Olivier ZIMMERMANN
Formateur LabVIEW DR11

LPSC Grenoble

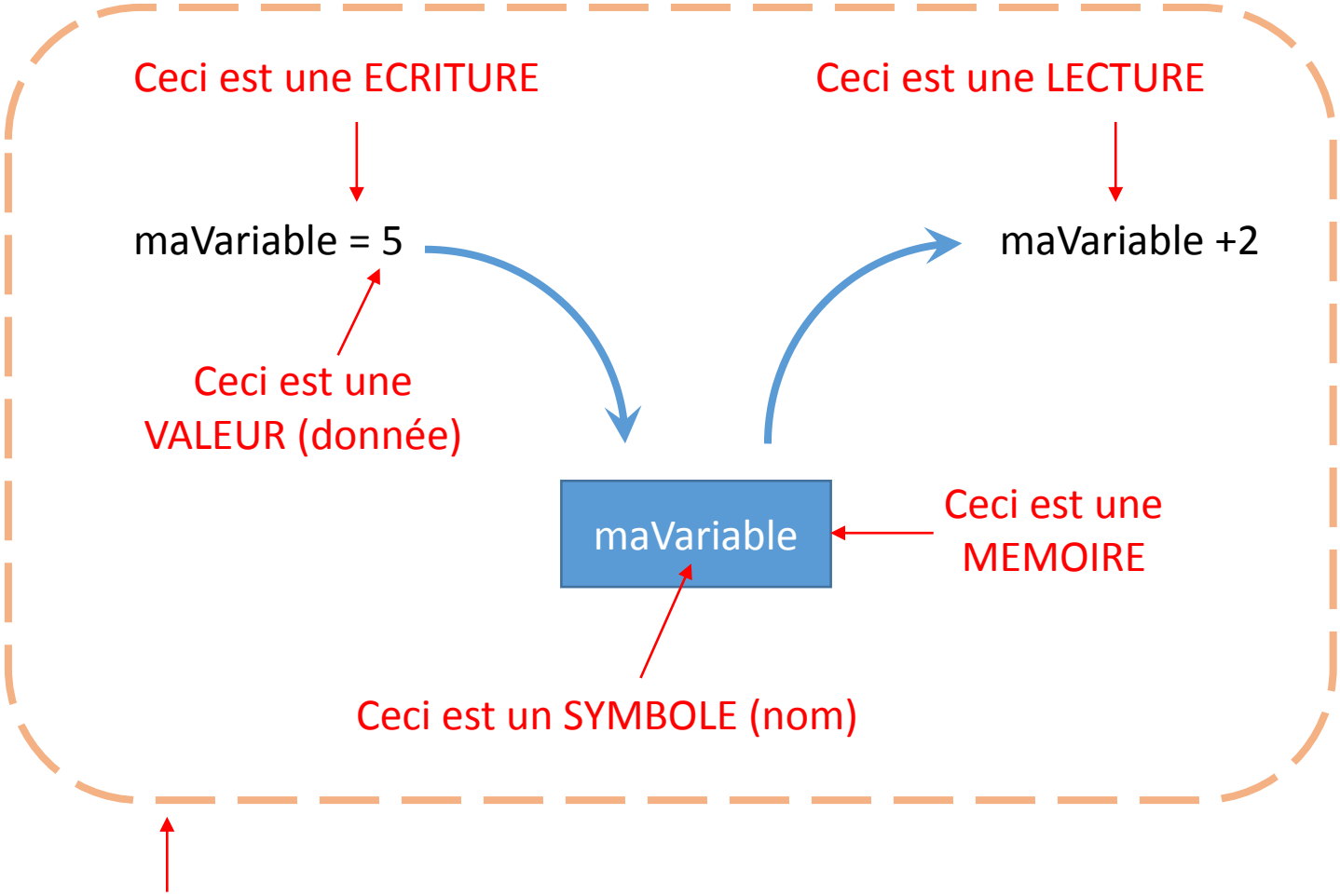
Réseau régional AlpesVIEW
Réseau national LabVIEW-ESR

Comprendre la FGV (Variable Globale Fonctionnelle) et ses extensions dans LabVIEW

Par --ChrisStuggi 22:33, 5. Aug. 2009 (CEST) — eigens Bild, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=995536

Sommaire

Qu'est-ce qu'une variable



Ceci est la PORTEE de la variable
(périmètre à l'intérieur duquel le symbole est accessible)

→ "Local" = ce fichier
→ "Global" = tous les fichiers

La portée des variables en langage G

- La portée des **flots de données** et des **registres à décalage** est limitée au processus parent (bloc, boucle). (1) (2)
- La portée de la **variable locale** LabVIEW couvre les processus du fichier parent seulement (blocs, boucles parallèles).
- La portée de la **variable globale** LabVIEW couvre tous les processus de l'application (blocs, boucles parallèles).

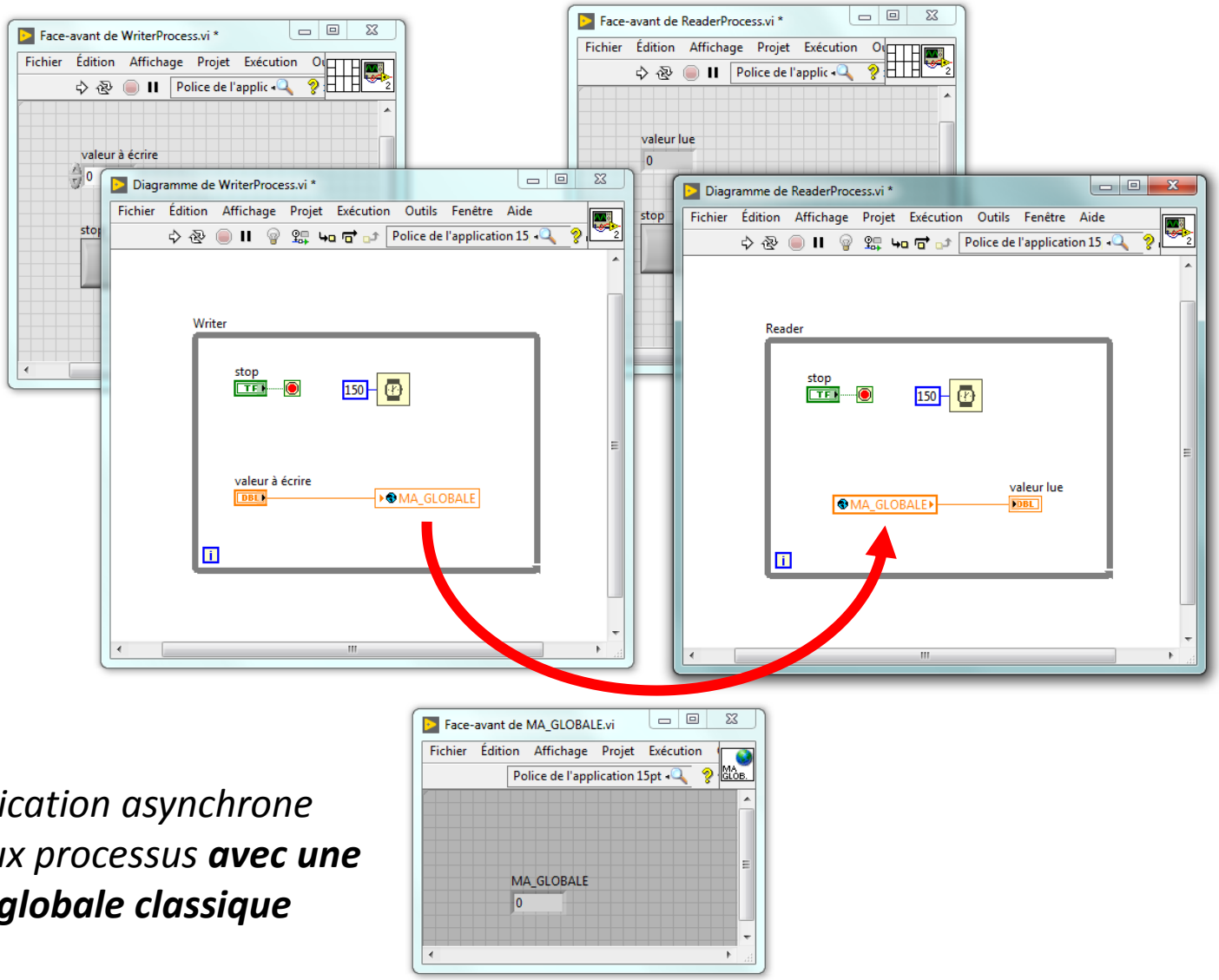
(1) Dans LabVIEW les valeurs sont normalement échangées par flots de données. *Leur portée limitée les protège*, et ils ne requièrent ni déclaration ni nommage explicite.

(2) On dit qu'il faut *casser le flot de données* pour faire communiquer des processus parallèles. Les variables locales ou globales sont les solutions les plus simples pour le faire.

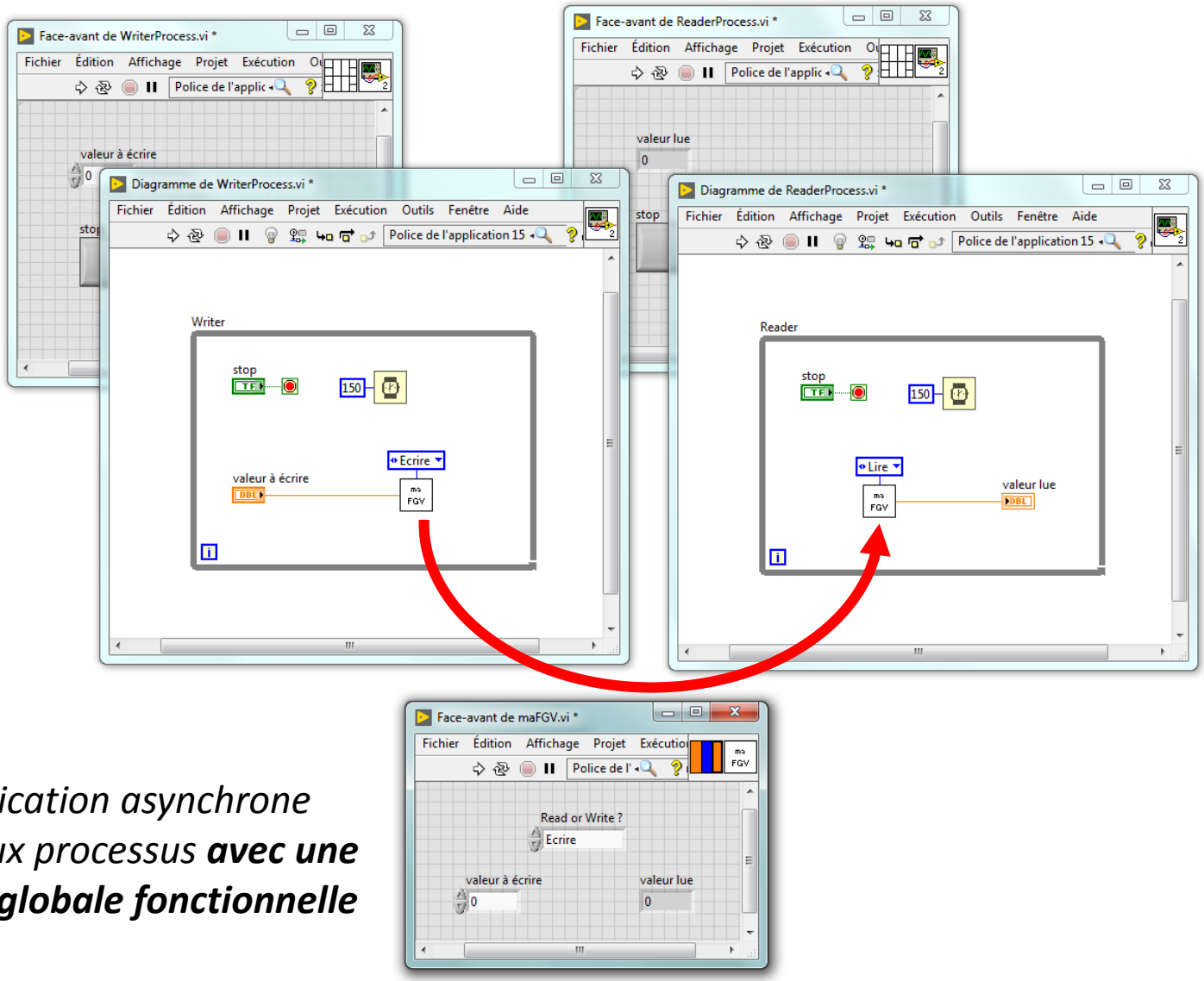
Variables Globales Fonctionnelles, ce qu'il faut savoir

- Obligatoires sous LabVIEW 2 ("**LabVIEW 2 style globals**")
- Technique des **registres non-initialisés** (USR)
- Puissance du concept de **fonction** (VI) (avec la FGV, la fonction "est une donnée")
- Depuis LV3 les **variables globales** "officielles" se substituent avantageusement aux FGV simples
- Comme les variables globales et locales, la FGV est dans LV un des moyens **permettant la communication entre processus parallèles** (autrement dit *cassant le flot de données*)

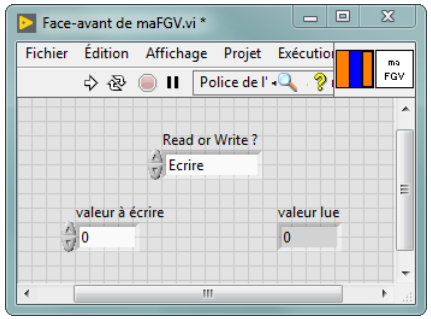
Démonstration

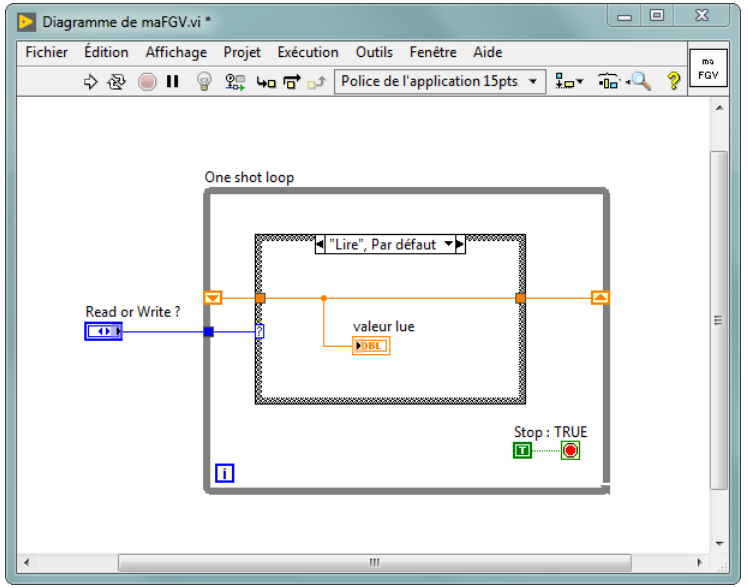


Communication asynchrone entre deux processus ***avec une variable globale classique***

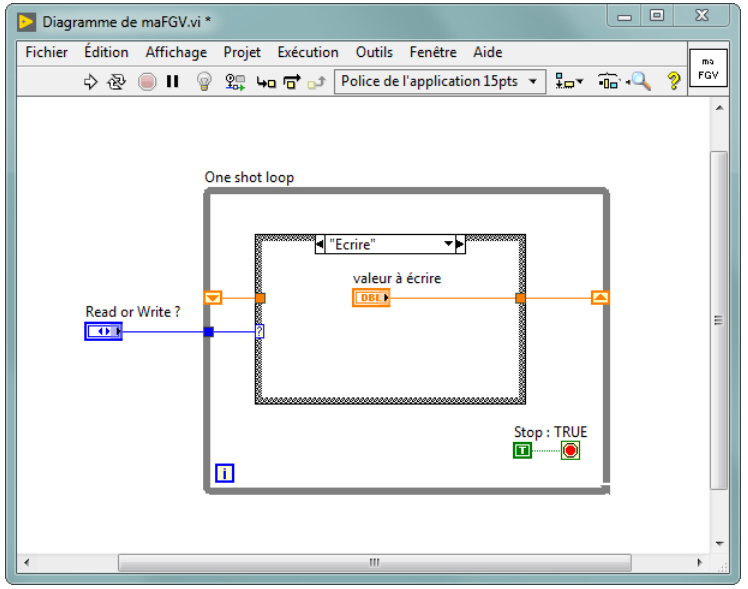


Communication asynchrone entre deux processus ***avec une variable globale fonctionnelle***





De quoi une FGV est-elle faite



Variables Globales Fonctionnelles, ce qu'il faut savoir

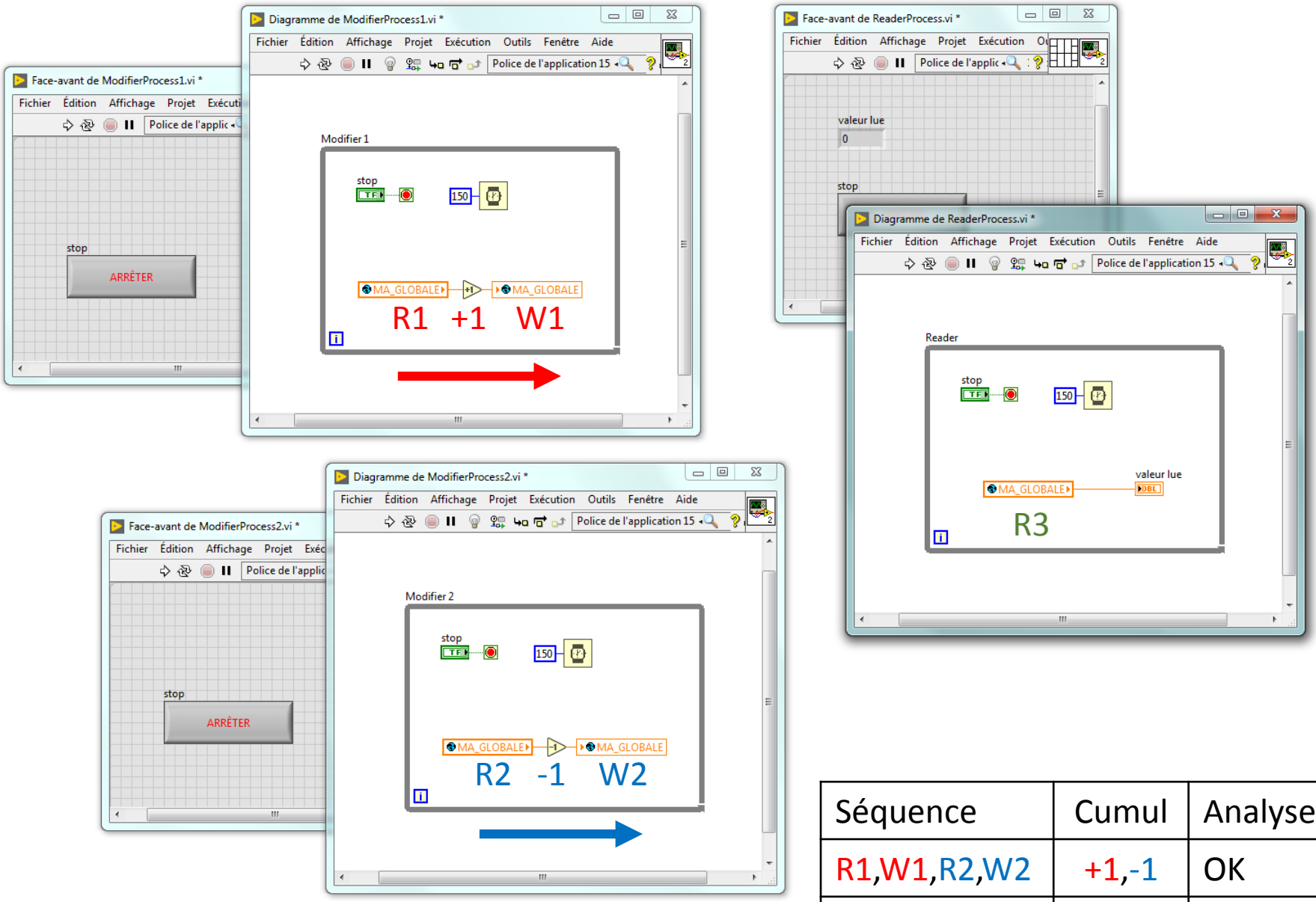
Remarques :

- Une FGV au sens strict n'implémente que deux opérations sur la valeur qu'elle contient : la **lecture** et l'**écriture**
- Le VI d'une FGV doit demeurer **non-réentrant**
  (sinon chaque appel au VI travaille avec son propre clone en mémoire: la variable perd son unicité et son sens)
- Le **nœud de rétroaction** moderne peut remplacer la combinaison boucle à itération unique + registre, un peu artificielle

Parallélisme et Concurrence Critique

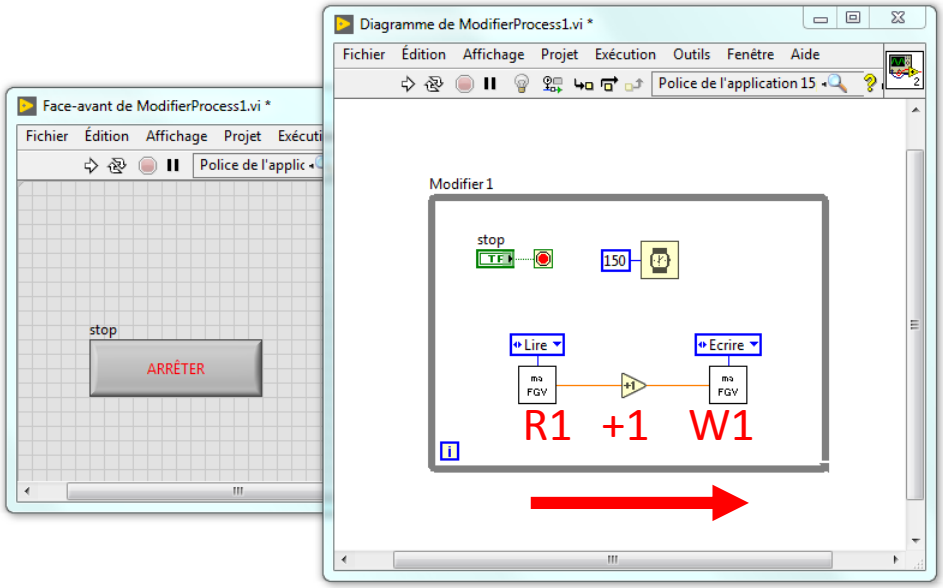
- Implicite dès que deux blocs (en particulier les boucles, mais pas seulement) ne sont pas reliés par au moins un flot de données… **le parallélisme/multiprocessus est omniprésent dans LabVIEW**
- En présence d'une ressource partagée (autrement dit *globale*), les blocs/processus parallèles deviennent **concurrents** vis-à-vis de cette ressource. Par ex. : instrument piloté, fichier du disque dur… ou variable en mémoire (locale ou globale, quelle que soit leur forme)
- Si les actions des processus concurrents ne sont pas protégées, une **situation de concurrence critique** (*race condition*) peut affecter leur résultat et produire un bug difficile à identifier
- Une **solution** consiste à sous-traiter dans un VI unique les opérations qui affectent la ressource partagée en les rendant mutuellement exclusives. Pour le cas d'une variable… **on retrouve là notre FGV**

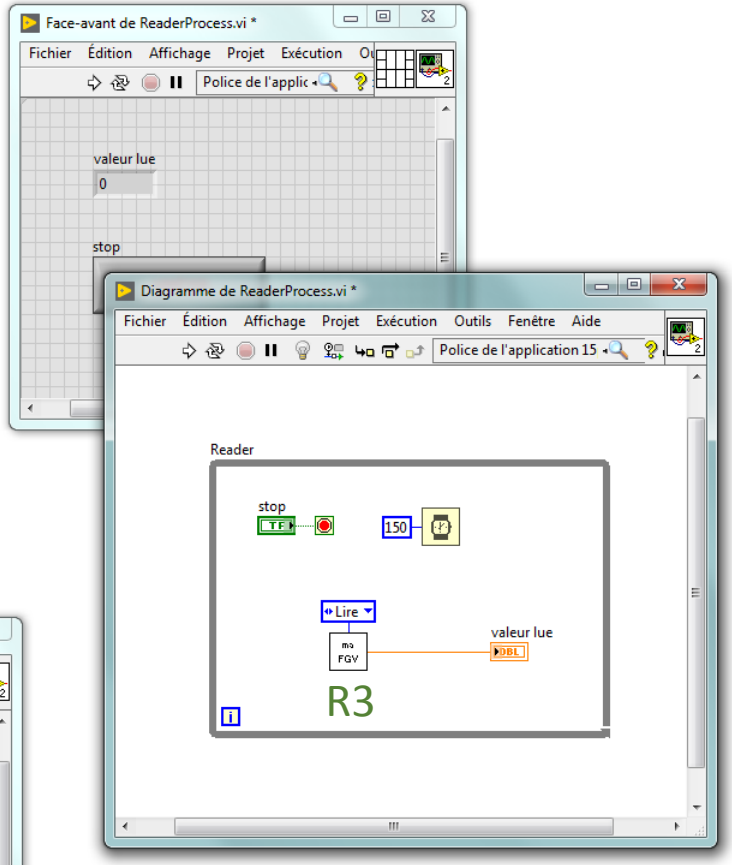
Illustration

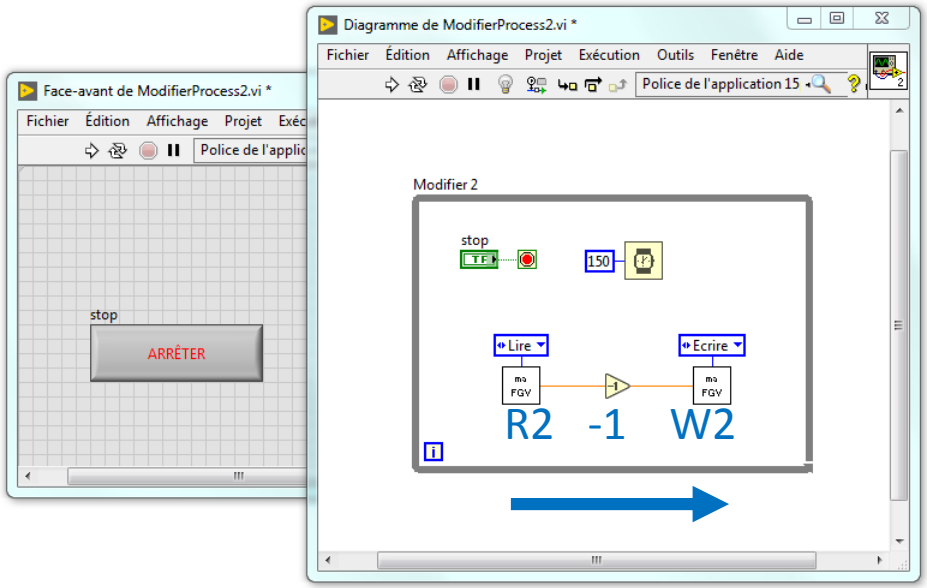


Concurrence critique ! *Si une opération débute alors que l'autre est en cours, une des deux sera perdue*

Séquence	Cumul	Analyse
R1,W1,R2,W2	+1,-1	OK
R1,R2,W1,W2	-1	+1 perdu ?!
R1,R2,W2,W1	+1	-1 perdu ?!



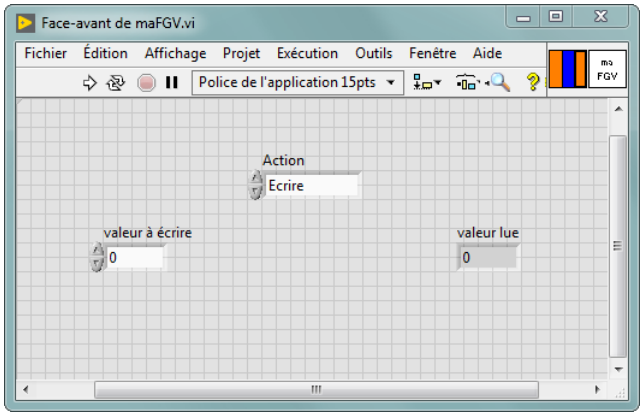


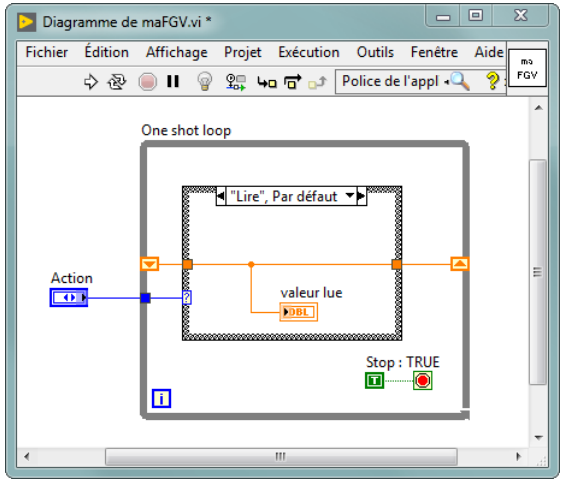


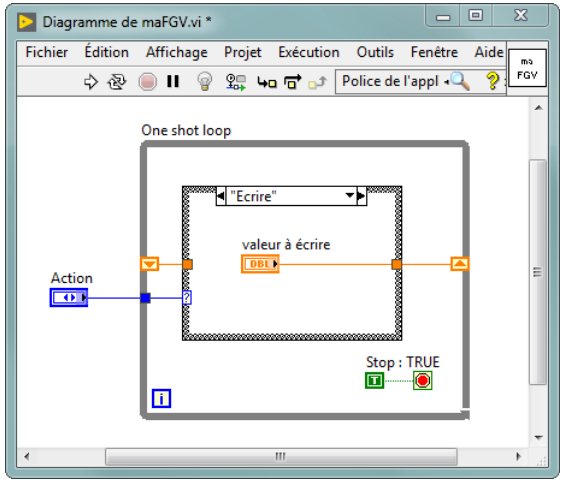
Le problème est le même avec une FGV simple !

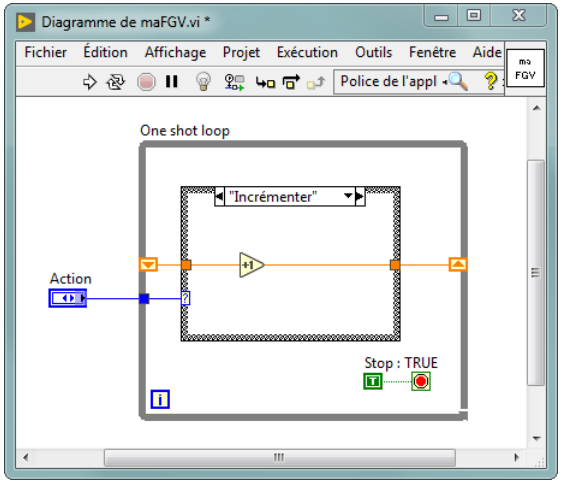
Séquence	Cumul	Analyse
R1,W1,R2,W2	+1,-1	OK
R1,R2,W1,W2	-1	+1 perdu ?!
R1,R2,W2,W1	+1	-1 perdu ?!

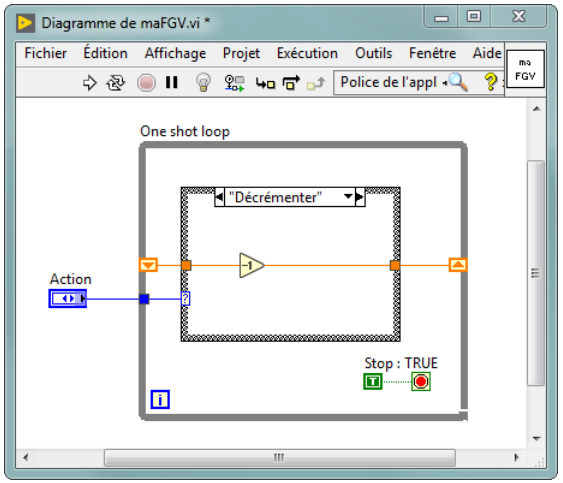
Concurrence critique ! *Si une opération débute alors que l'autre est en cours, une des deux sera perdue*









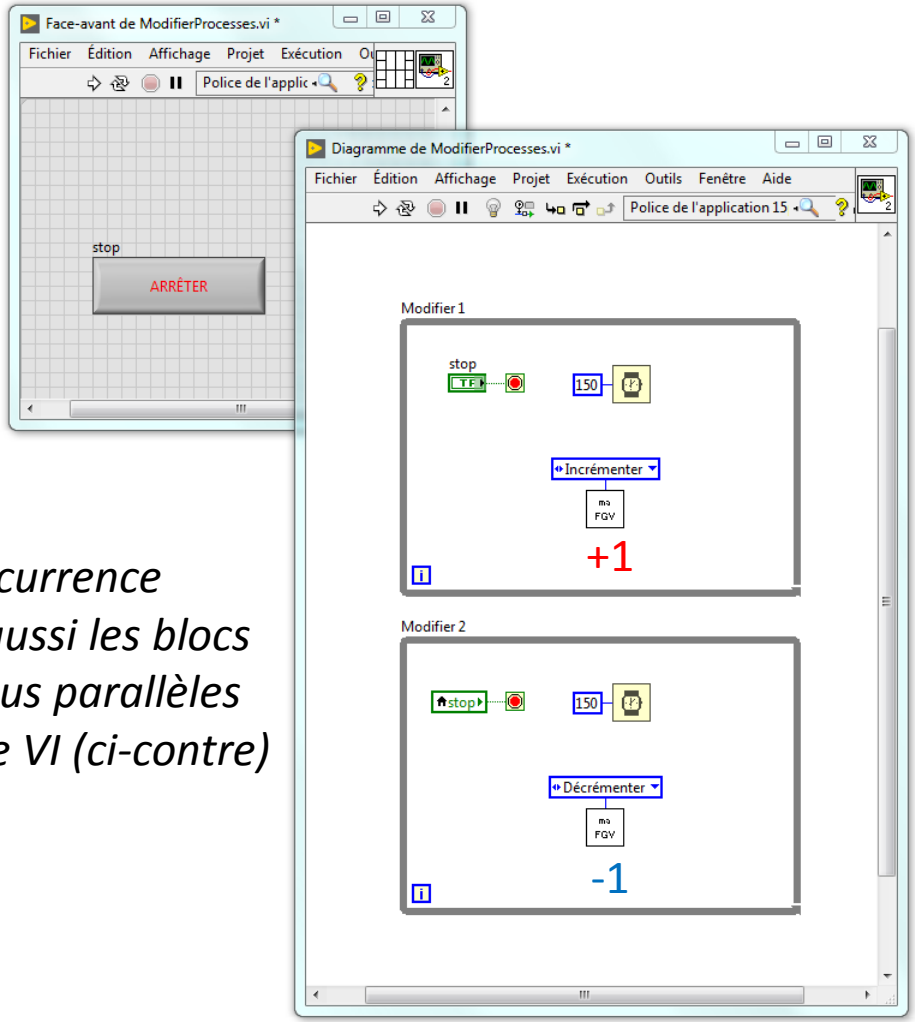


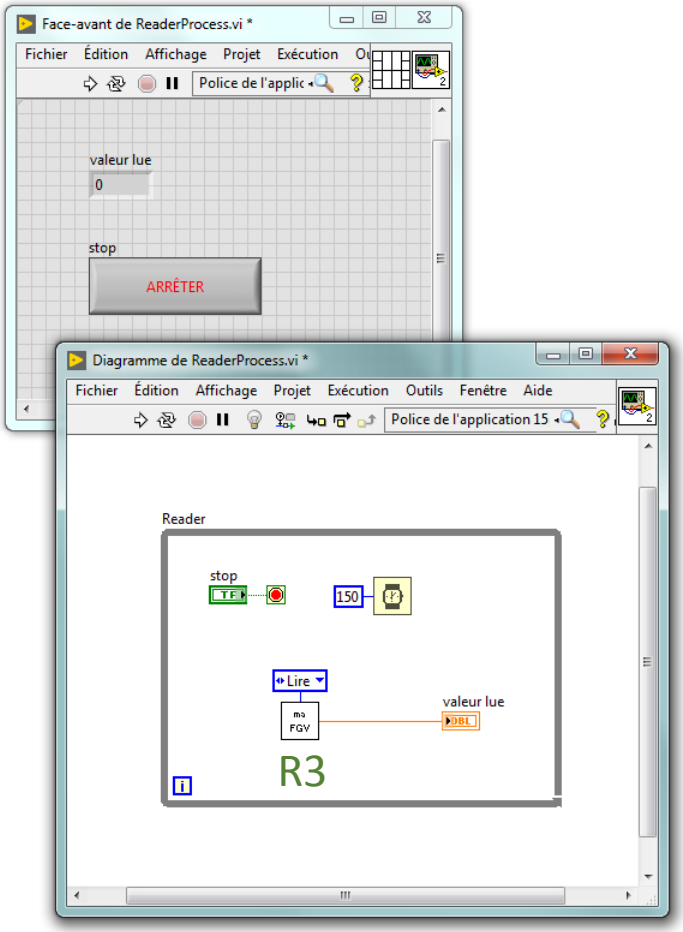
SOLUTION :

*Une FGV améliorée **qui sous-traite les opérations** grâce à des actions supplémentaires.*

Les sections critiques y sont protégées du risque de concurrence critique.

Sur une variable globale classique, cette amélioration n'est pas possible…





Rq : la concurrence concerne aussi les blocs ou processus parallèles d'un même VI (ci-contre)

*Notre FGV améliorée (non-réentrante) exécute les appels **l'un après l'autre**.*

Les opérations sont protégées (insécables).

*La FGV améliorée est **thread safe**.*

Séquence	Cumul	Analyse
(R1,W1),(R2,W2)	+1,-1	Toujours OK

Parallélisme et Concurrence Critique

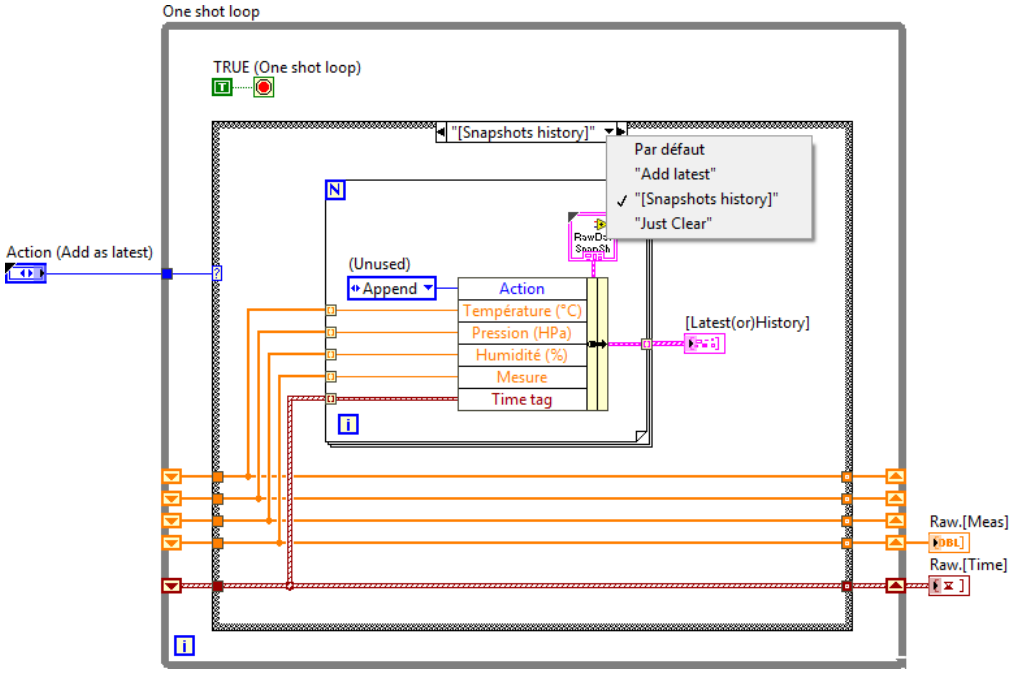
Conclusions :

- Si plusieurs processus concurrents manipulent la valeur d'une variable globale, d'une variable locale ou d'une FGV simple, il faut **vérifier l'absence de concurrence critique** entre eux.
- La FGV peut être **améliorée pour prévenir le risque de concurrence critique**, en sous-traitant pour le compte des appelants les opérations sur la valeur partagée.
- Ce faisant on opère petit à petit un glissement de la FGV vers un autre modèle de conception plus général...

L'Action Engine

- **La FGV est une forme particulière d'AE** dédiée à la seule gestion d'une mémoire interne
- **L'AE est une fonction (VI) pilotée par énuméré pour réaliser une opération parmi une liste d'opérations (procédures) possibles**
- L'AE n'est pas cantonné à la gestion d'une mémoire interne : il peut également s'acquitter de tâches qui n'en ont pas besoin
- Les procédures implémentées dans un AE sont mutuellement exclusives. Comme le VI de l'AE est non-réentrant, les appels issus des processus concurrents sont exécutés *l'un après l'autre*. En centralisant dans un AE les procédures agissant sur une ressource partagée (non seulement une mémoire globale, mais aussi par exemple un instrument) on **protège les appelants du risque de concurrence critique**

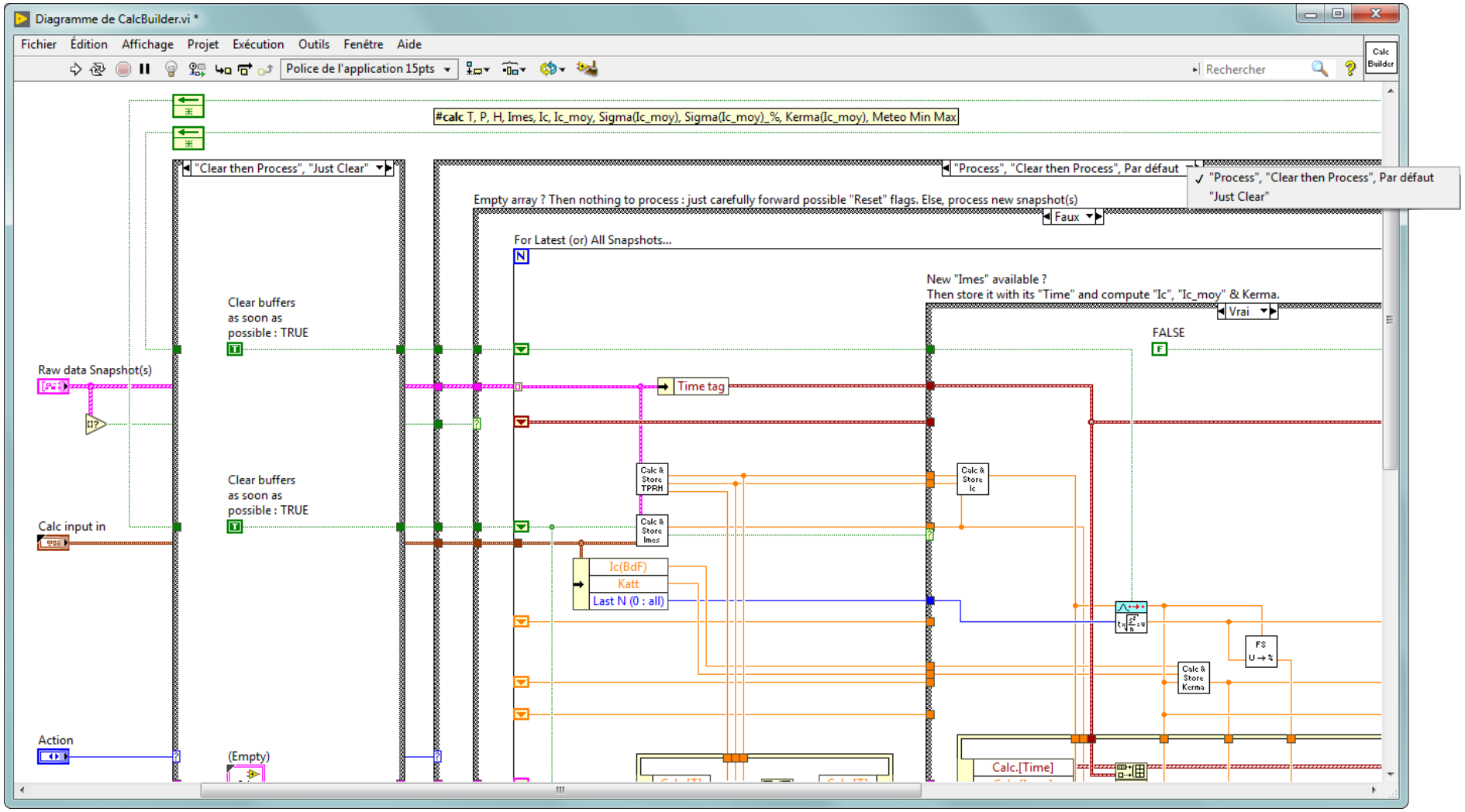
Exemples



AE "Gestionnaire d'historique"

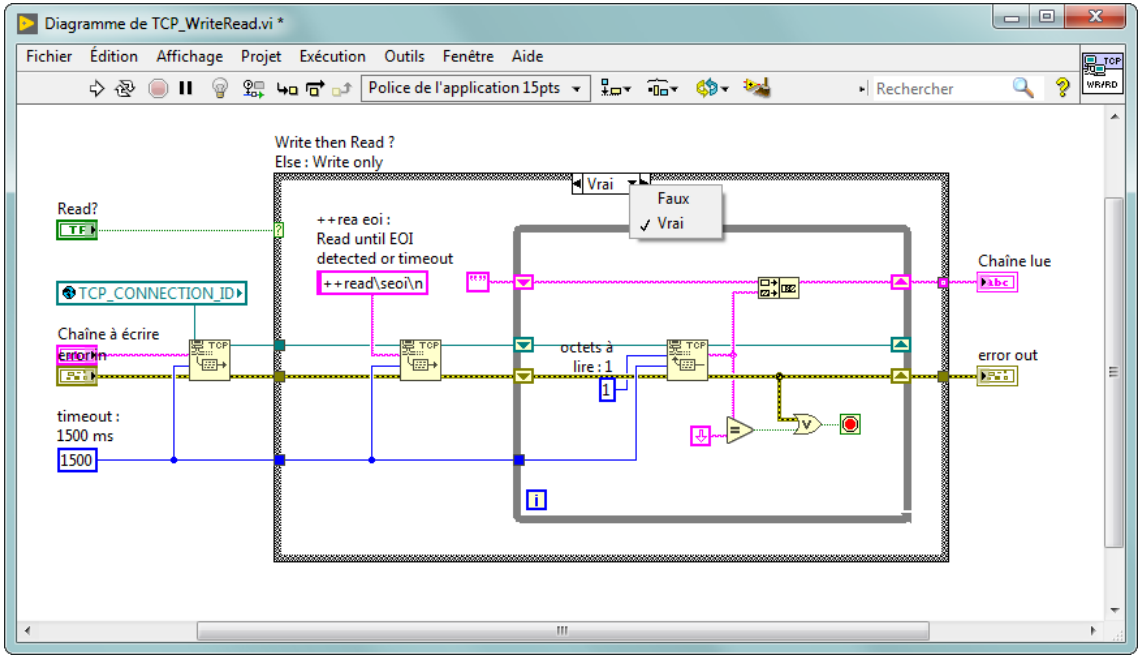
La mémoire gérée par cet AE est répartie dans plusieurs registres pour simplifier les opération sur son contenu et mettre à disposition des extraits partiels.

Rq : plusieurs actions spécifiques → penser à contrôler l'énuméré avec une ***définition de type stricte*** *(sinon il faudra modifier toutes les instances de l'énuméré en cas d'ajout d'une action non prévue au départ…)*



AE "CalcBuilder"

Cet AE complexe effectue des calculs itératifs. Cela implique une mémoire des résultats antérieurs et une possibilité de réinitialisation. Les actions proposées par l'AE combinent des opérations élémentaires (Process, Clear then Process, Just Clear…) pour exécuter chaque combinaison de façon protégée.



AE "WriteRead"

Un cas fréquent de concurrence critique sous LabVIEW concerne la communication avec un instrument : si l'instrument reçoit une seconde requête avant que sa première réponse ait été lue, le premier appelant risque de lire la seconde réponse au lieu de la première.

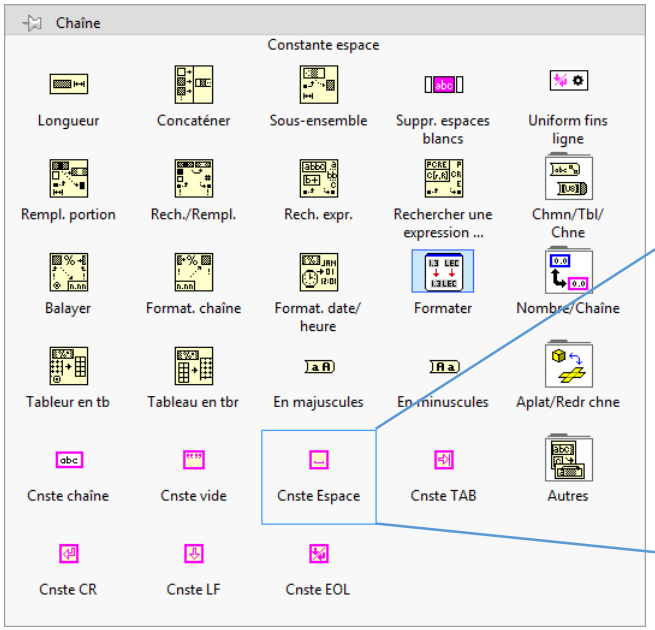
*On réalise une **API instrument thread safe** en confiant à un seul VI l'exclusivité de la communication avec l'instrument (écriture, écriture+lecture).*

*Rq : 2 actions seulement → un **booléen** peut remplacer l'énuméré habituel*

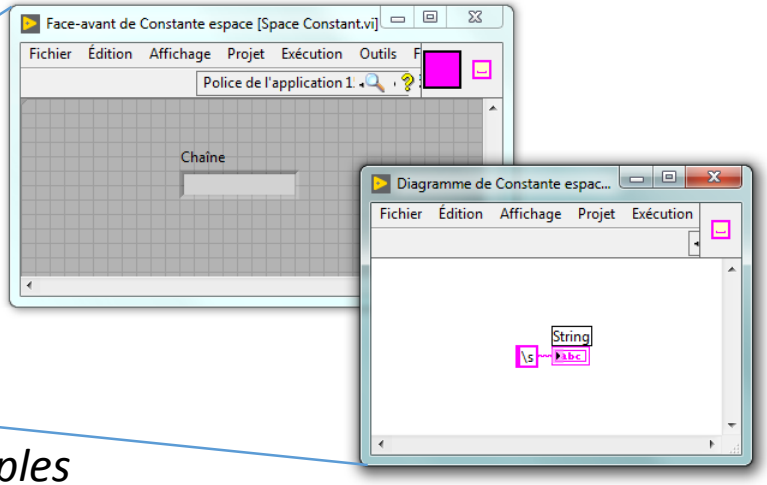
Constantes Globales Fonctionnelles

Sur le modèle de la FGV on réalise facilement une Constante Globale avec une simple fonction

Ce design pattern remplace très avantageusement la "globale avec valeur par défaut" pour toutes sortes de valeurs constantes ou de référence dans un programme.

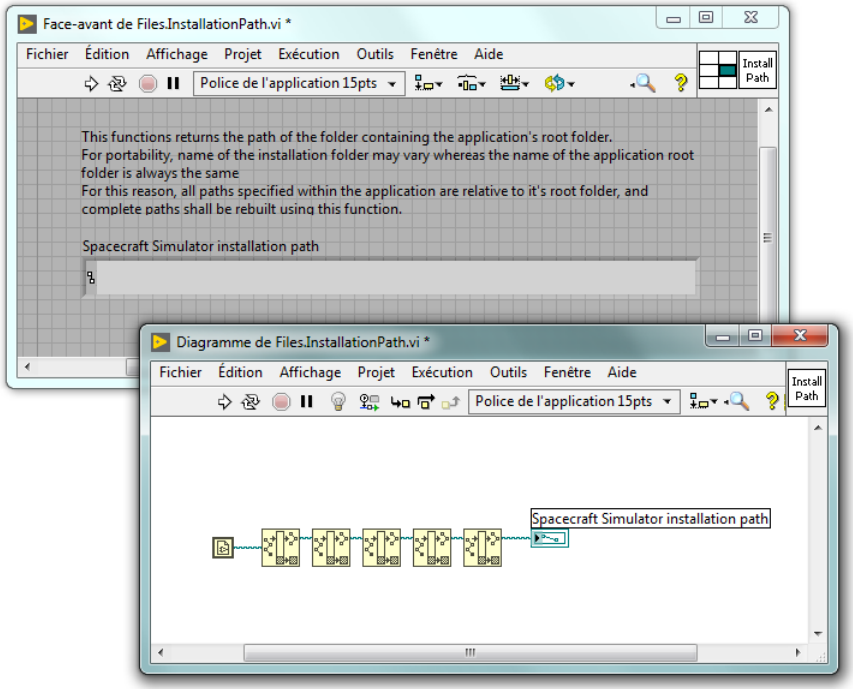


Exemple de la "Constante espace"
dans la palette standard de LabVIEW (!)



Exemples

Constantes Globales Fonctionnelles

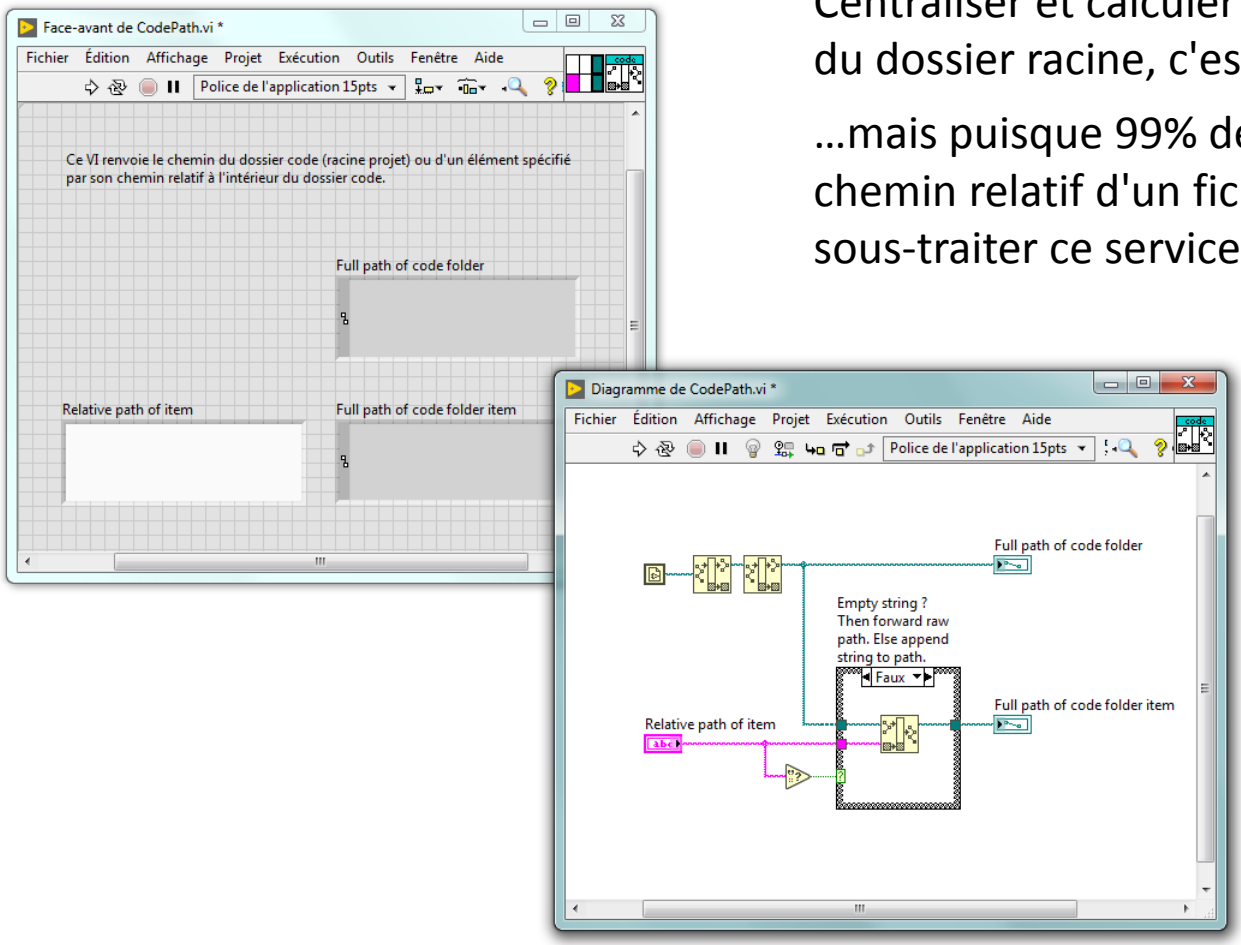


Constante Globale "dynamique"

Une définition centralisée du chemin racine du projet, c'est bien…

…si le chemin se calcule tout seul (dynamiquement) c'est encore mieux : on devient indépendant du dossier d'installation.

Constantes Globales Fonctionnelles



Constante Globale "service compris"

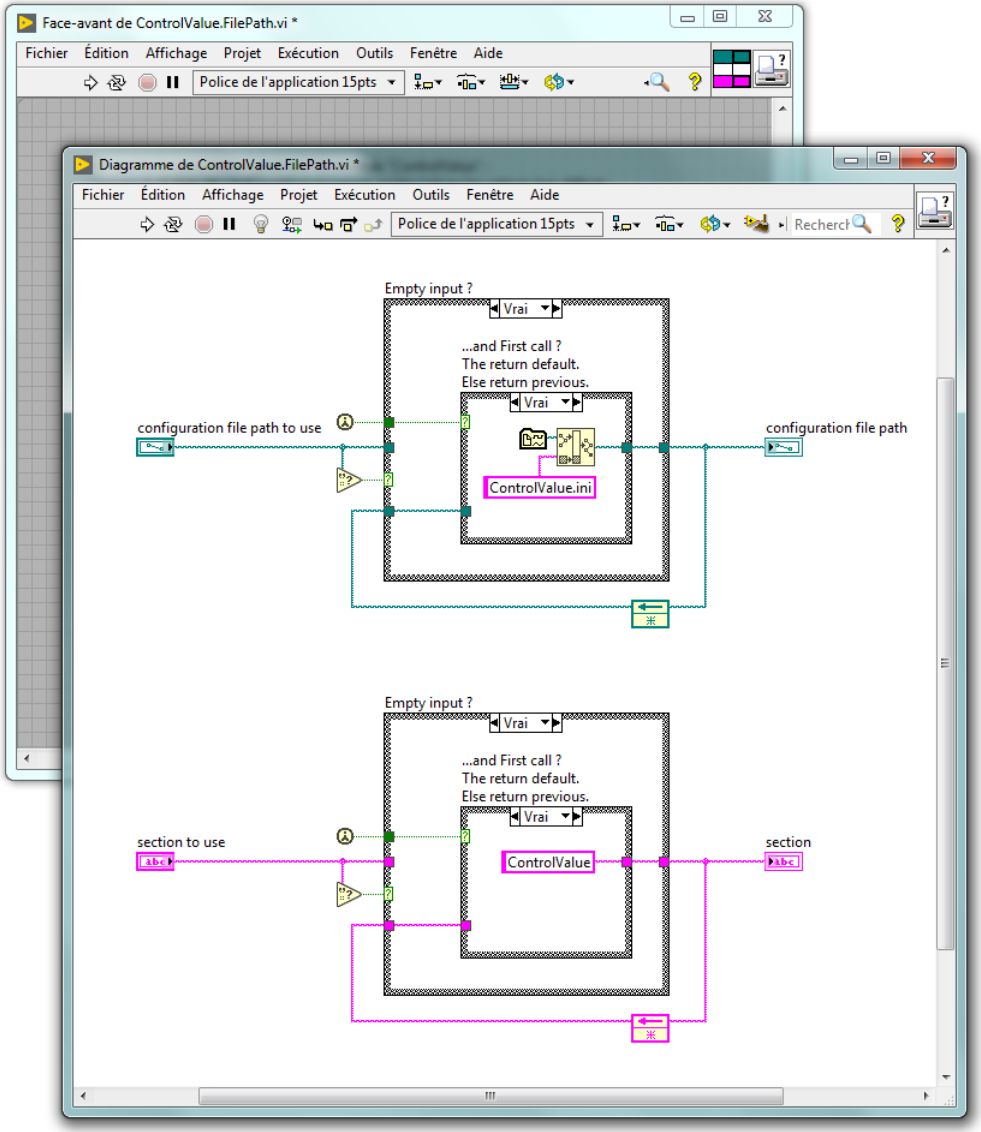
Centraliser et calculer dynamiquement le chemin du dossier racine, c'est bien…

…mais puisque 99% des appelants le combinent au chemin relatif d'un fichier cible, pourquoi ne pas sous-traiter ce service ?

Astuce : **Détection d'un paramètre entrant facultatif**

On affecte au contrôle du paramètre entrant une valeur par défaut distincte des valeurs d'usage. Si le code détecte cette valeur, c'est que le paramètre n'est pas fourni par l'appelant (ici : chaîne vide).

Constantes Globales Fonctionnelles



Constante Globale "auto-initialisée"

Si le calcul de sa valeur est couteux en temps on peut utiliser la fonction standard "Premier appel ?" pour l'initialiser au premier appel, et stocker le résultat pour le fournir aux appels suivants.

Rq : La **détection du paramètre entrant facultatif** déclenche ici une redéfinition de la valeur à utiliser : la frontière entre constante, fonction et variable est modulable...

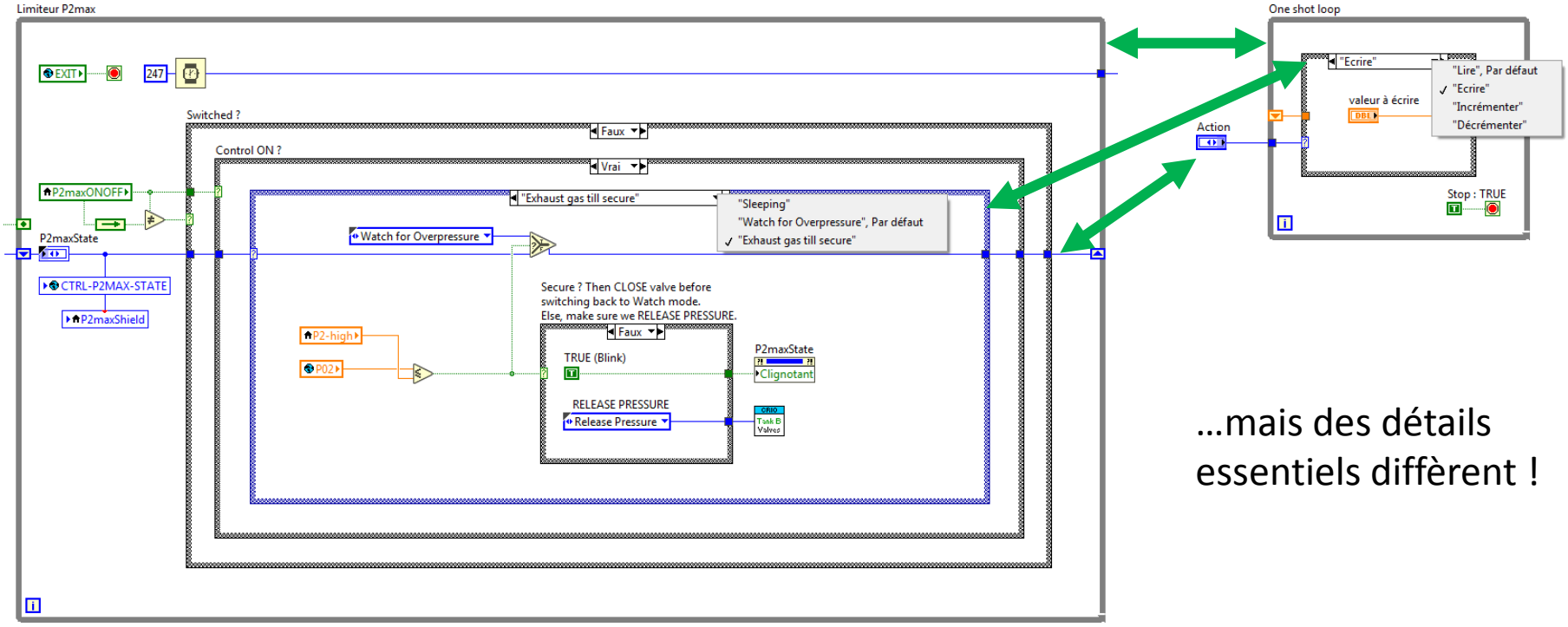
Différence entre FGV/AE et Machine à Etats ?

Les implémentations LabVIEW se ressemblent fortement

- Enuméré
- Boucle
- Structure conditionnelle pilotée par l'énuméré

FSM

FGV/AE



…mais des détails essentiels diffèrent !

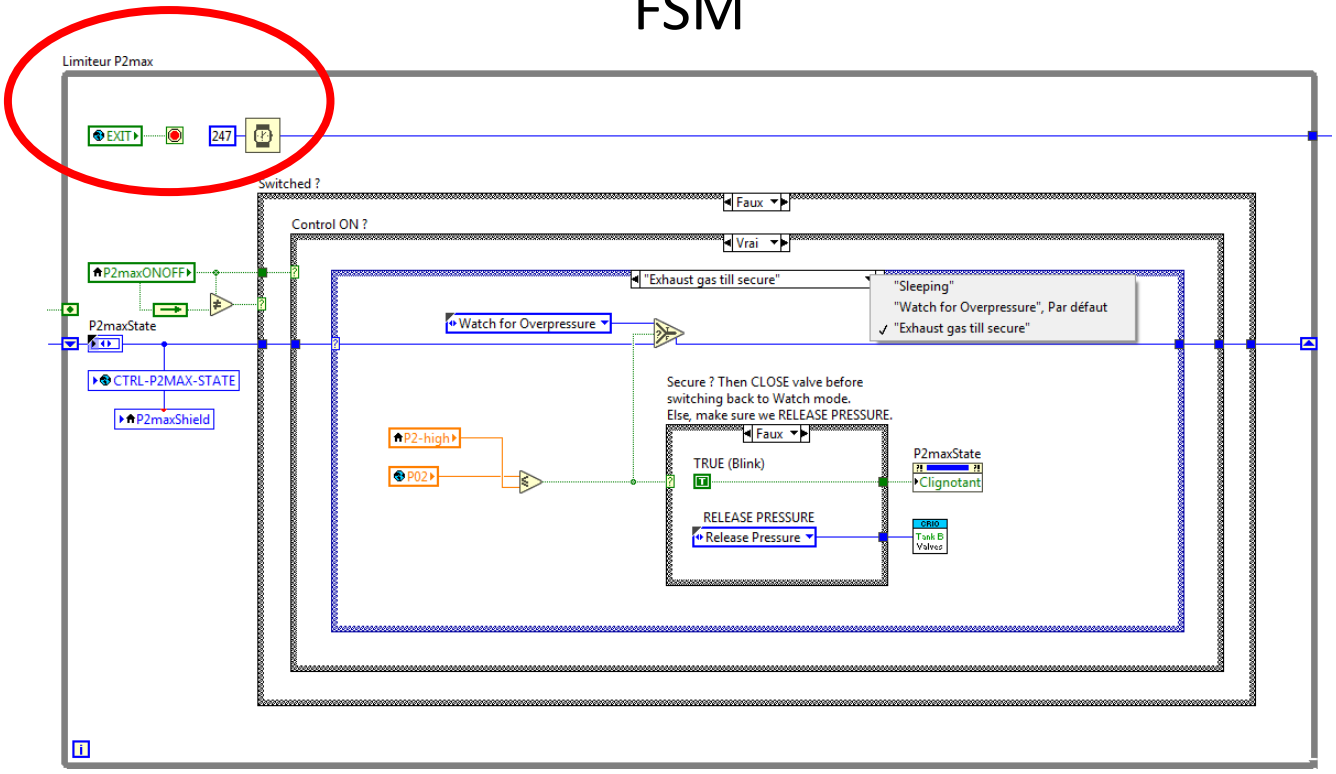
Différence entre FGV/AE et Machine à Etats ?

Caractère itératif ou non

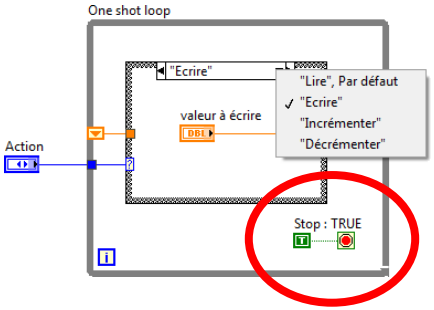
- La FGV/AE est procédurale. Elle agit à la demande, s'exécute et se termine.
- La FSM est un processus itératif, souvent long, susceptible d'enchainer des étapes dans un ordre variable. Les états sont caractérisés par des *attentes*. Ils sont modifiés par des *événements* extérieurs.

Itérations multiples

FSM



FGV/AE



Itération unique

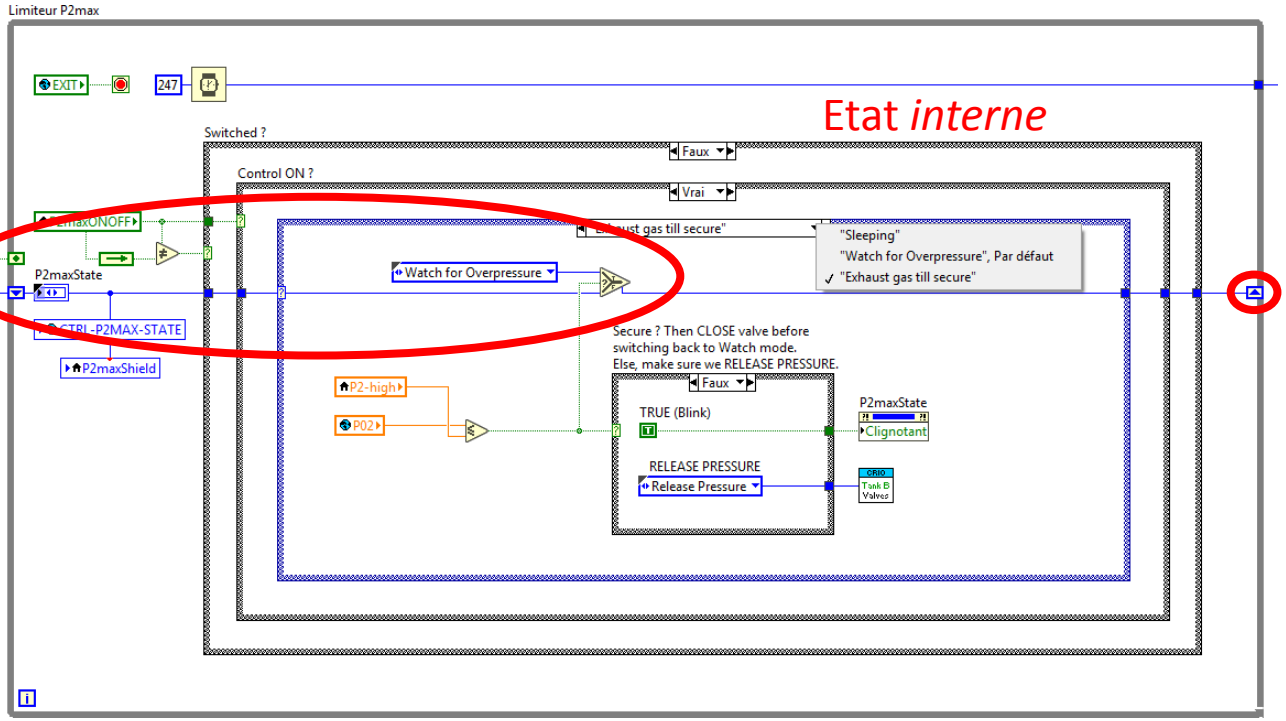
Rq : "fausse boucle", facultative depuis l'introduction du *nœud de rétroaction*

Différence entre FGV/AE et Machine à Etats ?

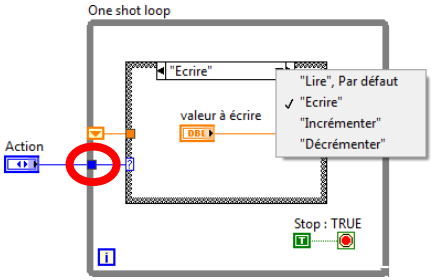
Enuméré interne ou passé en paramètre

- Pour la FGV/AE l'énuméré (action) est un paramètre spécifié par l'appelant.
- Pour la FSM l'énuméré (état) est une variable interne déterminé par chaque itération en vue de l'itération suivante.

FSM



FGV/AE



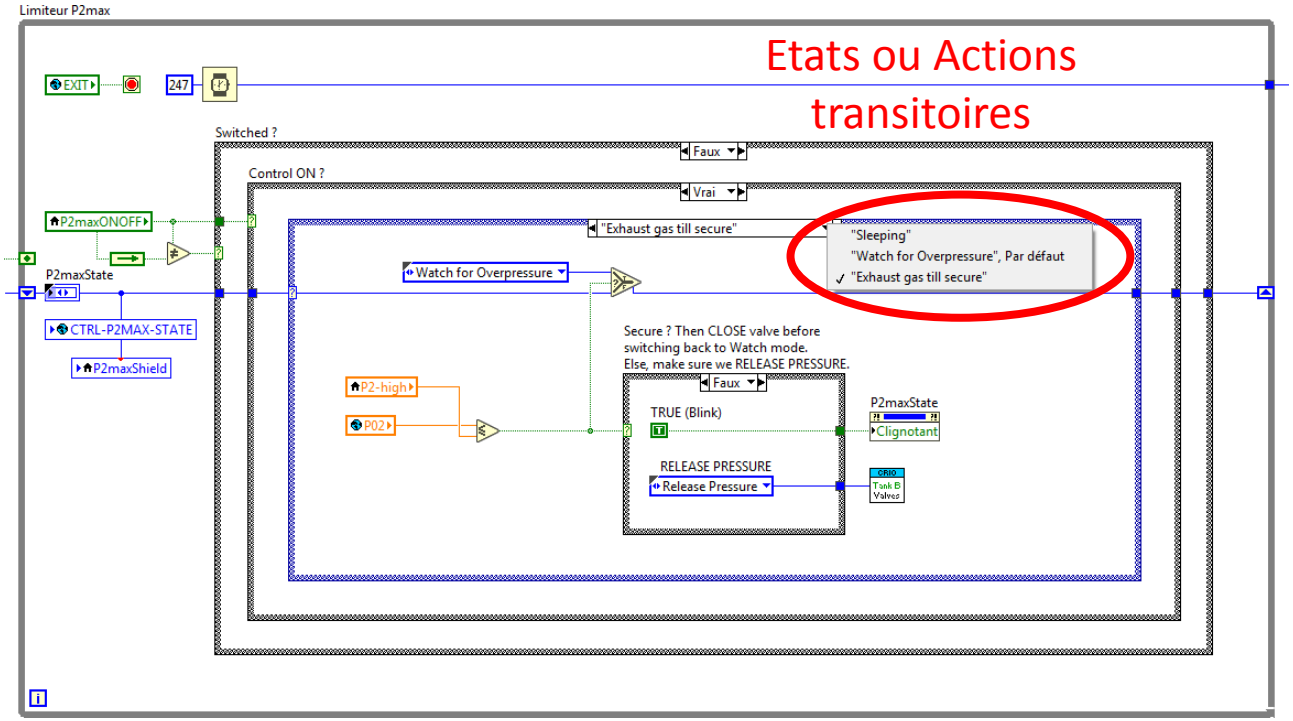
Action *spécifiée par l'appelant*

Différence entre FGV/AE et Machine à Etats ?

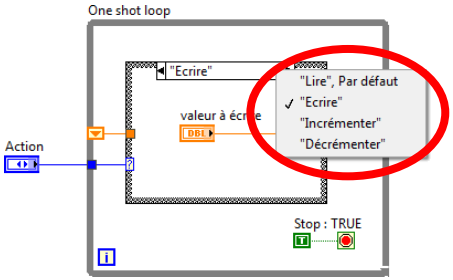
Enuméré Action ou Etat

- Pour la FGV/AE l'énuméré désigne toujours des ACTIONS.
- Pour la FSM l'énuméré désigne des ETATS ou des ACTIONS TRANSITOIRES.

FSM



FGV/AE



Actions

Que reste-t-il aux Variables Globales ?

Considérer le Poids de la donnée

- **La variable globale est très performante pour les valeurs simples, de petite taille** : scalaires, petits tableaux, types simples.
- Mais la lecture d'une variable globale génère forcément une **copie en mémoire** de sa valeur (le flot de donnée qui en sort). Pour une donnée volumineuse (très grand tableau…) la pénalité est double :
  - budget mémoire x 2
  - traitement ralenti par l'allocation
- **La FGV/AE est plus adaptée pour une donnée volumineuse** (très grand tableau…) si on prend soin d'y sous-traiter les opérations "en place" :
  - sans faire sortir la donnée stockée dans la FGV
  - sans générer de copies évitables en mémoire

Que reste-t-il aux Variables Globales ?

Architecture et risque de Concurrence Critique

- Une variable globale qui présente **un seul point d'accès en écriture** est immunisée contre les situations de concurrence critique. Peu importe le nombre de lectures.
- L'acronyme **WORM (Write Once, Read Many)** renvoie à ce principe.
- Plusieurs architectures remarquables s'y rattachent et fonctionnent très bien avec une variable globale *sans provoquer de concurrence critique*.

Exemples

Que reste-t-il aux Variables Globales ?

Pattern 1 : **Constante Globale**

Une seule écriture à l'initialisation, tous peuvent lire

C'est le pattern WORM au sens strict.

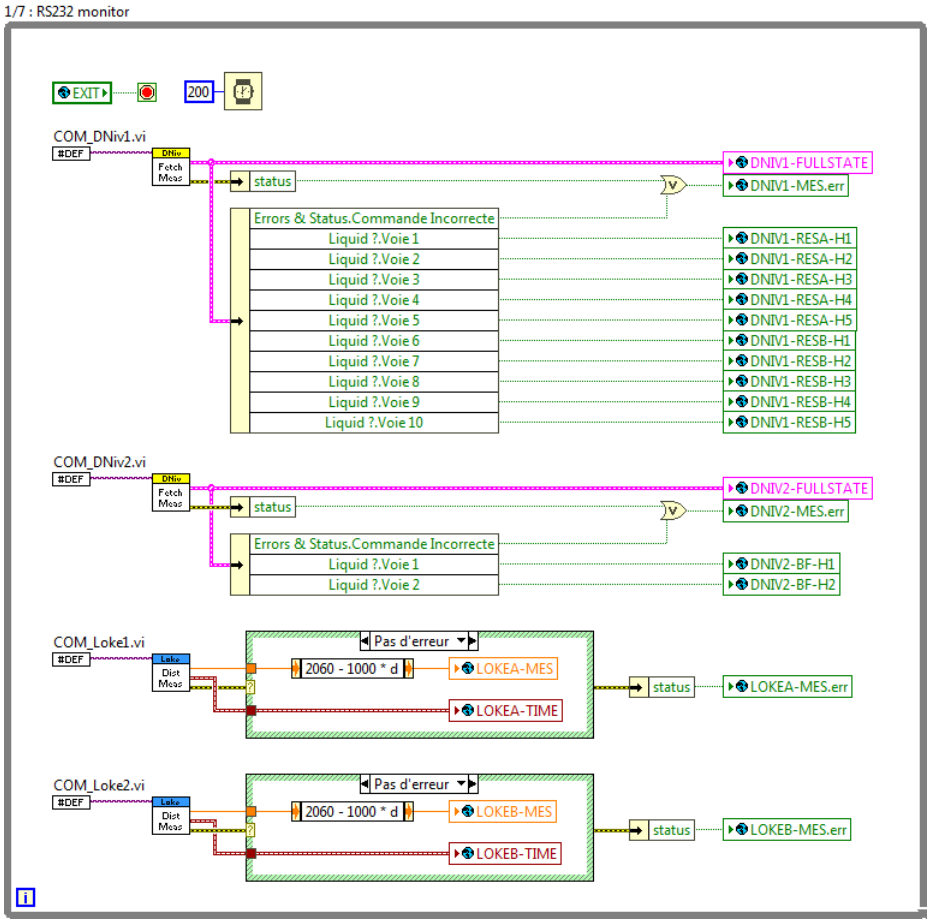
Rq : On peut aussi s'appuyer sur la valeur défaut de la globale (dans ce cas il n'y a *plus aucun accès en écriture* à cette variable dans tout le programme).

Recommandation : La technique des Constantes Globales Fonctionnelles permet d'exclure plus clairement toute possibilité d'une modification en cours d'exécution.

Que reste-t-il aux Variables Globales ?

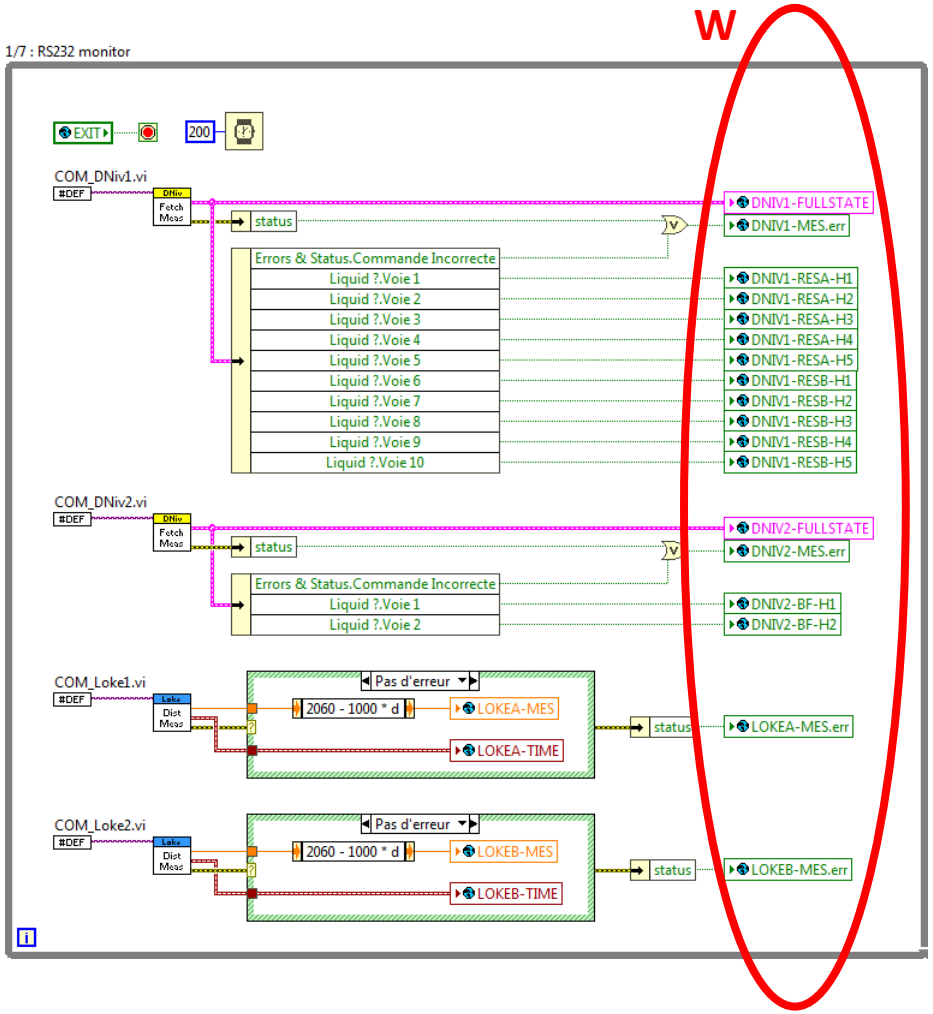
Pattern 2 : **Global Monitoring DataBase**

Un seul processus écrit, tous peuvent lire



Que reste-t-il aux Variables Globales ?

Pattern 2 : **Global Monitoring DataBase**

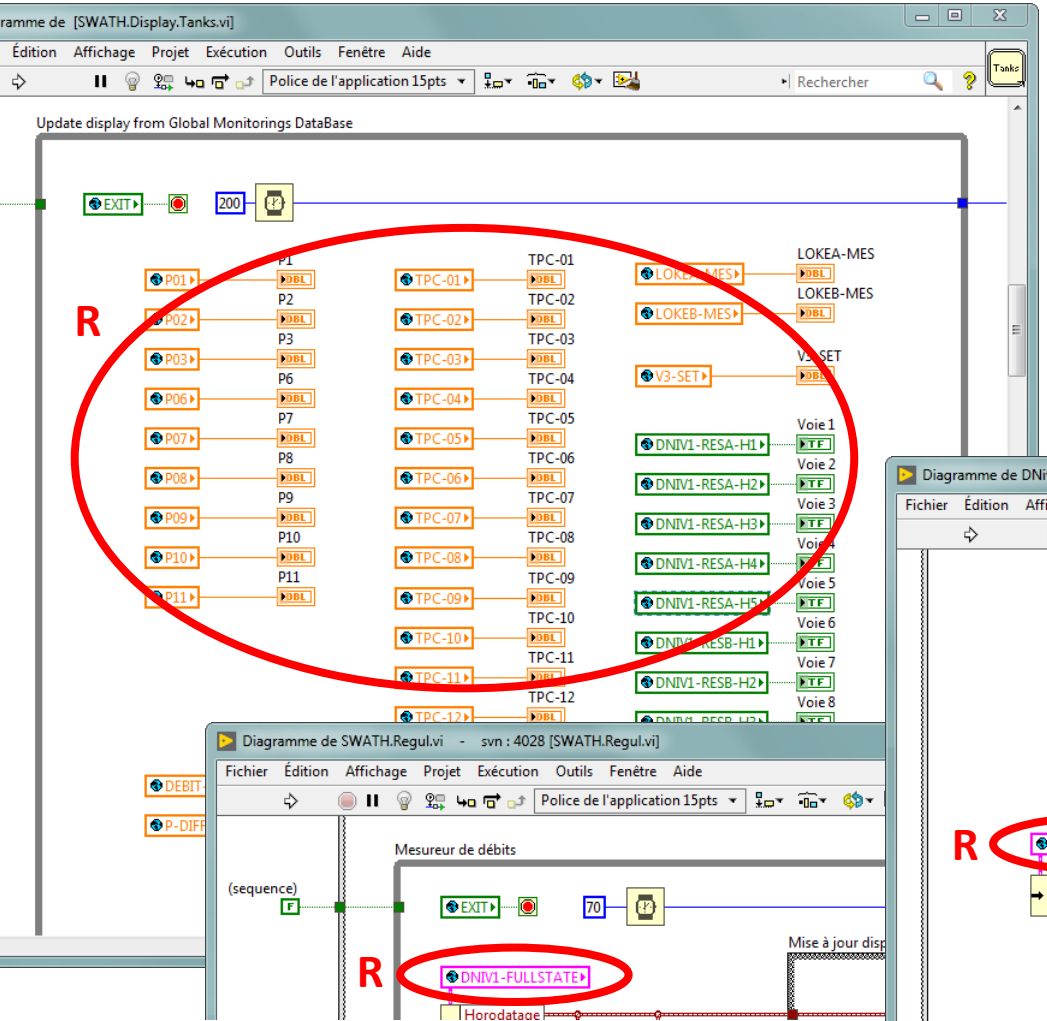


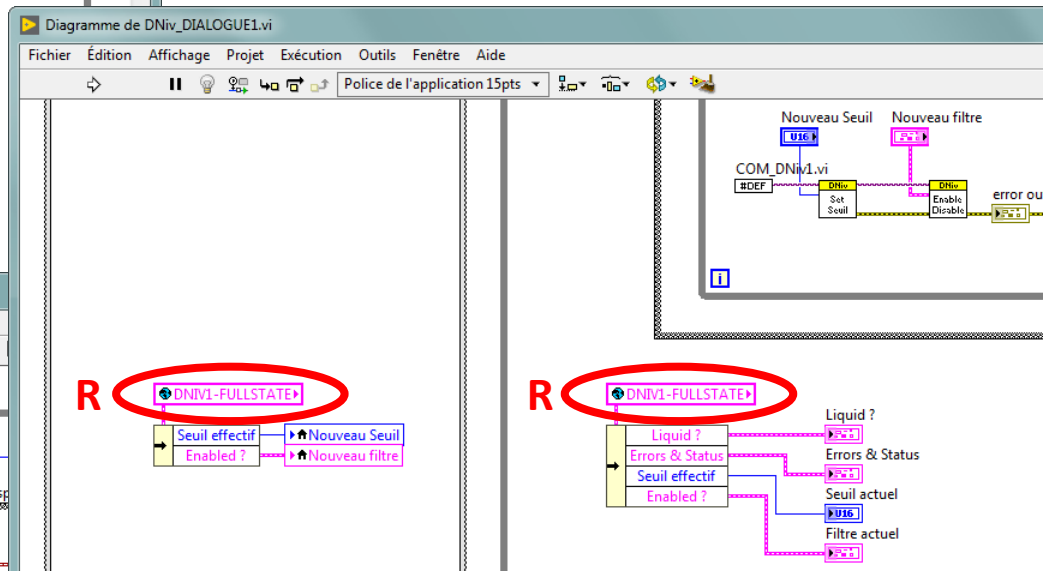
Le processus *moteur de mise à jour* relève en continu les grandeurs d'ambiance (température, pression…).

Il publie les dernières mesures en date dans une variable globale (*base de données*).

Que reste-t-il aux Variables Globales ?

Pattern 2 : **Global Monitoring DataBase**

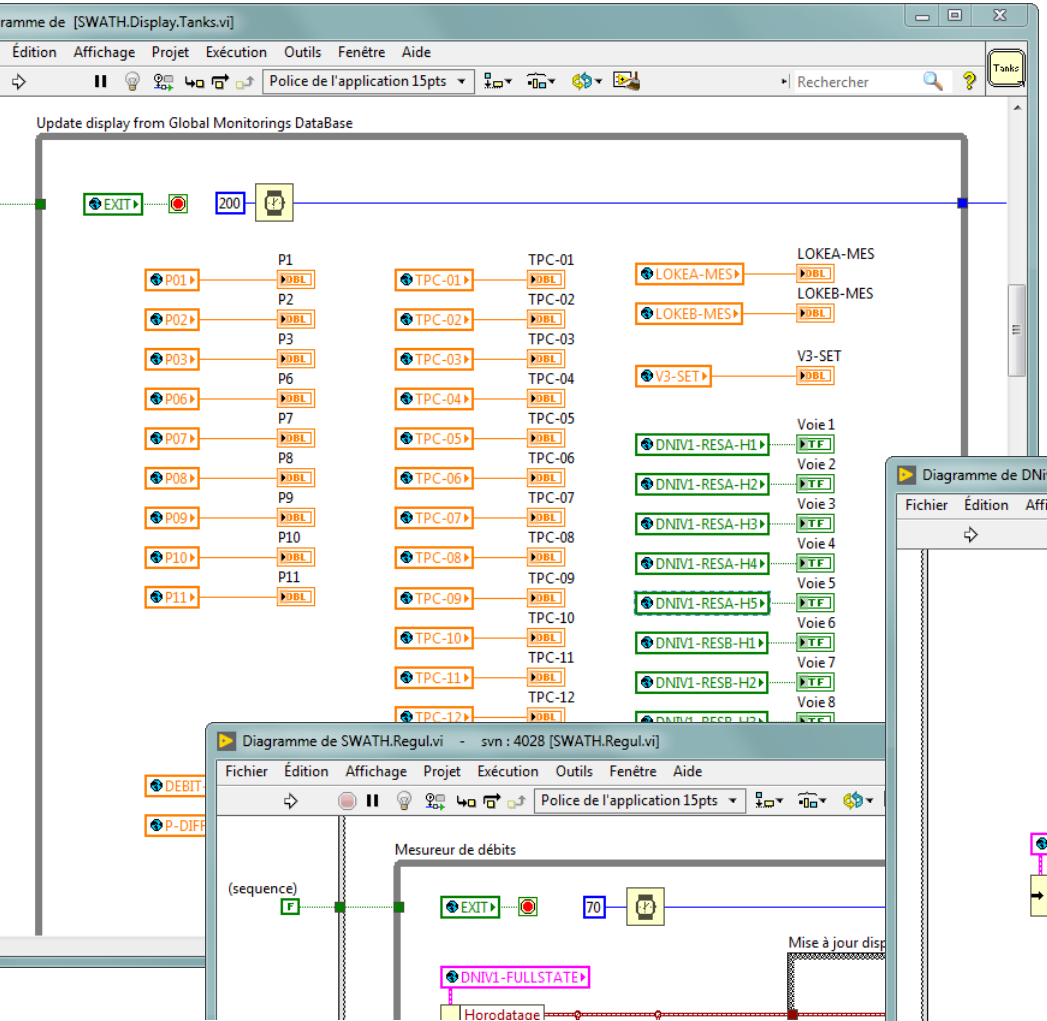


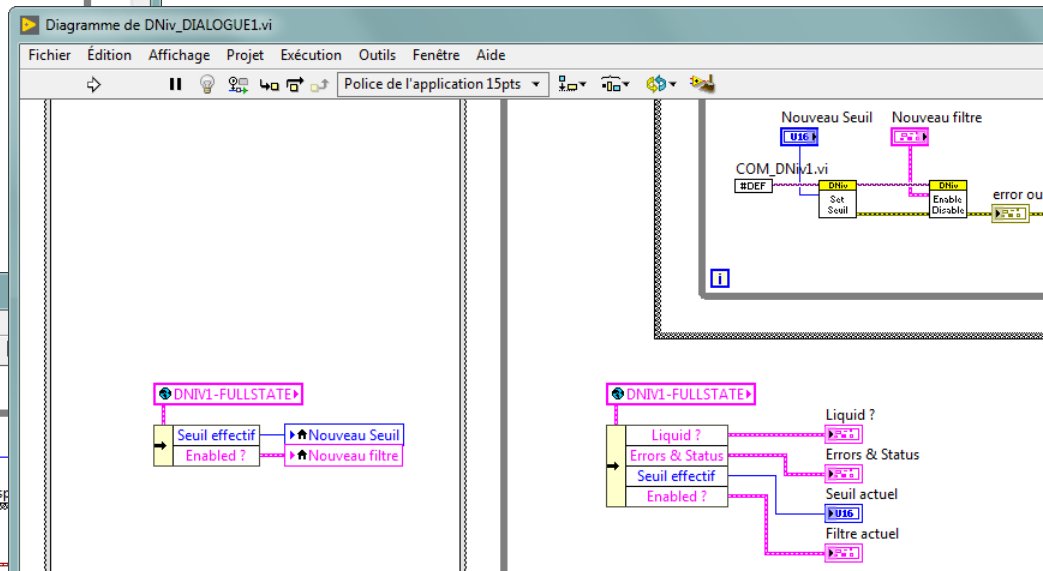


Les processus consommateurs accèdent en lecture seulement aux globales qui les intéressent.

Que reste-t-il aux Variables Globales ?

Pattern 2 : **Global Monitoring DataBase**





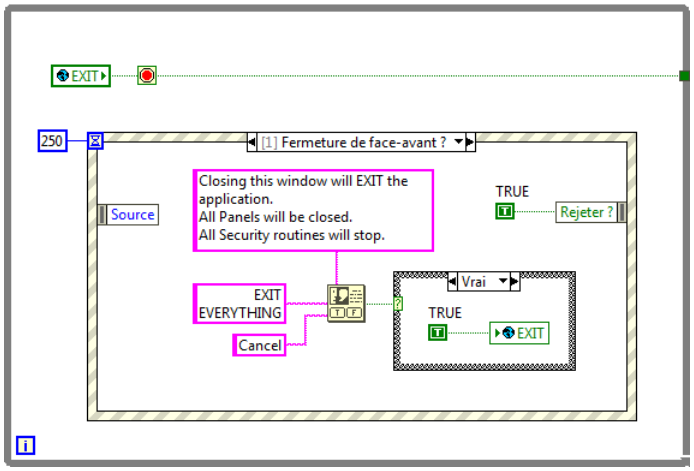
Rq : C'est un producteur-consommateurs asynchrone (*lectures et écritures surviennent indépendamment à tout instant*) tolérant les pertes (*les lecteurs acceptent de manquer une valeur ou de lire plusieurs fois la même*).

Que reste-t-il aux Variables Globales ?

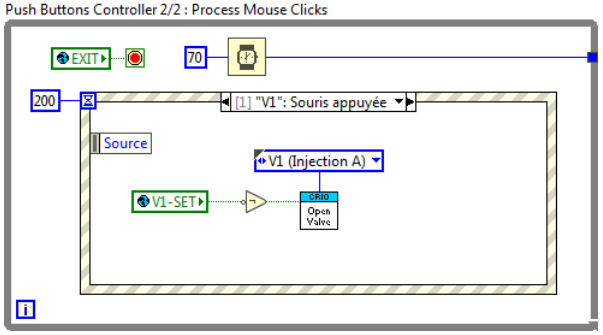
Pattern 3 : **Ordre d'arrêt général**

Un seul changement peut survenir (EXIT : FAUX→VRAI)
Tous les processus peuvent écrire VRAI, tous peuvent lire

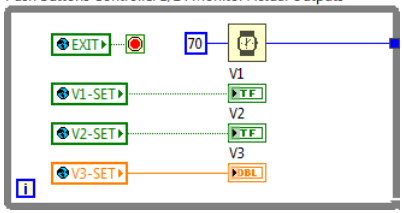
EXIT Handler



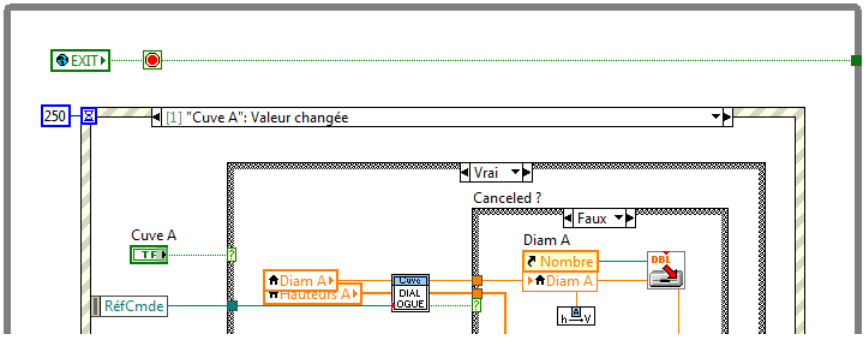
Push Buttons Controller 2/2 : Process Mouse Clicks



Push Buttons Controller 1/2 : Monitor Actual Outputs



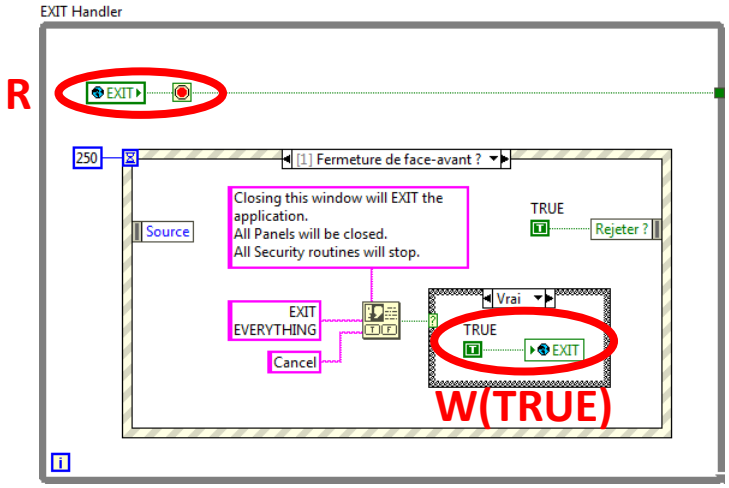
User actions

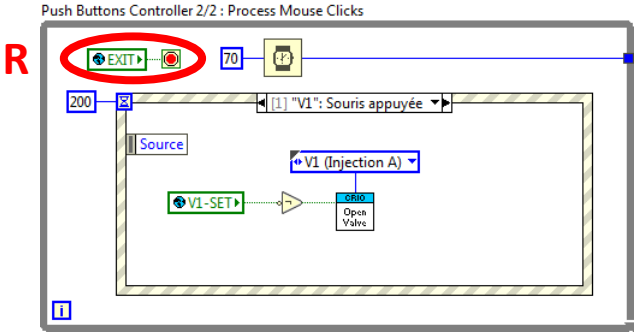


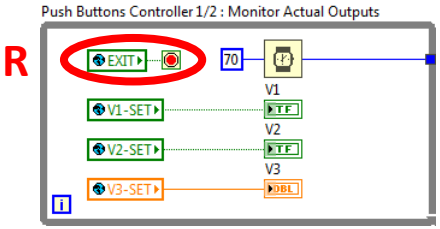
Que reste-t-il aux Variables Globales ?

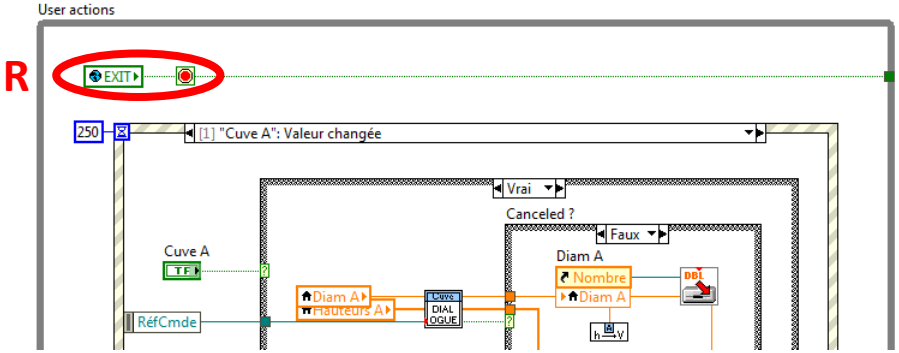
Pattern 3 : **Ordre d'arrêt général**

Tous les processus ont le droit d'envoyer l'ordre d'arrêt du programme en changeant la globale EXIT à VRAI.
Tous les processus scrutent la globale EXIT et se terminent si elle devient VRAI.





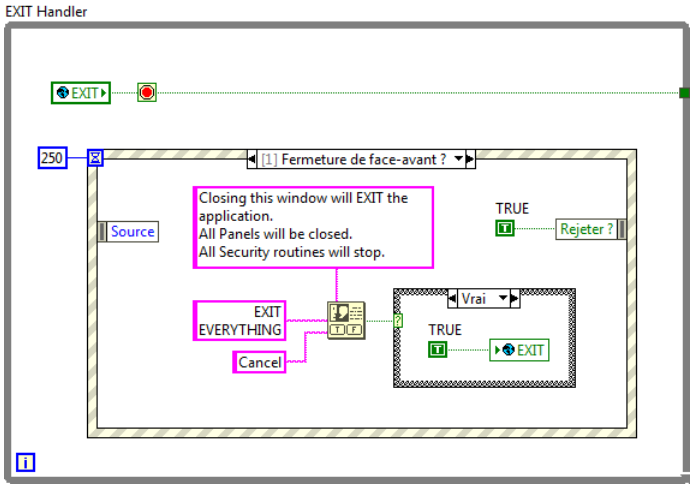


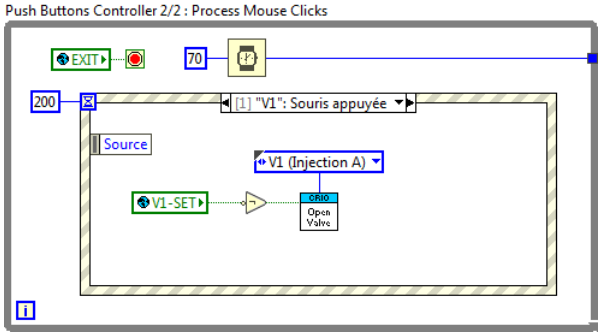


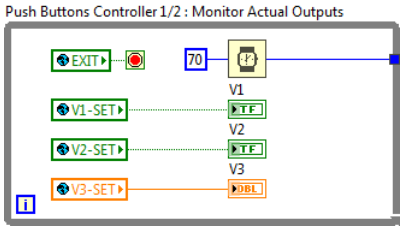
Que reste-t-il aux Variables Globales ?

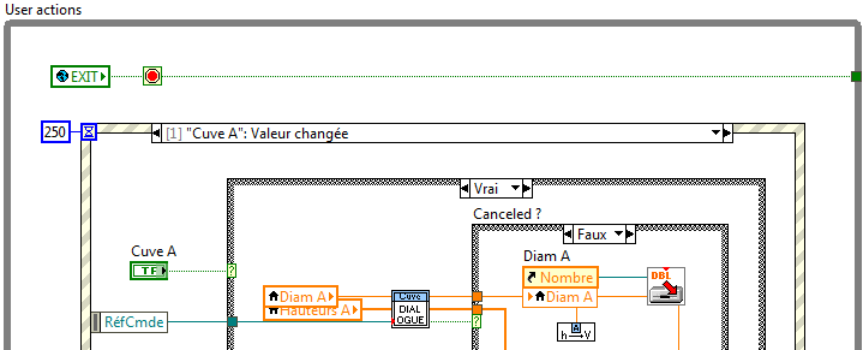
Pattern 3 : **Ordre d'arrêt général**

Rq : Une Variable Globale Fonctionnelle améliorée (Lire, Ecrire VRAI) permet d'exclure plus clairement la possibilité d'une écriture à FAUX.









Que reste-t-il aux Variables Globales ?

Facilité de mise en œuvre et Ergonomie

- Les variables globales sont **facile à mettre en œuvre et tout à fait performantes**. Les FGV/AE sont **plus complexes et longues à construire** (énumérés, définitions de type, USR, non-réentrance…).
- *Dans les deux cas* une bonne compréhension de la concurrence critique et de ses remèdes est prérequise.
- Une globale autorise toujours des accès en lecture ou en écriture. **Une FGV/AE peut restreindre plus efficacement sa propre utilisation**.
- Dans LabVIEW on peut **regrouper les globales par familles** dans différents fichiers, et **sélectionner la variable voulue dans une liste** des variables du fichier lors de la programmation. Ces facilités ergonomiques peuvent être appréciables et sont absentes avec une FGV/AE.

Références

Everything You Ever Wanted To Know About Functional Global Variables [Nancy Hollenback (NI), 2013]
http://vishots.com/wp-content/uploads/2013/12/ts2147hollenback.pdf

Utilisation mesurée des variables globales et locales [Aide LabVIEW 2017]
http://zone.ni.com/reference/fr-XX/help/371361P-0114/lvconcepts/using_local_and_global/

Variables globales fonctionnelles [Aide LabVIEW 2017]
http://zone.ni.com/reference/fr-XX/help/371361P-0114/lvconcepts/suggestions_for_exec/#Functional_Global_Variables

Thread Safety [Wikipedia]
https://en.wikipedia.org/wiki/Thread_safety

Situation de compétition [Wikipedia]
https://fr.wikipedia.org/wiki/Situation_de_comp%C3%A9tition

Variable globale [Wikipedia]
https://fr.wikipedia.org/wiki/Variable_globale

Automate fini (*Finite-State Machine* : Machine à Etats) [Wikipedia]
https://fr.wikipedia.org/wiki/Automate_fini

I Like Global Variables [Darren Nattinger (NI), 2010]
https://labviewartisan.blogspot.fr/2010/06/i-like-global-variables.html

Table des exemples