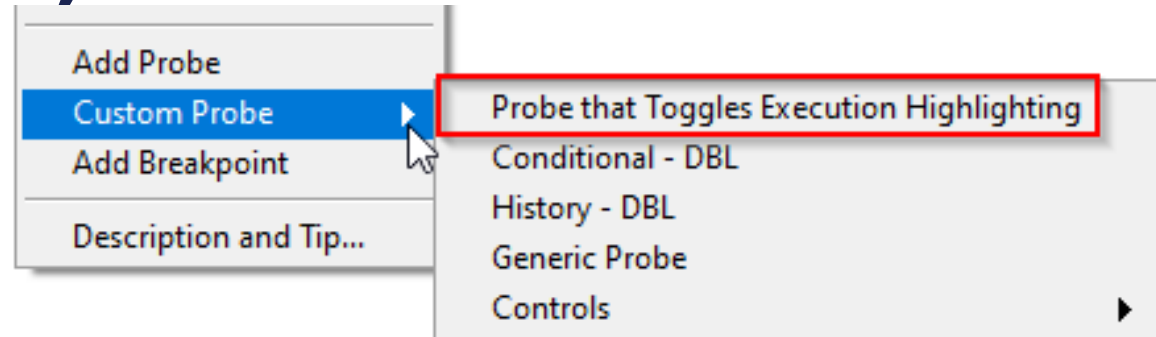
6) Animation de l'exécution



L'animation de l'exécution présente le mouvement des données d'un nœud à un autre sur le diagramme en utilisant des "bulles" qui se déplacent le long des fils de liaison.



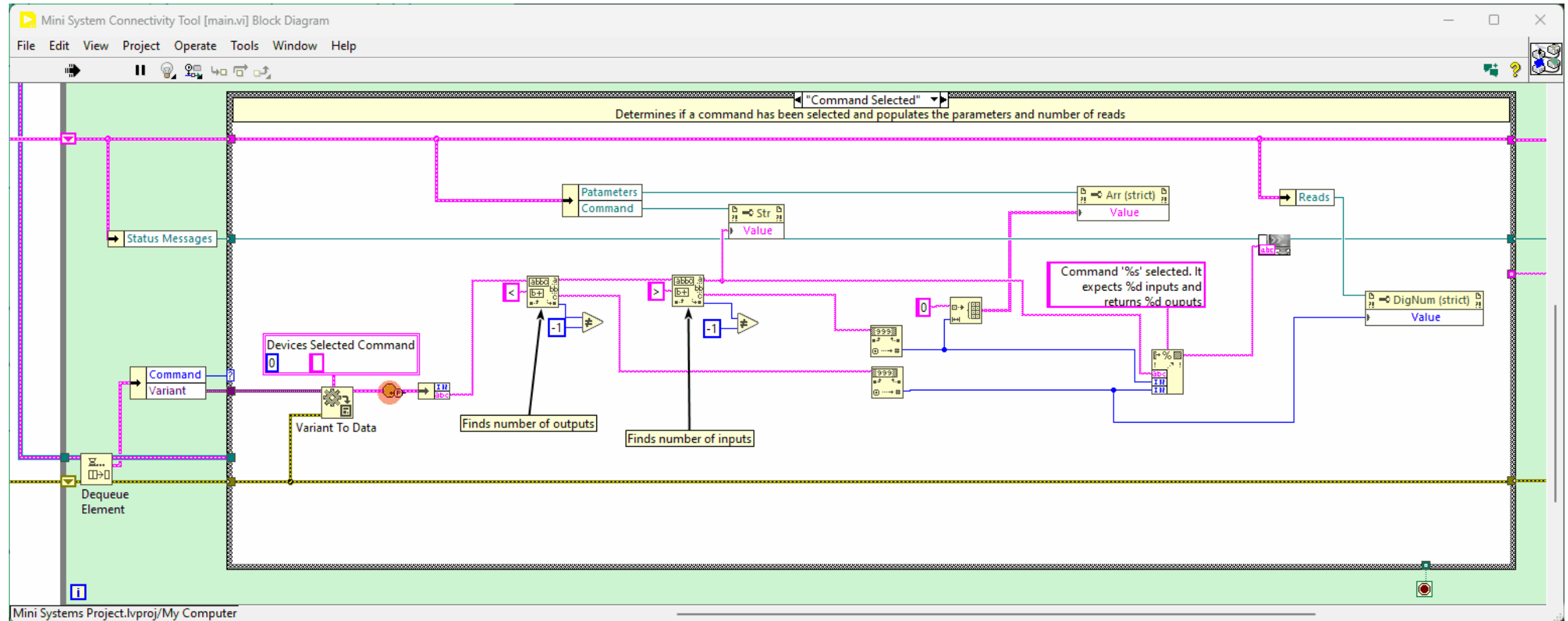
7) Animation de l'exécution – Sonde déclenchant l'animation



Probes and Breakpoints	Value	Last Update
main.vi (Mini Systems Project.lvproj/My Computer)		
☐ [1] queue out	Execution Highlighting Disabled	1:37:03.045 PM

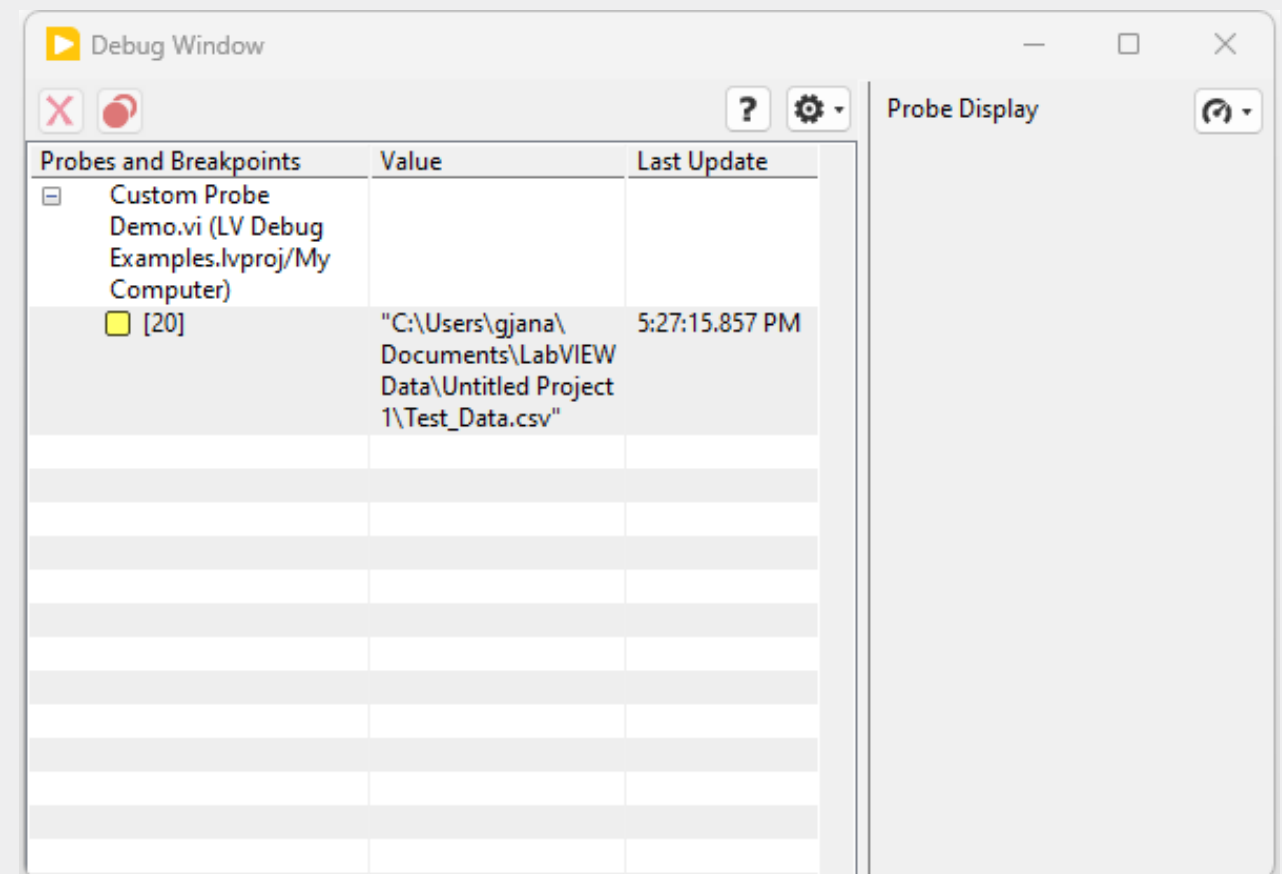
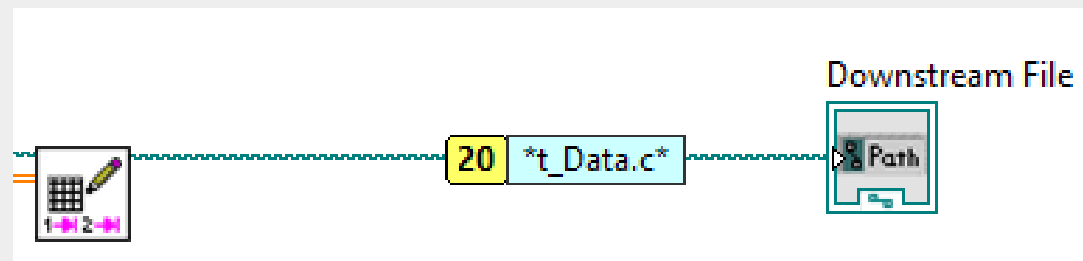
- Les sondes et points d'arrêt sont gérés depuis une fenêtre Et la sonde d'erreur conditionnelle qui mets un point d'arrêt en cas d'erreur

Animation de l'exécution – Sonde déclenchant l'animation



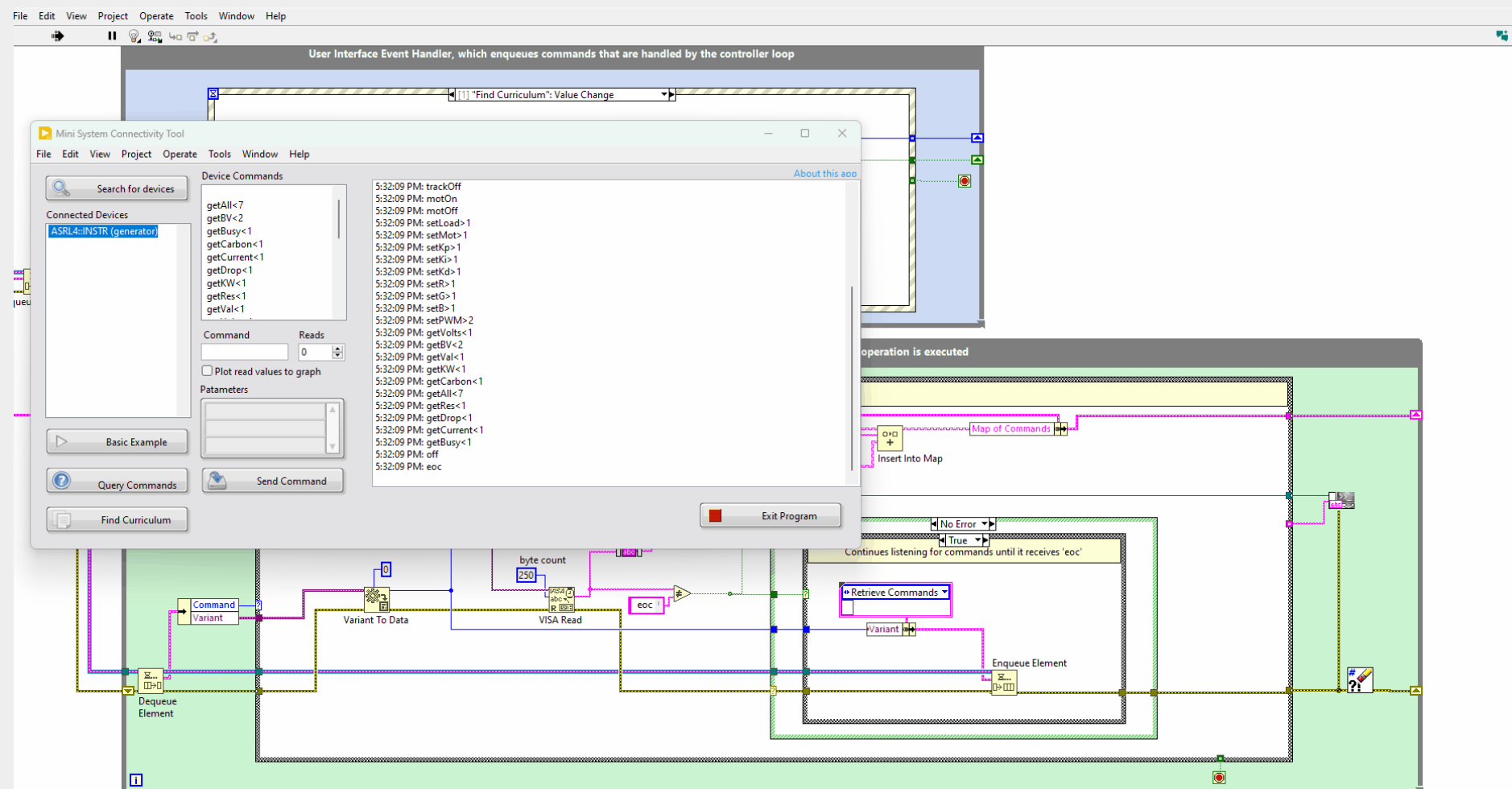
8) Sondes – Sondes génériques Vls. Sondes personnalisées

Les sondes génériques sont uniquement utilisées pour visualiser les données « sur un fil de donnée »,
Mais qu'en est-il si nous voulons avoir davantage d'informations ?

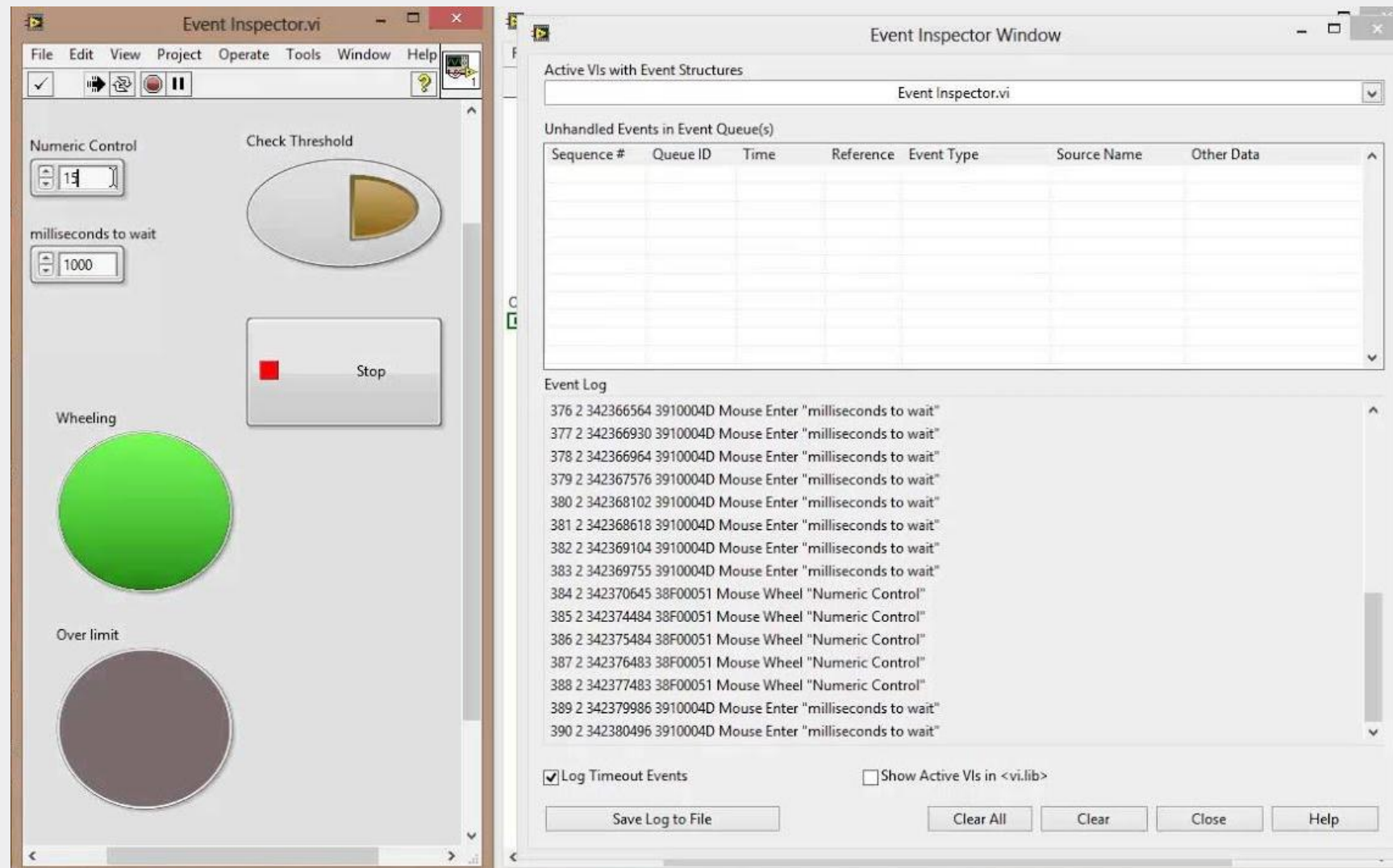


Sondes – Sondes d'historique

Les sondes d'historique peuvent être créées pour des numériques, des chaînes de caractères, des chemins et des booléens.



9) L'observateur d'évènements



10) La gestion d'erreur automatique : Inconvénients et divergences d'avis entre les développeurs

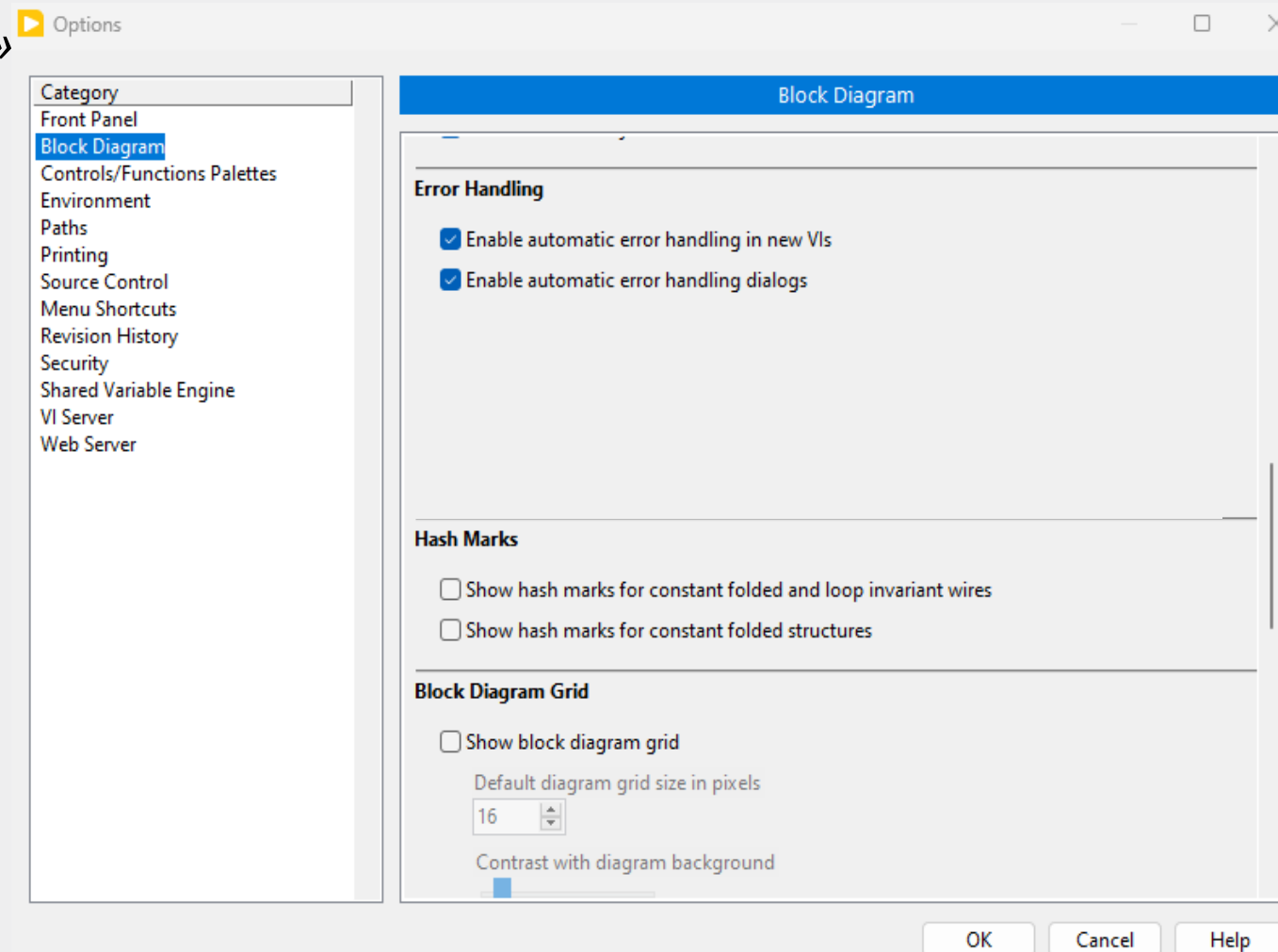
Amaury LAURENT : « Utiliser la boîte d'erreur automatique, c'est comme faire du vélo avec les roulettes »

Luc DESRUELLE : « je l'active pour améliorer ma gestion d'erreur »

Suspend/Stop l'exécution du code source sur les erreurs qui ne sont pas câblées

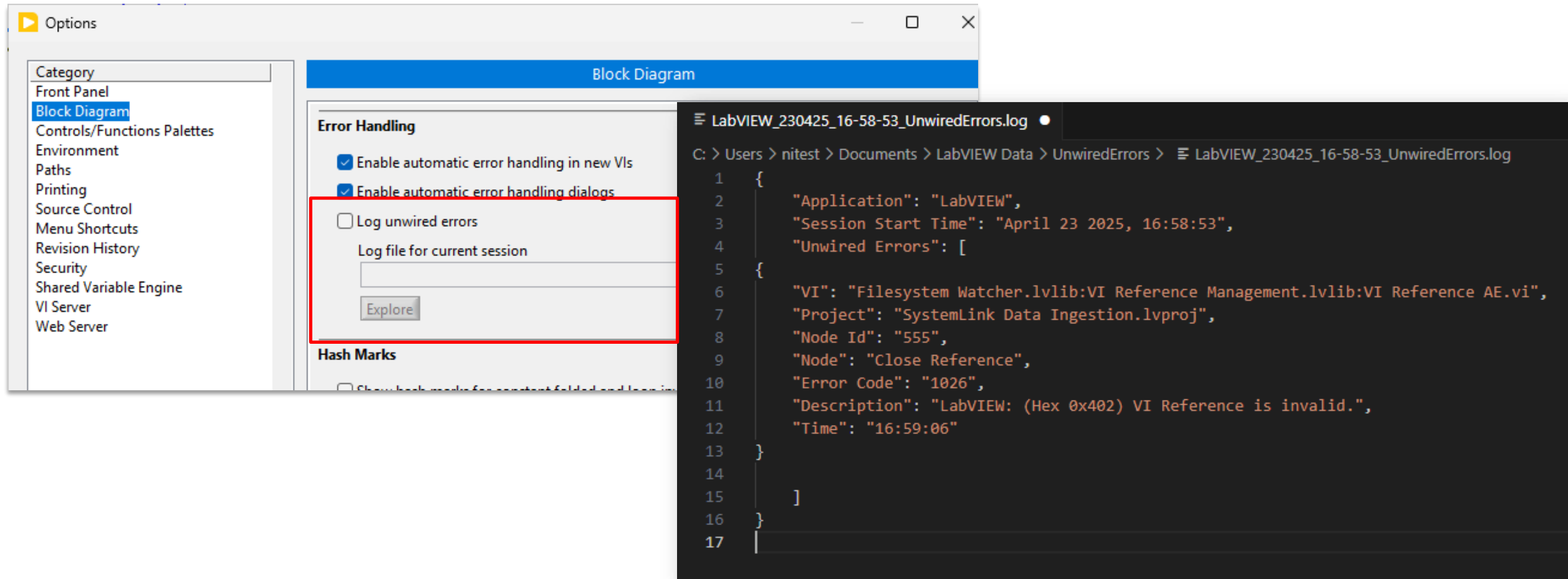
Pourquoi certains veulent désactiver cette fonctionnalité

- Eviter les interruptions par l'environnement de développement LabVIEW. Sur erreur non câblées, avec cette option, l'EDI LabVIEW interrompt toutes les exécutions de code source, en complet
- Dans les versions antérieures, cette fonctionnalité est disponible uniquement en source donc dans l'IDE, Pas dans les exécutables.



Gestion d'erreur automatique : Enregistrer les erreurs non câblées en source et.... En executable

NEW
LabVIEW 2025Q3



The image shows two overlapping windows from the LabVIEW 2025Q3 interface. The background window is the 'Options' dialog, with the 'Block Diagram' category selected. Under the 'Error Handling' section, the 'Log unwired errors' checkbox is highlighted with a red rectangle. The foreground window is a text editor showing the contents of a log file named 'LabVIEW_230425_16-58-53_UnwiredErrors.log'. The log file contains a JSON entry for an error.

Options Dialog - Error Handling Section:

- ☒ Enable automatic error handling in new VIs
- ☒ Enable automatic error handling dialogs
- ☐ Log unwired errors
- Log file for current session: [Empty text box]
- Explore

Error Log File Content:

```
1 {
2   "Application": "LabVIEW",
3   "Session Start Time": "April 23 2025, 16:58:53",
4   "Unwired Errors": [
5     {
6       "VI": "Filesystem Watcher.lvlib:VI Reference Management.lvlib:VI Reference AE.vi",
7       "Project": "SystemLink Data Ingestion.lvproj",
8       "Node Id": "555",
9       "Node": "Close Reference",
10      "Error Code": "1026",
11      "Description": "LabVIEW: (Hex 0x402) VI Reference is invalid.",
12      "Time": "16:59:06"
13    }
14  ]
15 }
16 }
17 |
```

*Pour activer cette fonctionnalité dans un EXE ou PPL, vous devez inscrire cette clé dans le fichier .ini :
LogUnwiredErrors=True

Les avantages de la mise au point avec NI LabVIEW

La programmation graphique offre plusieurs atouts :

1. Les données ont un typage strict et un code couleur associé.
2. Le flux de données rend l'ordre d'exécution explicite.
3. Toutes les fonctions (sous-VIs) ont une interface utilisateur.
4. Des sondes peuvent être ajoutées sans modification du code source.
5. L'accès concurrent est clairement visible dans le code.
6. La gestion des erreurs est explicitement visible
7. La compilation du code source est automatique et constante. (flèche brisée si impossible)
8. L'environnement de développement (EDI) NI LabVIEW a de nombreuses fonctions de mise au point
9. Il existe de nombreux outils - professionnels NI et autres - pour aider à la mise au point

Memory Leaks

La spécificité des fuites mémoire

Symptômes des fuites mémoire

Symptômes de l'hôte

1. Les applications crashent après un long temps d'exécution ou de nombreuses itérations.
2. Ralentissement progressif ou instabilité.
3. Mémoire insuffisante ou crashes.
4. Les processus doivent être « tués » pour récupérer de l'espace mémoire.
5. La mémoire augmente régulièrement au fil du temps dans le gestionnaire des tâches.

Évolution du cours du baril de pétrole "Brent crude oil" depuis 1 an (en dollars)

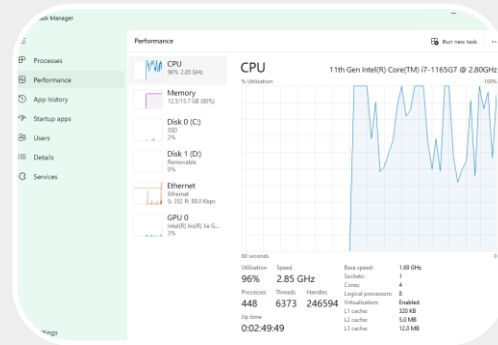


Causes principales des fuites mémoire

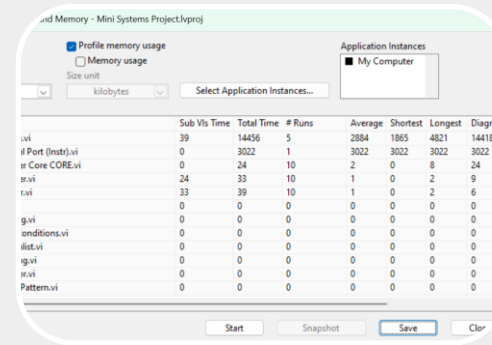
Les causes les plus courantes d'une fuite mémoire sont :

1. L'utilisation d'un « **Build Array** » dans une boucle **Tant que** : enregistrer des données sans limite définie.
2. **La concaténation de chaînes de caractères** : construction d'une chaîne « infinie », sans jamais la limiter.
3. **FIFO Queue, une file d'attente utilisée avec un buffer infini** : producteur plus rapide que le consommateur → la file s'allonge... jusqu'à la fin de vos jours.
4. **Structures de variants basées sur des DVR** qui ne libèrent jamais leurs données.
5. **Graphique (Chart) en face-avant** avec un historique de données volumineux lors d'une acquisition continue.
6. **Références** ouvertes en boucle et **non fermées**

Outils de détection des fuites mémoires

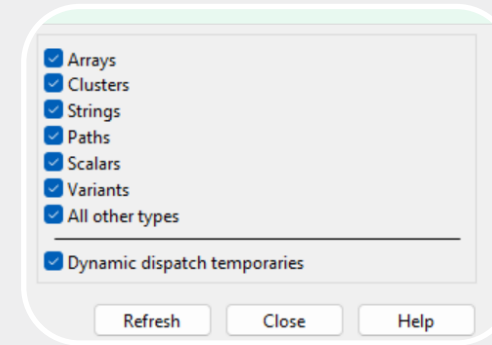


Gestionnaire des tâches



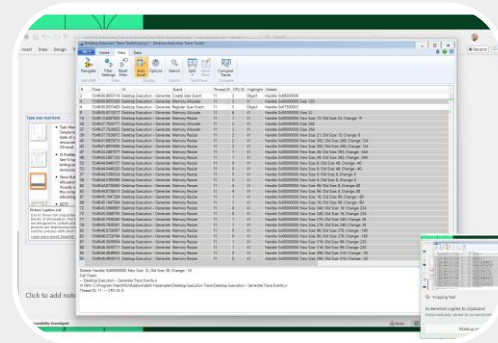
VI profiler - Profil Performance et Mémoire

Identifier les VIs les plus sollicités et mesurer temps d'exécution + mémoire



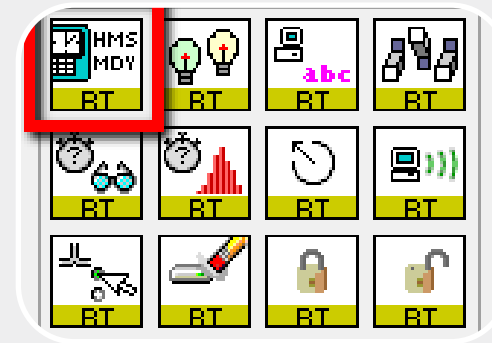
Afficher la visualisation des buffers

Voir où le compilateur alloue de la mémoire



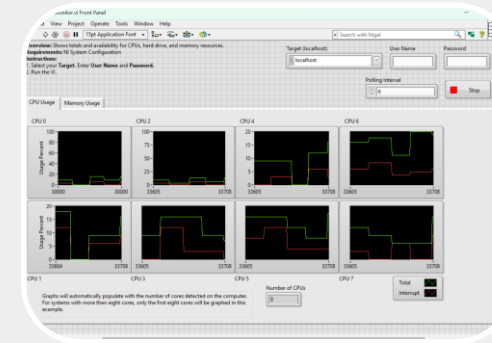
DETT

Analyser et surveiller dynamiquement l'exécution du code



RT Utilities Palette

VIs pour monitorer la mémoire, CPU et le jitter (variations de latence)



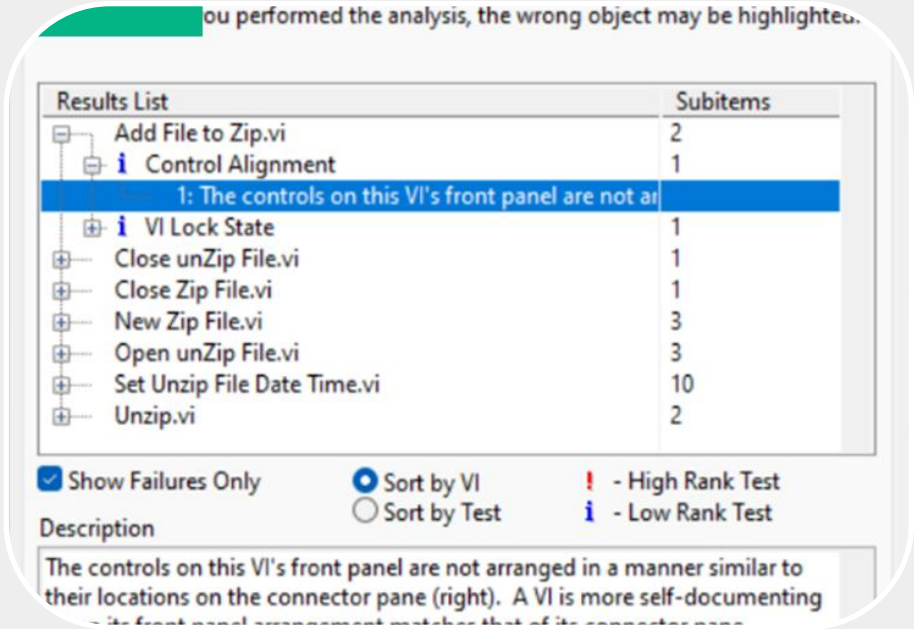
Resource Monitor

Exemple fourni pour lire les informations CPU et mémoire sur systèmes distants

Les outils NI pour vous aider

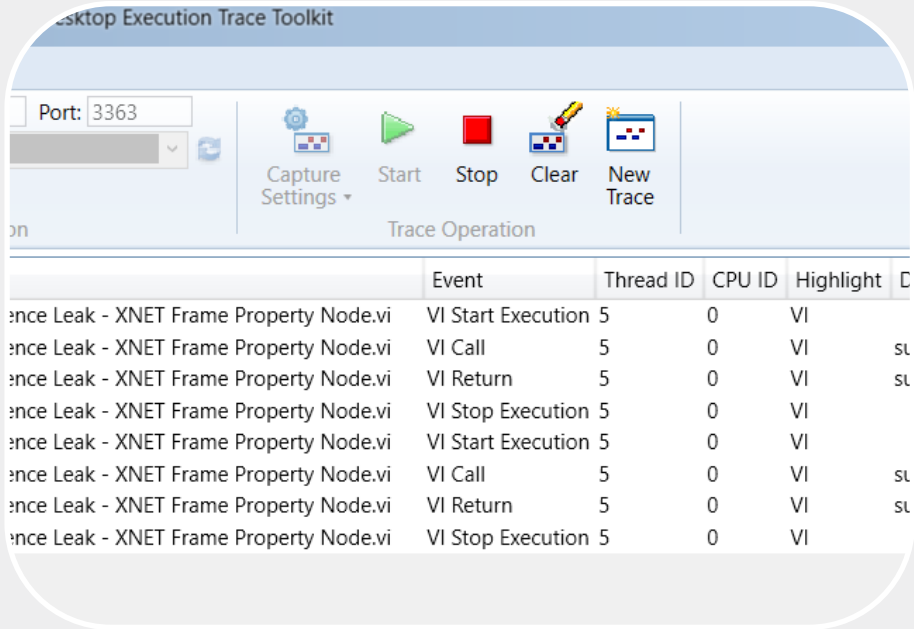
Les outils NI pour la qualité logicielle

Outils NI pour isoler les problèmes logiciels



VI Analyser

- Analyse statique du code source
- Analyse la qualité du code
- Possibilité d'ajouter des tests

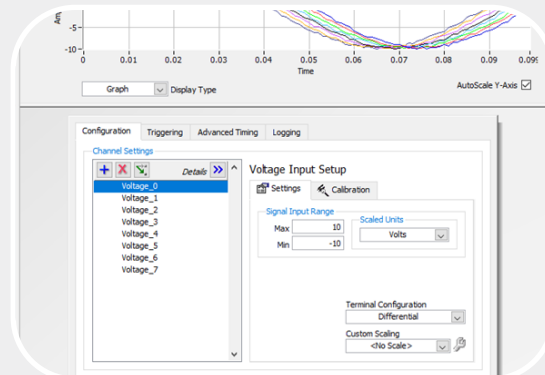


Desktop Execution Trace Toolkit

Analyse dynamique, en RunTime

Outils NI pour isoler les problèmes matériels

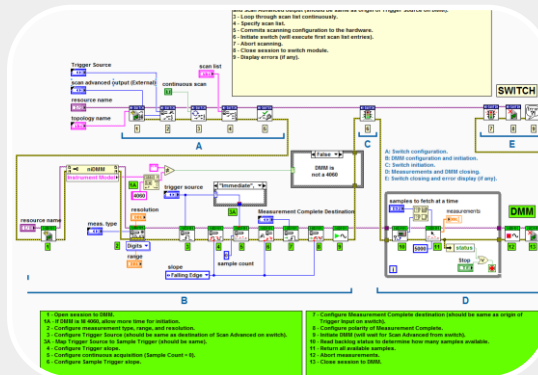
Solutions logicielles pour évaluer les performances et la connectivité du matériel



NI MAX

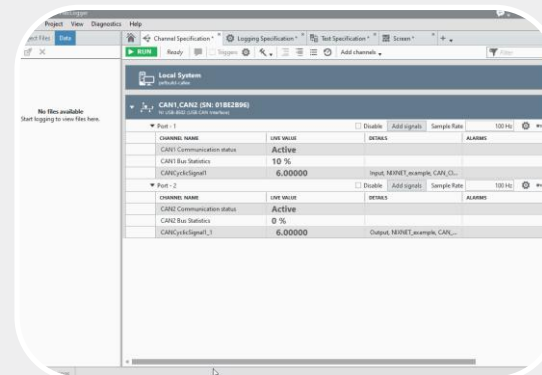
Measurement & Automation
eXplorer
L'outil de base

Vérifie l'identité des appareils, des
connexions et l'état du pilote.



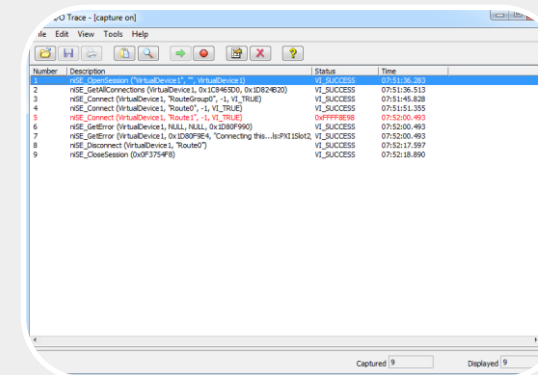
Exemples NI LabVIEW

Exécuter un code de référence fonctionnel pour valider l'interaction du pilote avec le matériel.



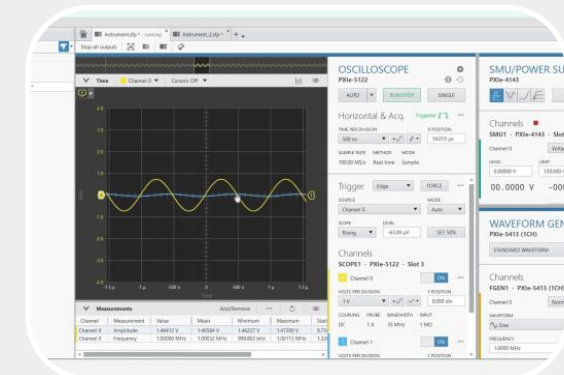
FlexLogger

Se connecte directement au matériel NI-DAQ pour visualiser les données en direct.



IO Trace

Permet de voir exactement les appels/commandes envoyés et leur date d'envoi.



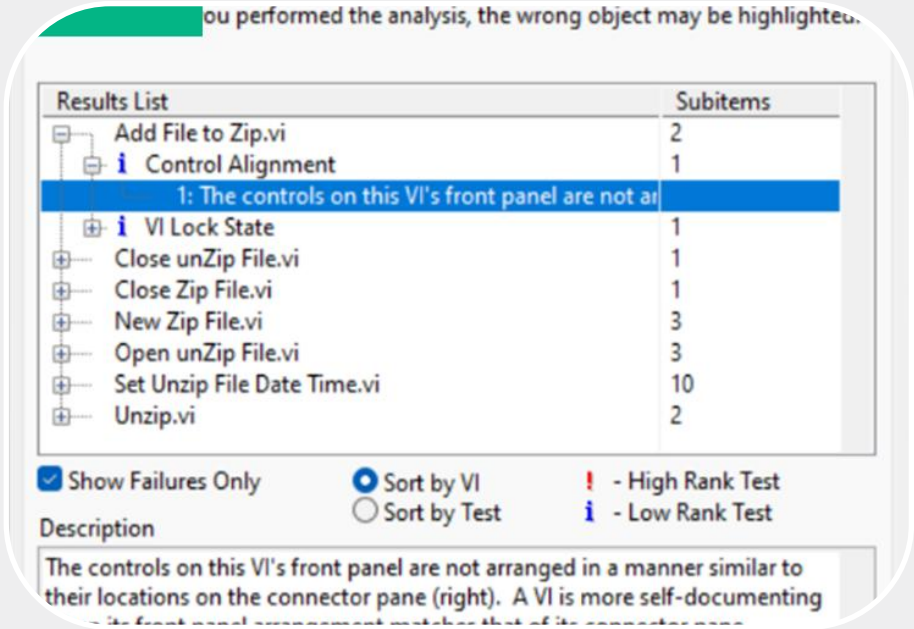
Instrument Studio

Interagit directement avec le matériel NI-PXI pour voir les données en direct et injecter des signaux bruts.

Outils NI pour isoler des problèmes de code source

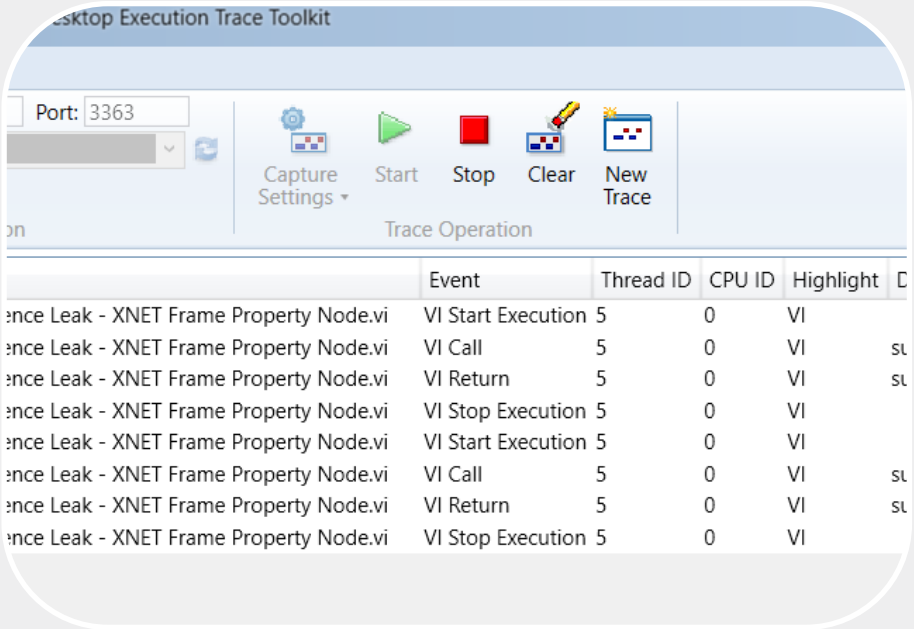
Exemples d'utilisation des outils NI pour la qualité logicielle

Outils NI pour isoler les problèmes logiciels



VI Analyser

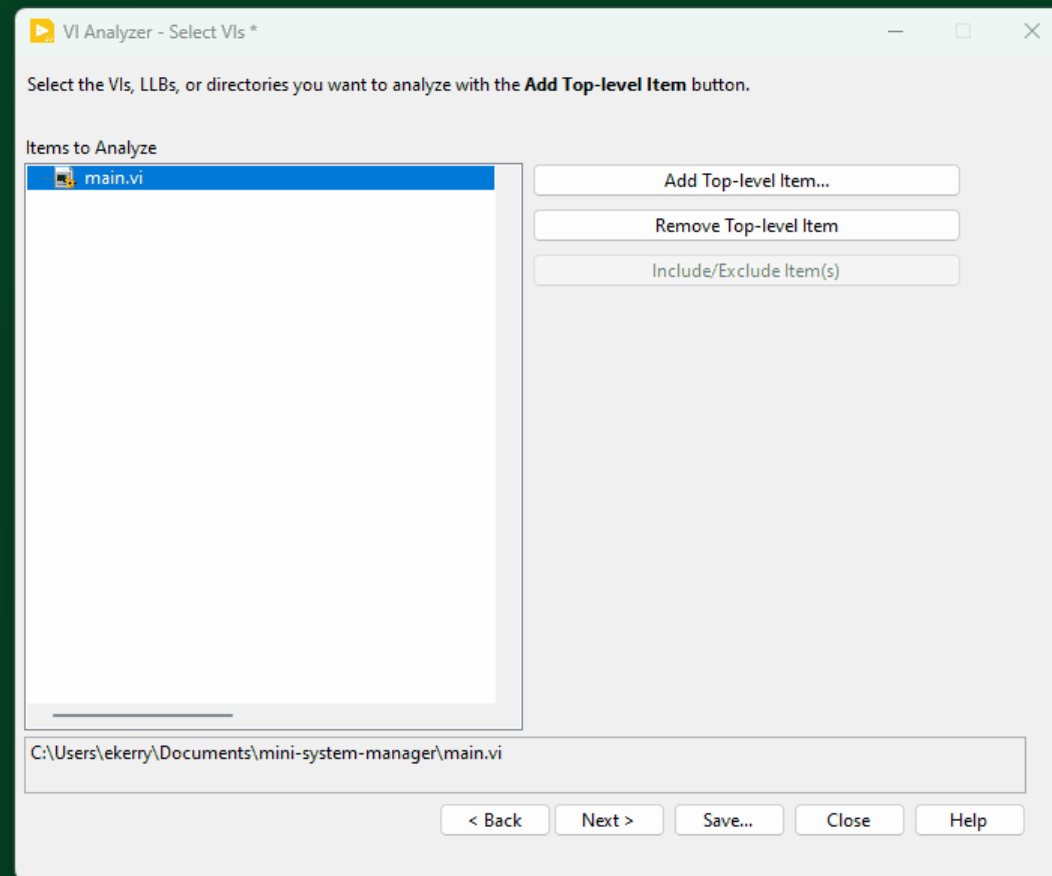
- Analyse statique du code source
- Analyse la qualité du code
- Possibilité d'ajouter des tests



Desktop Execution Trace Toolkit

Analyse dynamique, en RunTime

VI Analyzer : Analyse statique du code source



- Inspecter le code, de façon interactive ou automatisée, pour identifier les causes des problèmes courants grâce à plus de 90 tests standardisés.
- Pratique pour des revues de code interactives et l'analyse de la qualité d'un code.
- Tests configurables pour des problèmes impactant la **performance, mémoire, lisibilité** etc...
- Possibilité de créer des tests pour appliquer des règles et **directives spécifiques**.
- API programmable pour une utilisation dans le cadre des processus d'intégration continu.

Engineer
Ambitiously.

Vérifier la qualité de votre code : VI Analyser Toolkit – SAST

NI VI Analyser



Results Window

Testing ErrorsSummary

Failure occurrence to highlight the failure object in the VI. If the VI you performed the analysis, the wrong object may be highlighted.

Results List	Subitems
Add File to Zip.vi	2
Control Alignment	1
1: The controls on this VI's front panel are not arranged in a manner similar to their locations on the connector pane (right). A VI is more self-documenting when its front panel arrangement matches that of its connector pane.	
VI Lock State	1
Close unZip File.vi	1
Close Zip File.vi	1
New Zip File.vi	3
Open unZip File.vi	3
Set Unzip File Date Time.vi	10
Unzip.vi	2

☒ Show Failures Only

☒ Sort by VI

- High Rank Test

☐ Sort by Test

- Low Rank Test

Description

The controls on this VI's front panel are not arranged in a manner similar to their locations on the connector pane (right). A VI is more self-documenting when its front panel arrangement matches that of its connector pane.

Export...

Load...

Save...

Done

Help



Nerys Group SAST Toolkit

VI Analyzer RG

File Edit Help

Mesulog

VI Analyzer RG

V1.0.0b.24110510:07:5826/02/2025

	Source	Common
HMI/T&D HMI Decode Test Connection Remote PC.vi	8	
Unused Code	3	
Overlapping Controls	5	
HMI Data Simulator/T&D SIM Write Disconnect.vi	4	
Tunnel Position	3	
For Loop Iteration Count	1	
HMI/T&D HMI Init Bay's name.vi	0	
Panel Size and Position	1	
Overlapping Controls	1	
For Loop Iteration Count	2	
HMI/T&D HMI Read Breaker ID From DataBase.vi	2	2
Protected/MES XLS Workbook protection init.vi		1
Accessors/Write VI Refnum.vi	1	
Accessors/Write SubPnl Refnum.vi	1	
Source/T&D _Tree.vi	0	
ServiceStopper/T&D UTL ProcessStop(EmptyVI).vi	1	
Utl/T&D UTL Get DEM_EV DataSocket URL.vi		1
BackupData/T&D UTL Backup File.vi	1	
HMI Mobile/T&D UMPC _Main.vi	1	
Mimic Test/T&D MIM XLS Save Error Data.vi	1	
Mimic Test/T&D MIM XLS Save Compartment Data.vi	1	
Mimic Test/T&D MIM XLS Generate Report.vi	1	
HMI/T&D HMI FGV Get Pictures File Path.vi	1	
Source/T&D DSM Global Data.vi		1
DataFormatter/T&D DFO Manage Diagnostic Nb CPU.vi	1	
Decode Data/T&D DFO Decode Event Data.vi	1	
DataFormatter/T&D DFO CONF Read NbRFC.vi	1	
Decode data/T&D DEM Manage Event Density Average Data.vi	1	
DataEventManager/T&D DEM FIL Manage New File.vi	1	
DataEventManager/T&D DEM Density Calculate Density average.vi	1	
DataBase/T&D DB_ Manage Event Only ID Gateway.vi		1
DataBase/T&D DB_ Encode Events _Internal_Arc.vi		1
DataBase/T&D DB_ Encode Event Only ID Gateway.vi		1
DataBase/T&D DB_ Decode Events _Perturbographie_Arc.vi		1
DataBase/T&D DB_ Decode Events _Offset.vi		1
DataBase/T&D DB_ Decode Events _Module_Number.vi		1
DataBase/T&D DB_ Decode Events _Leakage_Speed.vi		1
DataBase/T&D DB_ Decode Events _Density_Average.vi		1
DataBase/T&D DB_ Decode Events _Delta_P.vi		1
DataBase/T&D DB_ Decode Events_DS_Ref Full Binary.vi		1
DataBase/T&D DB_ Decode Events _ConfigVersion.vi		1

C:\Mesulog\T-D1 Supervision_V11.0_LV2024\Source\HMI\T&D HMI Init Bay's name.vi

VI description

Permet de créer la liste des "Bay's" avec l'ID du RFC et de la bay.

Standard VI

Open VI

VIA Ignore List

Panel Size and Position

Panel Size and Position NI : Medium / Mesulog : Medium (Source 1 / Common 1)

Checks that a front panel completely resides within the bounds of the screen. The test also checks whether the front panel is larger than the maximum specified width and height. If you are using a multi-monitor system, the test fails if the panel does not reside entirely within the bounds of the

Test comment (for report)

Cosmétique

add '#via_ignore Panel Size and Position' for 'HMI/T&D HMI Init Bay's name.vi'

add '#via_ignore Panel Size and Position' for all failed vi

Source

Common

Delete all '#via_ignore Panel Size and Position'

UpdateTree

Cosmétique

Auto fix functions ...

Update #via_ignore

Read 1 vi

Build Report

65 Source / 35 Common

jeudi 7 novembre 2024 00:09:06

☐ Masquer les erreurs

Réinitialiser les erreurs

0

1

Desktop Execution Trace Toolkit : analyse dynamique

Le **LabVIEW Desktop Execution Trace Toolkit (DETT)** s'avère particulièrement utile lorsqu'une analyse approfondie du comportement d'exécution est nécessaire, notamment pour le débogage de code complexe, événementiel ou multithread.

Scenario:	Role of DETT:
Erreurs difficiles à reproduire	Enregistrer des données au fil du temps, cela aidera à détecter les problèmes qui n'apparaissent pas à chaque exécution.
Fuites mémoire or fuites de référence	Repérer la création and destruction des références, vous pouvez ainsi repérer les fuites dans les files d'attente, notificateurs, ou évènements utilisateur.
Pertes de performance inattendue	Afficher les évènements, changement de threads, et appels bloquants, vous aidera à repérer les ralentissements.
Messagerie complexe	Visualiser les messages et leur synchronisation, rendra plus simple le débogage d'architecture producteur/consommateur ou QMH.
Déboguer des applications (.exe)	Se connecter aux exécutables avec VI Server activé, vous permettra de déboguer du code déployé.
Suivre des appels dynamiques de VI	Enregistrement de données lorsque les VIs sont chargés, appelés, et déchargés, ce qui est essentiel dans des architectures basées sur des plugins.
Diagnostiquer des crashes/blocages	Révéler les impasses, boucles infinies, ou erreurs non gérées en récupérant la séquence d'évènements avant le crash.

Desktop Execution - Generate Trace Events.vi Front Panel on Desktop Execution Trace Toolkit.lvproj/My Computer

File Edit View Project Operate Tools Window Help

15pt Application Font

Search with Nigel

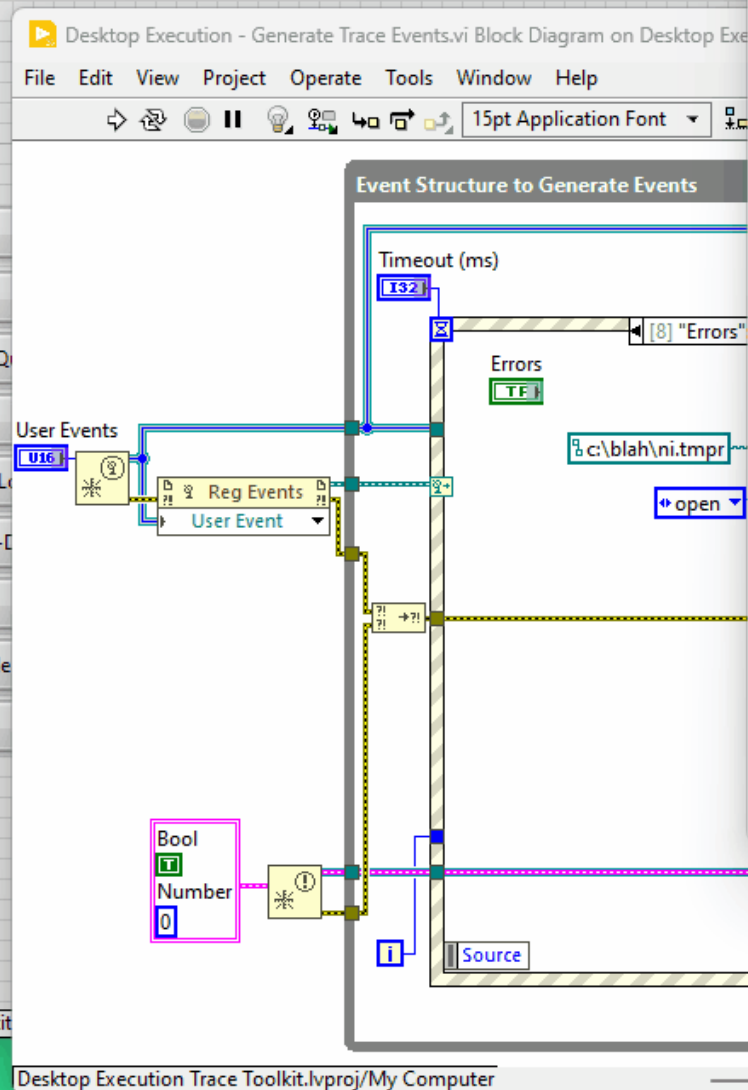
Overview: Demonstrates the types of events that the Desktop Execution Trace Toolkit can trace.

Requirements: Desktop Execution Trace Toolkit

Instructions:

Note: If you open this example by following the **Getting Started** section in the **LabVIEW Desktop Execution Trace Toolkit**

1. Launch the Desktop Execution Trace Toolkit by selecting **Tools>>Profile>>Trace Execution** from the LabVIEW
2. Select **Tools>>Options** from the LabVIEW menu bar and select **VI Server** from the **Category** list to enable VI Se
3. In the Desktop Execution Trace Toolkit, click the lower part of the **Capture Settings** button and select the Mem
4. Click **Start** on the **Home** tab.
5. In LabVIEW, run the VI. Click the buttons on the front panel to generate different types of events for the Desktop Desktop Execution Trace Toolkit.



Mini Systems Project.lvproj 1 - Desktop Execution Trace Toolkit

File Home View Data

Machine: localhost Port: 3365

Application Instance: Desktop Execution Trace Toolkit.I

Capture Settings Start Stop Clear New Trace

Target Configuration Trace Operation

#	Time	VI	Event	Thread ID	CPU ID	Highlight	Details
---	------	----	-------	-----------	--------	-----------	---------

Engineer
Ambitiously

Utiliser DETT pour comparer les traces d'exécution

Compare 1

Session Name: Desktop Execution Trace Toolkit.lvproj 2 : session 1 : 15:49:00.9054975

Begin Index: 0 End Index: 3354

	VI	Event	Thread ID	CPU ID	Highlight
09.1411776	Desktop Execution - Generate Trace Events.vi	Memory Free	0	0	VI
09.1411790	Desktop Execution - Generate Trace Events.vi	Memory Free	0	0	VI
09.1411800	Desktop Execution - Generate Trace Events.vi	Memory Free	0	0	VI
09.1411829	Desktop Execution - Generate Trace Events.vi	Memory Allocate	0	0	VI
09.1411843	Desktop Execution - Generate Trace Events.vi	Memory Allocate	0	0	VI
09.1411852	Desktop Execution - Generate Trace Events.vi	Memory Allocate	0	0	VI
09.1411867	Desktop Execution - Generate Trace Events.vi	Memory Free	0	0	VI
09.1411881	Desktop Execution - Generate Trace Events.vi	Memory Free	0	0	VI
09.1411891	Desktop Execution - Generate Trace Events.vi	Memory Free	0	0	VI
09.1411933	Desktop Execution - Generate Trace Events.vi	Memory Free	0	0	VI
09.1412620	Desktop Execution - Generate Trace Events.vi	Memory Allocate	0	0	VI
09.1447806	Desktop Execution - Generate Trace Events.vi	Memory Allocate	0	0	VI
09.1447863	Desktop Execution - Generate Trace Events.vi	Memory Free	0	0	VI
09.1447906	Desktop Execution - Generate Trace Events.vi	Memory Allocate	0	0	VI
09.1447916	Desktop Execution - Generate Trace Events.vi	Memory Free	0	0	VI
09.1447925	Desktop Execution - Generate Trace Events.vi	Memory Allocate	0	0	VI
09.1447949	Desktop Execution - Generate Trace Events.vi	Memory Allocate	0	0	VI
09.1448068	Desktop Execution - Generate Trace Events.vi	Memory Free	0	0	VI
09.1448078	Desktop Execution - Generate Trace Events.vi	Memory Free	0	0	VI

Details: Handle: 0x00000000; Size: 4
Call Chain:
-- Desktop Execution - Generate Trace Events.vi
VI Path: C:\Program Files\NI\LVAddons\det\1\examples\Desktop Execution Trace\Desktop Execution - Generate Trace Events.vi
Thread ID: 0 --- CPU ID: 0

Compare 2

Session Name: Desktop Execution Trace Toolkit.lvproj 2 : session 2 : 16:14:14.0066041

Begin Index: 3355 End Index: 43032

#	Time	VI	Event	Thread ID	CPU ID	Highlight	Details
6408	16:14:14.3349208	Resource Monitor.vi	Memory Free	0	2	VI	Handle: 0x00000000
6409	16:14:14.3349509	Resource Monitor.vi	Memory Resize	0	2	VI	Handle: 0x00000000
6410	16:14:14.3349528	Resource Monitor.vi	Memory Resize	0	2	VI	Handle: 0x00000000
6411	16:14:14.3349537	Resource Monitor.vi	Memory Allocate	0	2	VI	Handle: 0x00000000
6412	16:14:14.3349547	Resource Monitor.vi	Memory Allocate	0	2	VI	Handle: 0x00000000
6413	16:14:14.3349556	Resource Monitor.vi	Memory Free	0	2	VI	Handle: 0x00000000
6414	16:14:14.3349566	Resource Monitor.vi	Memory Free	0	2	VI	Handle: 0x00000000
6415	16:14:14.3349862	Resource Monitor.vi	Memory Resize	0	2	VI	Handle: 0x00000000
6416	16:14:14.3349881	Resource Monitor.vi	Memory Resize	0	2	VI	Handle: 0x00000000
6417	16:14:14.3349890	Resource Monitor.vi	Memory Allocate	0	2	VI	Handle: 0x00000000
6418	16:14:14.3349900	Resource Monitor.vi	Memory Allocate	0	2	VI	Handle: 0x00000000
6419	16:14:14.3349905	Resource Monitor.vi	Memory Free	0	2	VI	Handle: 0x00000000
6420	16:14:14.3349914	Resource Monitor.vi	Memory Free	0	2	VI	Handle: 0x00000000
6421	16:14:14.3350205	Resource Monitor.vi	Memory Resize	0	2	VI	Handle: 0x00000000
6422	16:14:14.3350224	Resource Monitor.vi	Memory Resize	0	2	VI	Handle: 0x00000000
6423	16:14:14.3350234	Resource Monitor.vi	Memory Allocate	0	2	VI	Handle: 0x00000000
6424	16:14:14.3350243	Resource Monitor.vi	Memory Allocate	0	2	VI	Handle: 0x00000000
6425	16:14:14.3350248	Resource Monitor.vi	Memory Free	0	2	VI	Handle: 0x00000000
6426	16:14:14.3350262	Resource Monitor.vi	Memory Free	0	2	VI	Handle: 0x00000000

Details: Handle: 0x00000000; New Size: 16; Old Size: 16; Change: 0
Call Chain:
-- Resource Monitor.vi
VI Path: C:\Program Files\NI\LVAddons\nisyscfg\1\examples\nisyscfg\Resource Monitor.vi
Thread ID: 0 --- CPU ID: 2

- Isoler l'impact des modifications du code sur le comportement de l'application
- Charger des traces enregistrées avec des filtres personnalisés
- Définir des plages d'indices de trace pour cibler le débogage.

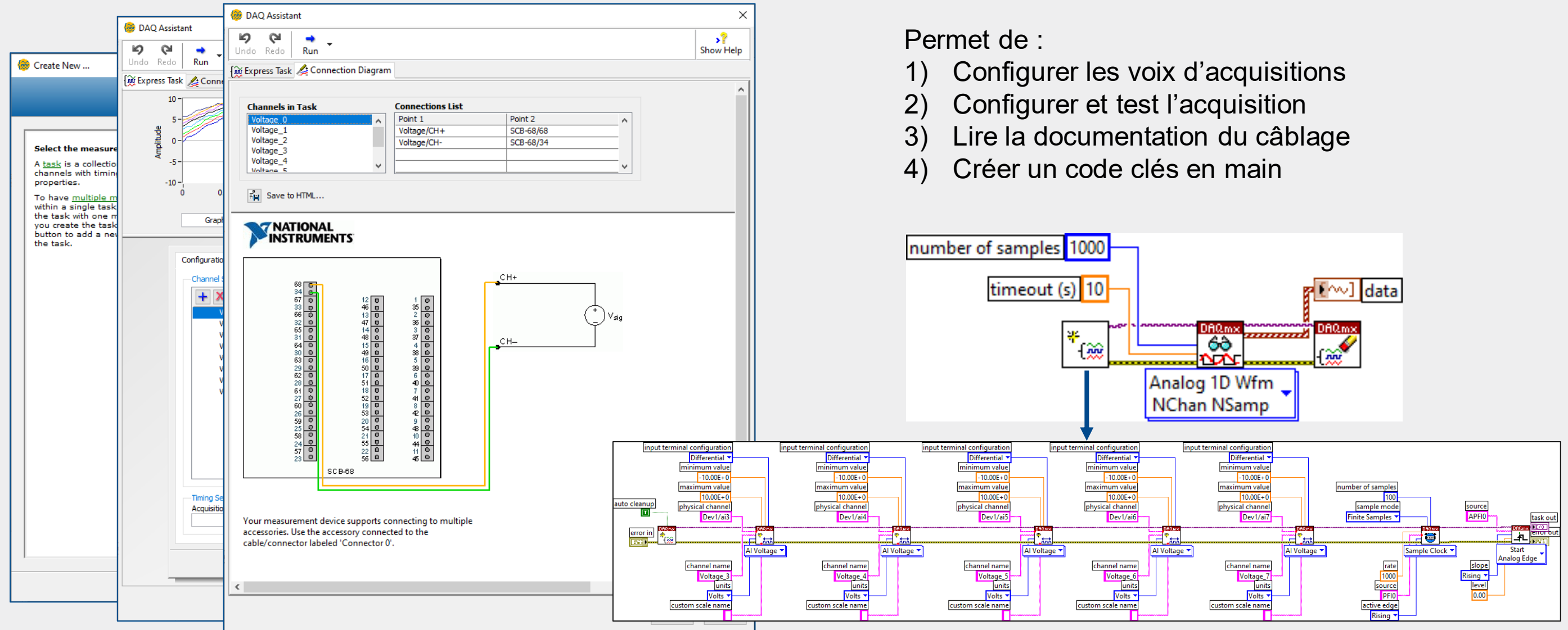
Outils NI pour isoler des problèmes matériels

Exemples d'utilisation des outils NI pour la qualité matériels

NI MAX, la base : utilitaire NI Measurement & Automation eXplorer

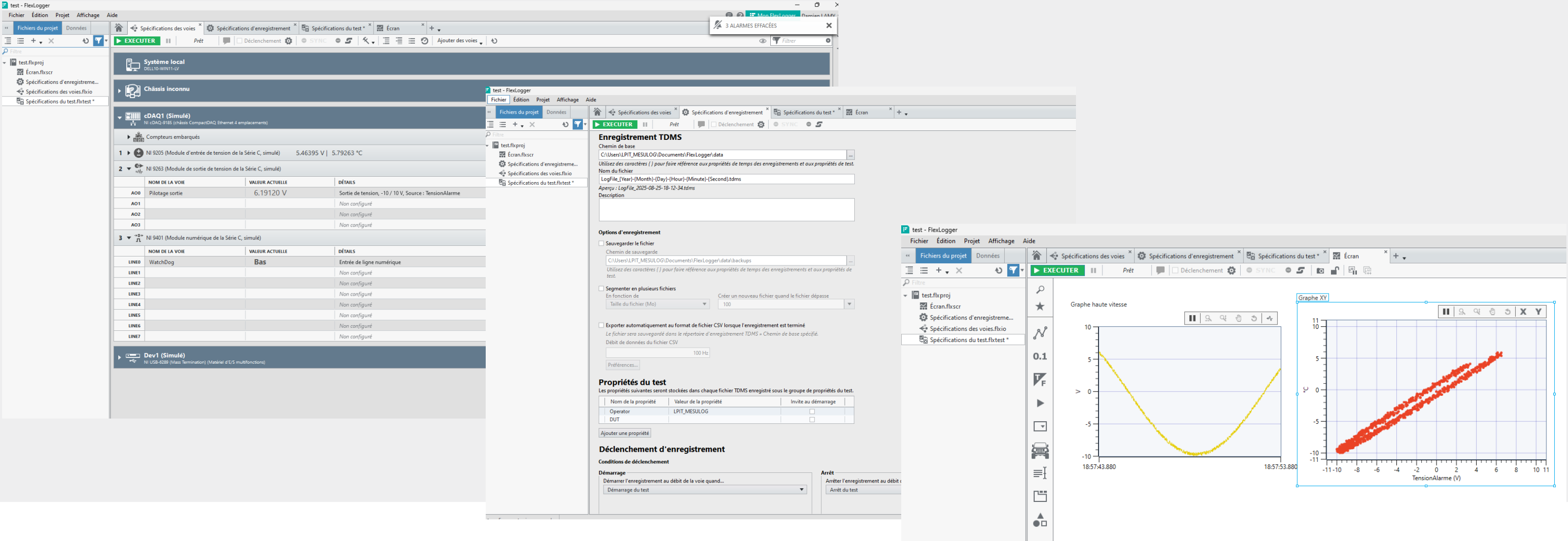
Permet de :

- 1) Configurer les voix d'acquisitions
- 2) Configurer et test l'acquisition
- 3) Lire la documentation du câblage
- 4) Créer un code clés en main

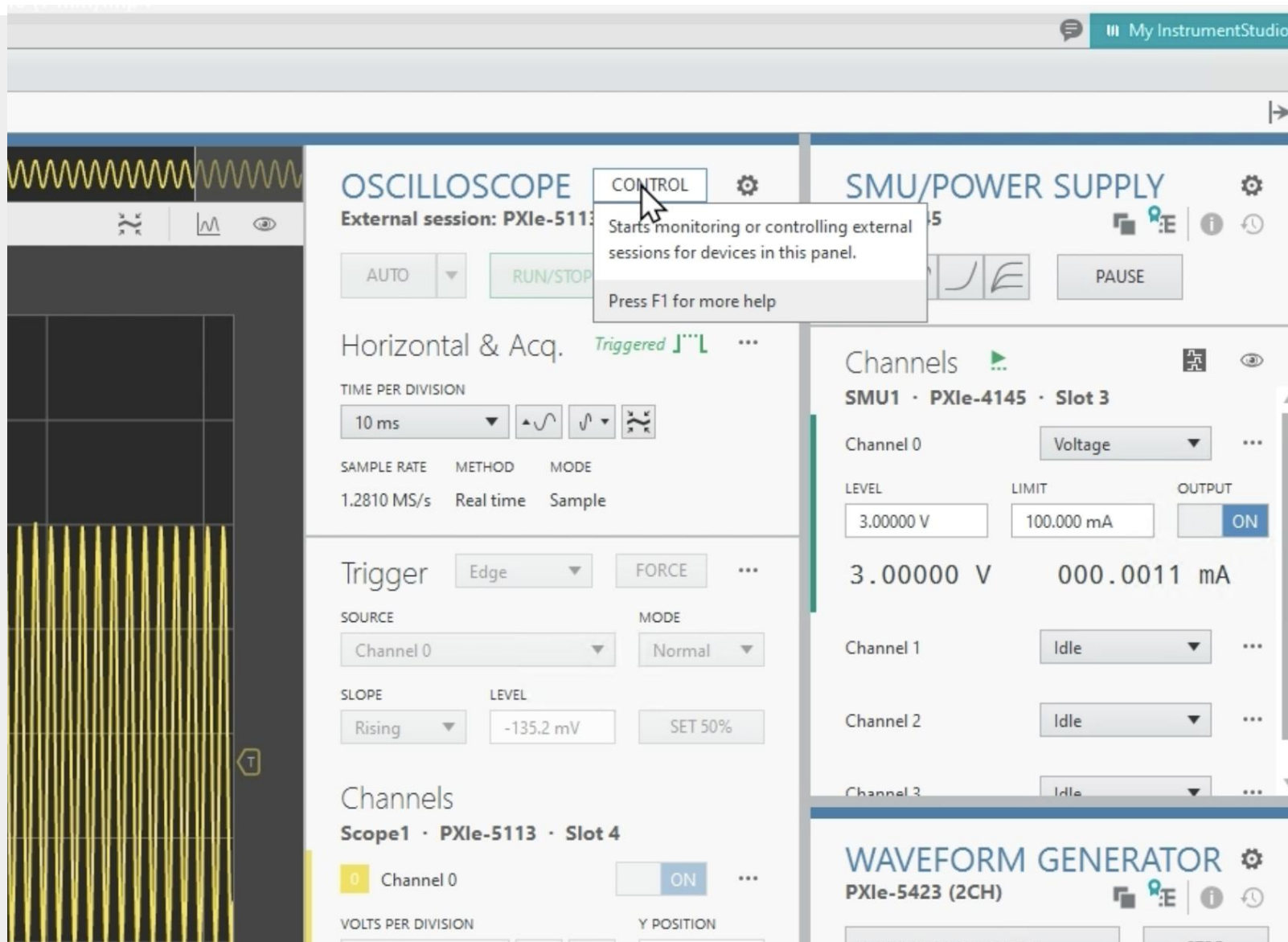


NI FlexLogger

Permet de réaliser des acquisitions, des enregistrements, des alarmes, et de visualiser les données



InstrumentStudio pour déboguer une session avec du matériel

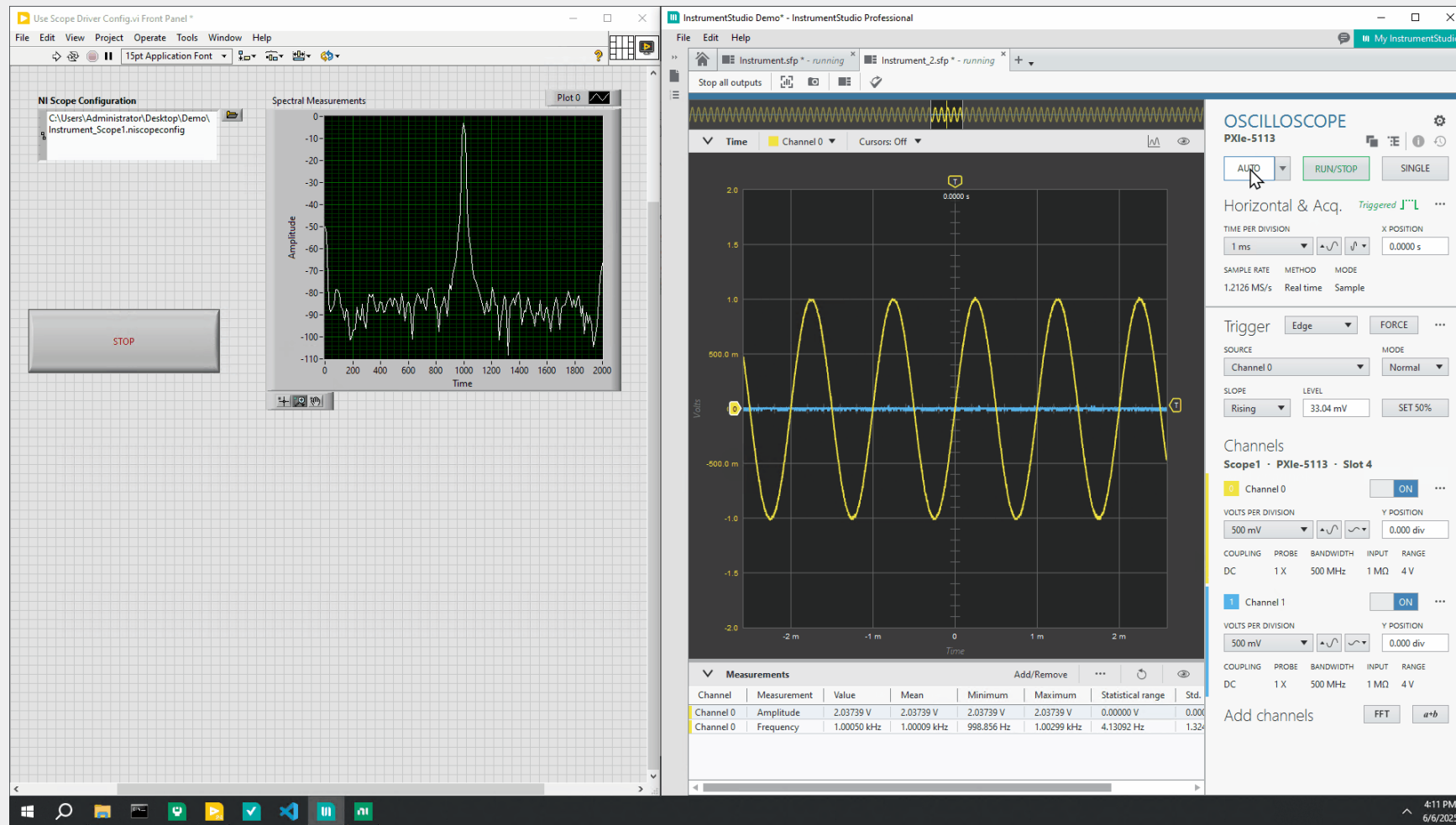


Mutualisation des sessions avec InstrumentStudio

1. Mettre en pause votre VI (point d'arrêt ou arrêt explicite) avant d'utiliser InstrumentStudio pour déboguer une session en direct.
2. Si vous cliquez sur CONTROL pendant que le VI est en cours d'exécution, LabVIEW reprendra automatiquement la session.
3. InstrumentStudio exécute une session en priorité basse — LabVIEW et TestStand prennent automatiquement la priorité.
4. CONTROL force InstrumentStudio à prendre le contrôle, mais toute activité de session externe (appels de pilote LabVIEW/TestStand) reprendra le contrôle sauf si le VI est arrêté.



Logiciel de surveillance et de débogage



- Accélère la mise au point des périphériques et des systèmes
- Identifie les problèmes matériels pendant l'exécution des programmes de test
- Contrôle interactif pour tester des hypothèses

Conclusion

Fin de l'enquête ! Merci