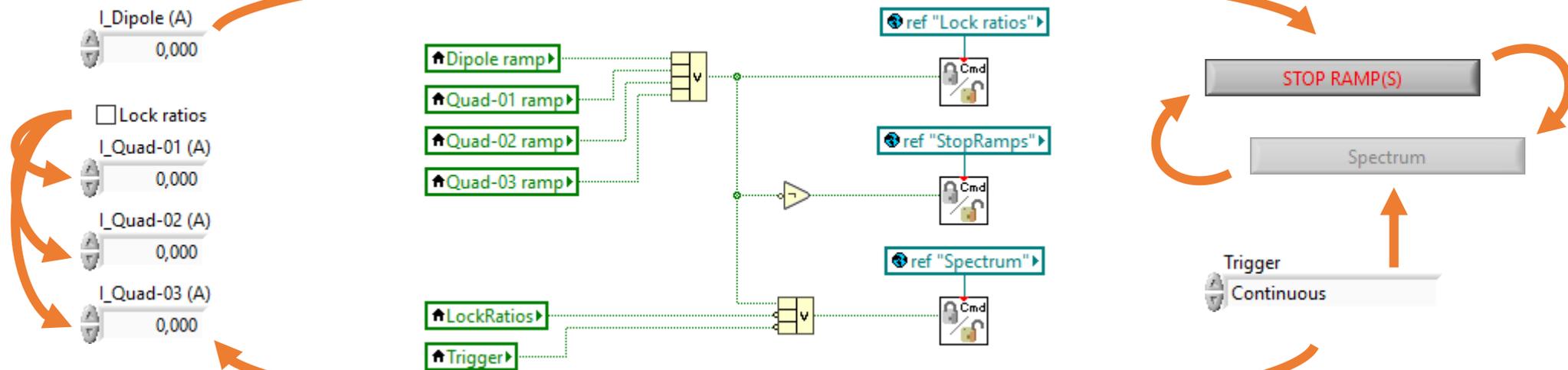


GUI Interlocks :

un dispositif performant pour contrôler les verrouillages mutuels dans LabVIEW



Vocabulaire

Interlocks \Leftrightarrow Verrouillages mutuels
(de sécurité)

Vocabulaire

Dans le domaine ferroviaire :

Enclenchement

(...) ensemble d'appareils de signalisation qui matérialise physiquement, dans la zone d'action d'un [poste d'aiguillage](#) (...) une incompatibilité de manœuvre entre différents organes de commande d'appareils de voie ou de signaux dans le but de n'autoriser le passage d'un mouvement sur les différents [appareils de voie](#) que lorsque toutes les conditions de sécurité nécessaires à ce mouvement sont réalisées.

<https://fr.wikipedia.org/wiki/Enclenchement>

<https://en.wikipedia.org/wiki/Interlocking>

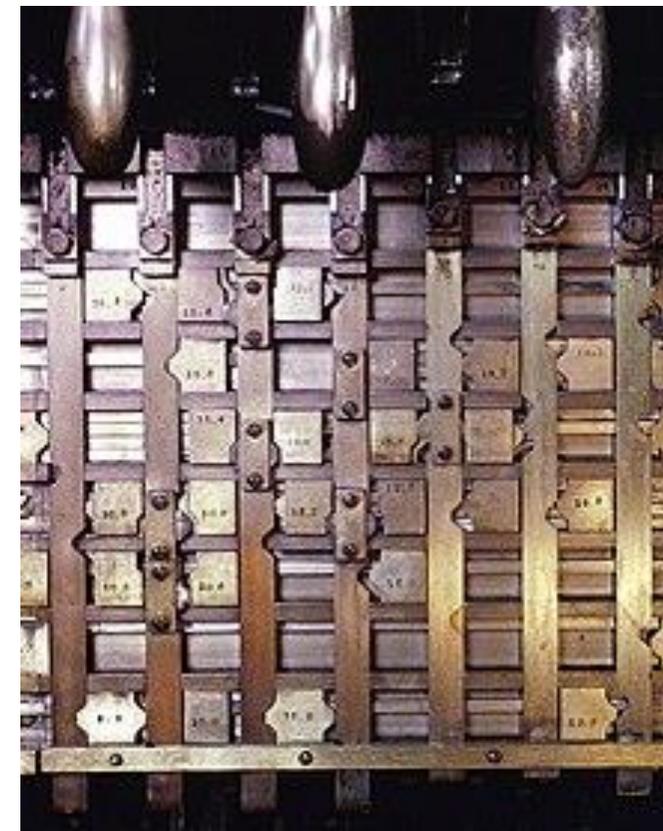
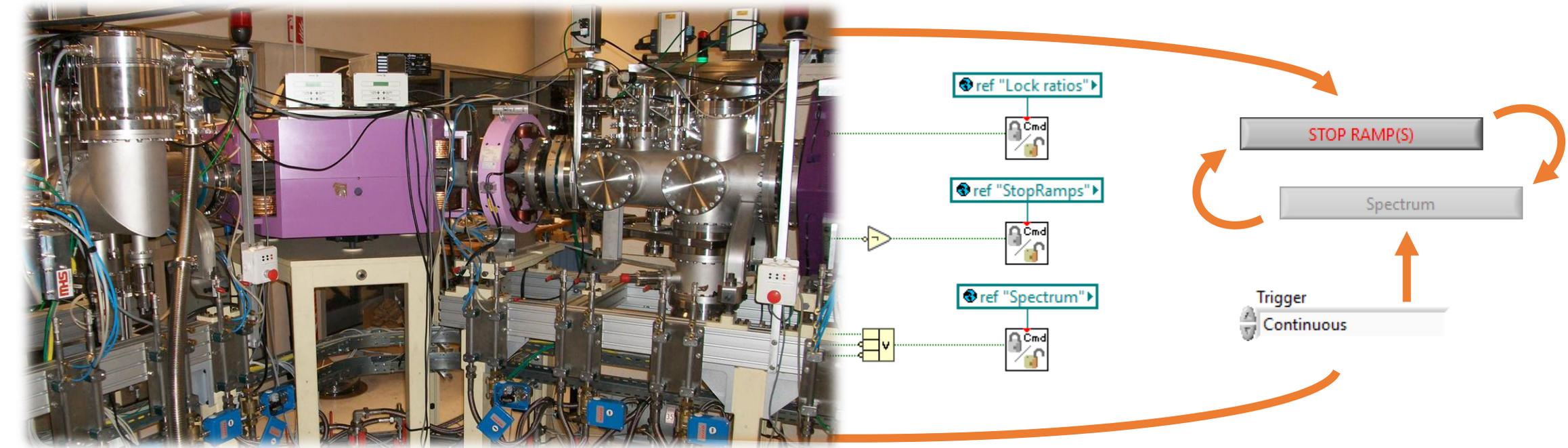


Table d'enclenchements

Problématique

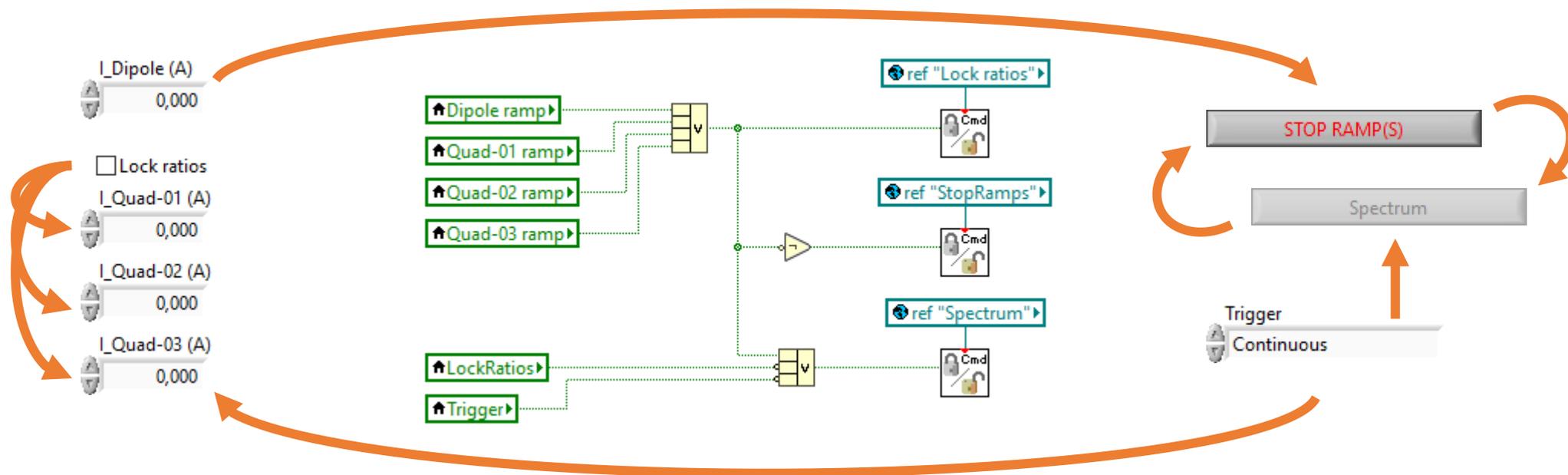
Dans le domaine logiciel... et spécialement le contrôle-commande !



Sécurisation logicielle de l'installation + Sécurisation de l'interface utilisateur
*règles de sécurité, contrôle de l'accessibilité des commandes, onglets, contrôles
masqués ou désactivés+grisés...*

Problématique

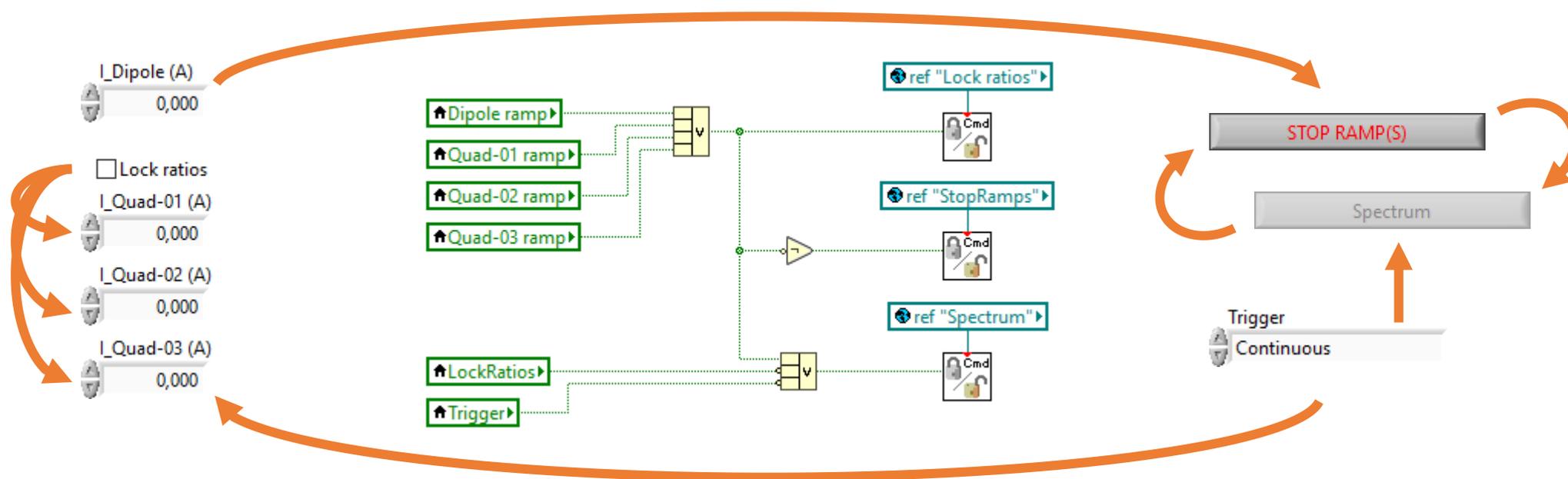
Quel modèle pour implémenter les règles de sécurité ? en LabVIEW ? en général ?



Moyens "passifs" ? Masquage, grisage ? Où placer le code qui gère les verrouillages ? Distribué au plus proche de ses objets ou centralisé ? Quand évaluer les règles logiques ? Comment limiter le couplage avec les autres facettes du logiciel ?...

*** GUIInterlocks ***

Un design pattern efficace ; des VIs modèles à adapter dans votre appli LabVIEW.



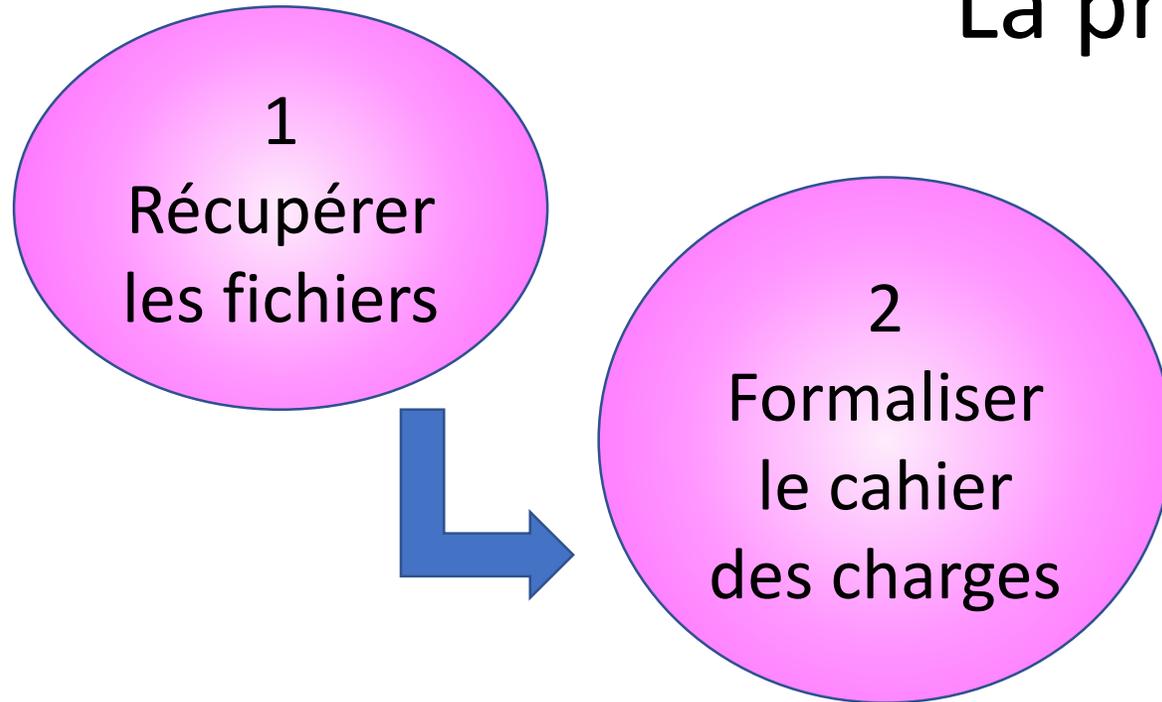
<https://gitlab.in2p3.fr/oz-reuse/labview/ozr-GUIInterlocks>

La procédure

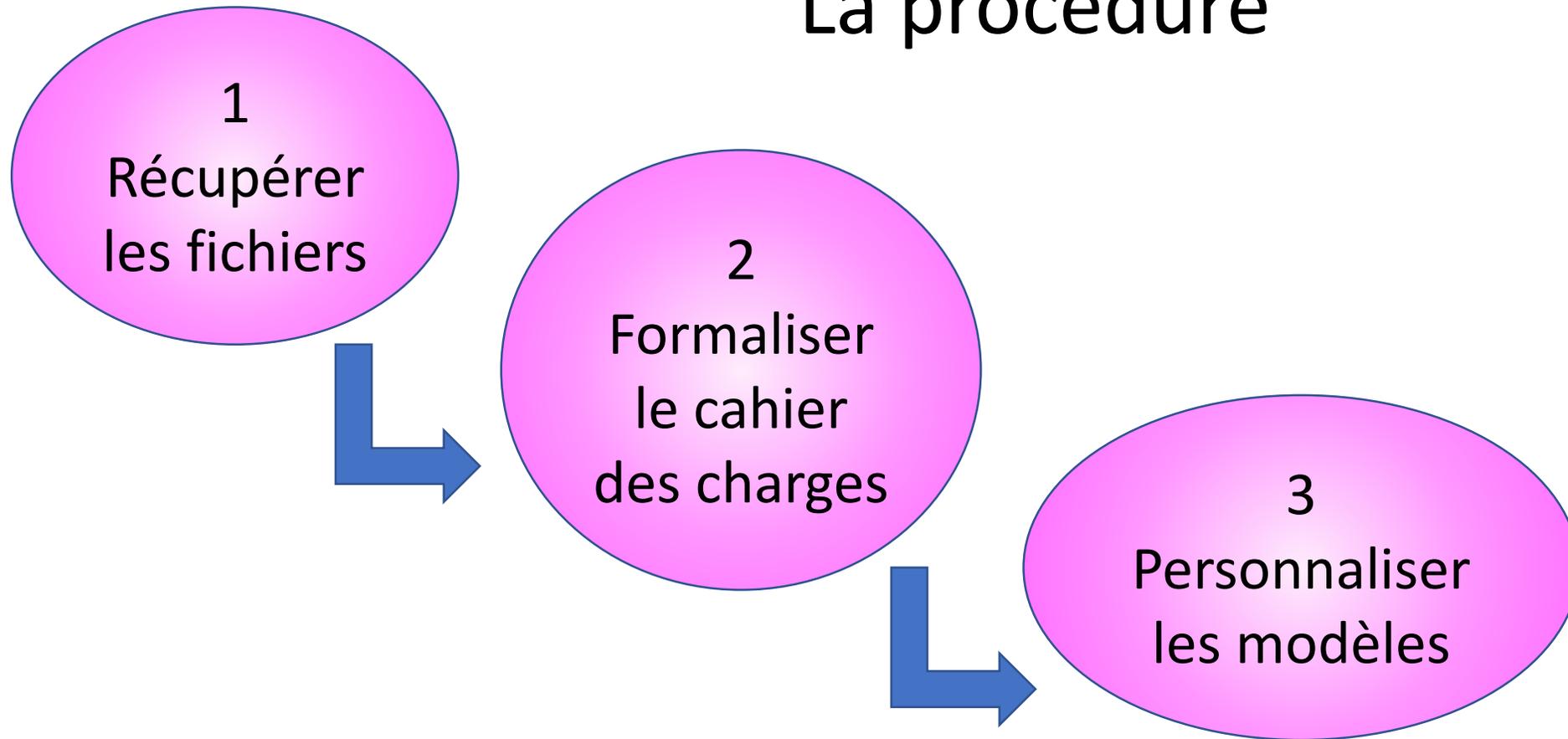
1

Récupérer
les fichiers

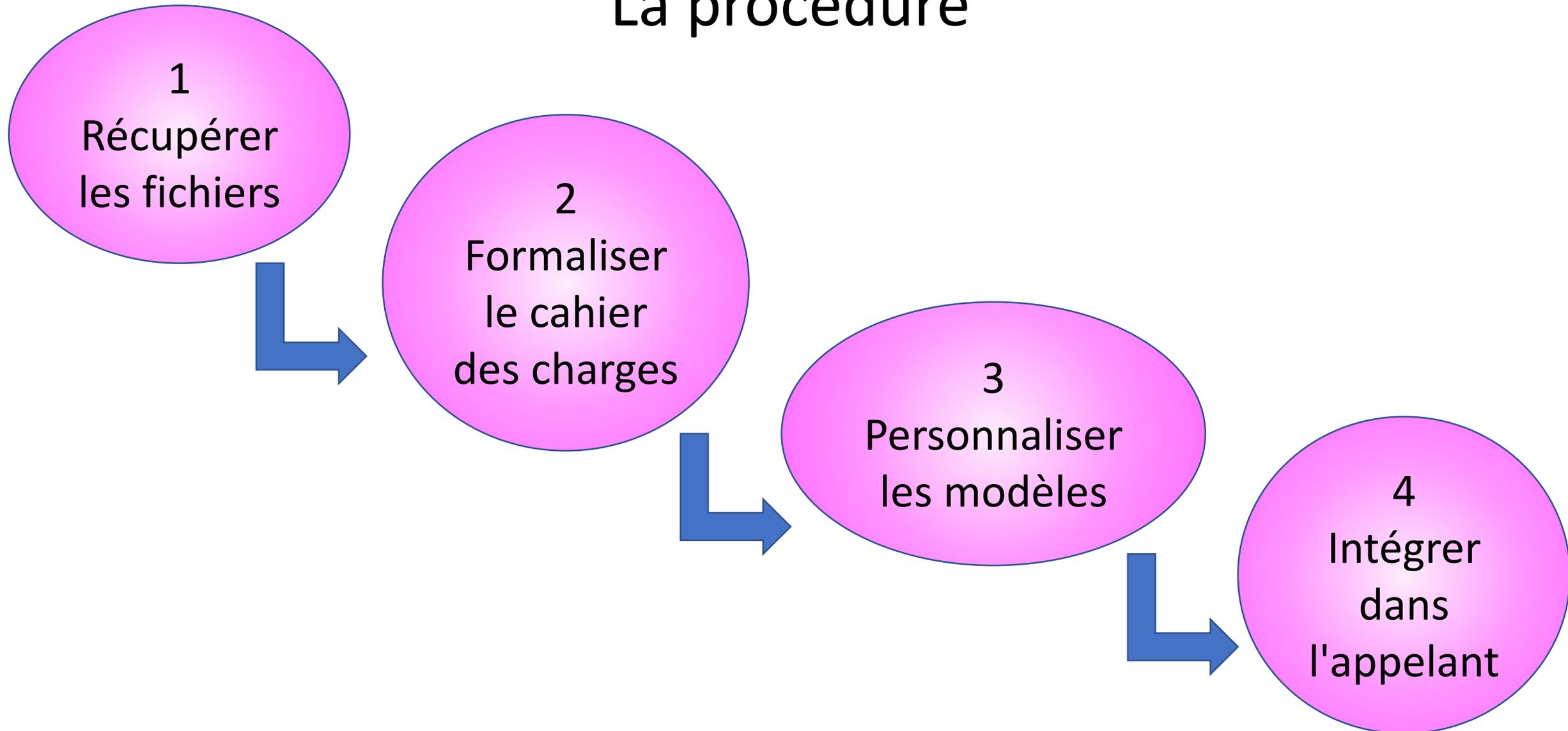
La procédure



La procédure



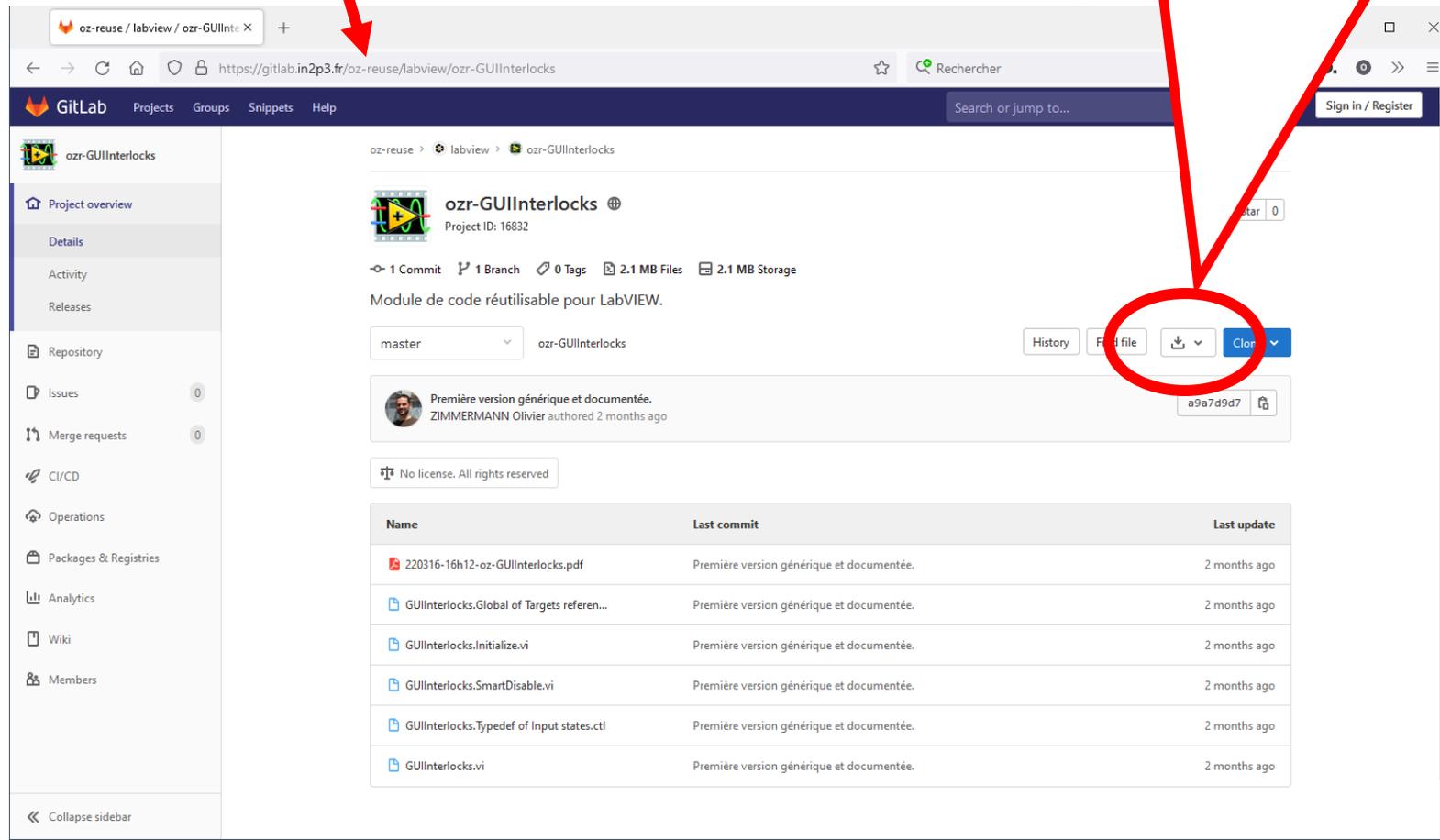
La procédure



1. Récupérer les fichiers

1.1 Récupérer en ligne les fichiers modèles

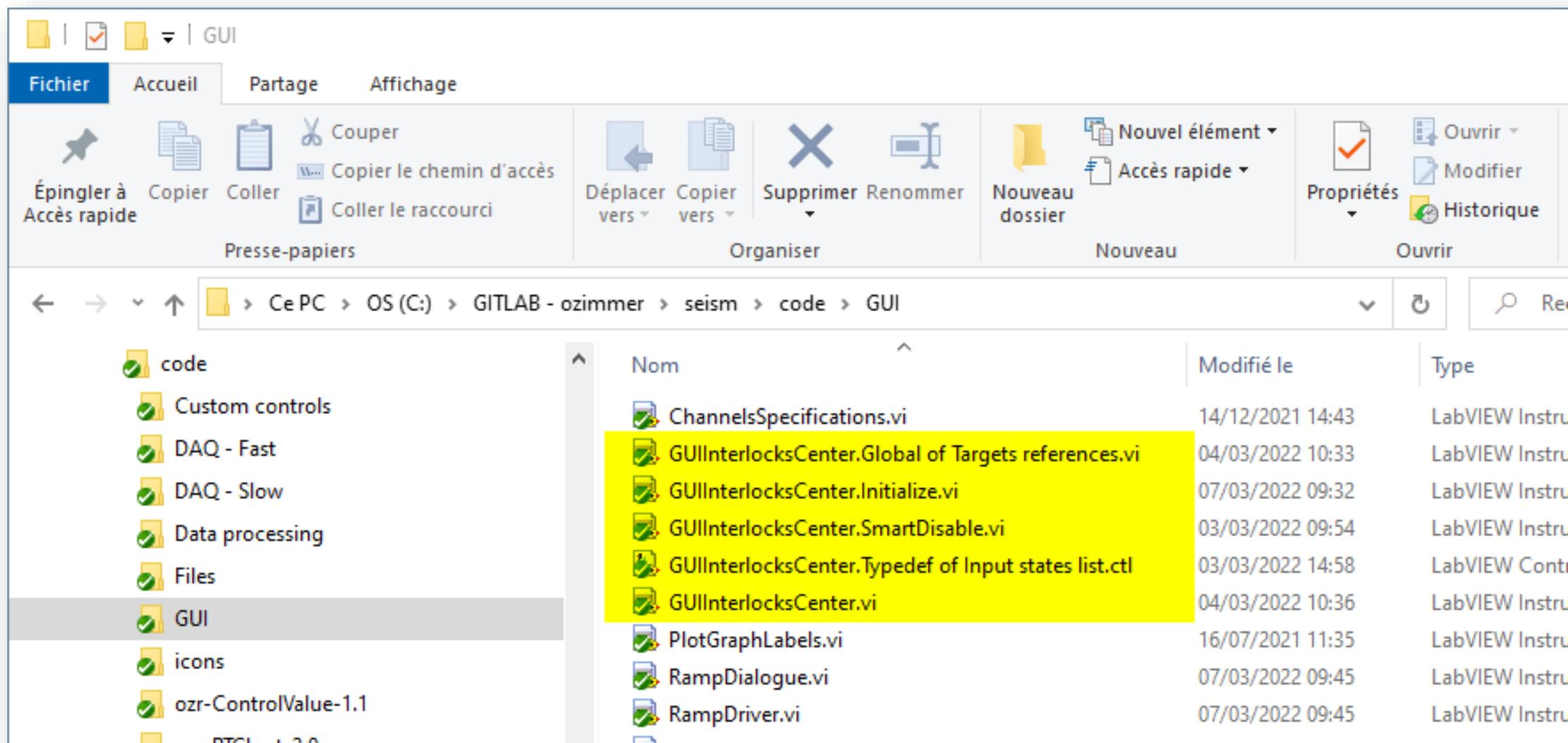
<https://gitlab.in2p3.fr/oz-reuse/labview/ozr-GUIInterlocks>



The screenshot shows the GitLab interface for the repository `ozr-GUIInterlocks`. The browser address bar contains the URL `https://gitlab.in2p3.fr/oz-reuse/labview/ozr-GUIInterlocks`. The repository page displays the project name, ID (16832), and a commit history table. A red callout box highlights the 'Download' button and its dropdown menu, which is open to show options for downloading source code in `zip`, `tar.gz`, `tar.bz2`, or `tar` format.

Name	Last commit	Last update
220316-16h12-oz-GUIInterlocks.pdf	Première version générique et documentée.	2 months ago
GUIInterlocks.Global of Targets referen...	Première version générique et documentée.	2 months ago
GUIInterlocks.Initialize.vi	Première version générique et documentée.	2 months ago
GUIInterlocks.SmartDisable.vi	Première version générique et documentée.	2 months ago
GUIInterlocks.Typedef of Input states.ctl	Première version générique et documentée.	2 months ago
GUIInterlocks.vi	Première version générique et documentée.	2 months ago

1.2 Ajoutez les 5 fichiers modèles du dispositif dans votre code source.



2.

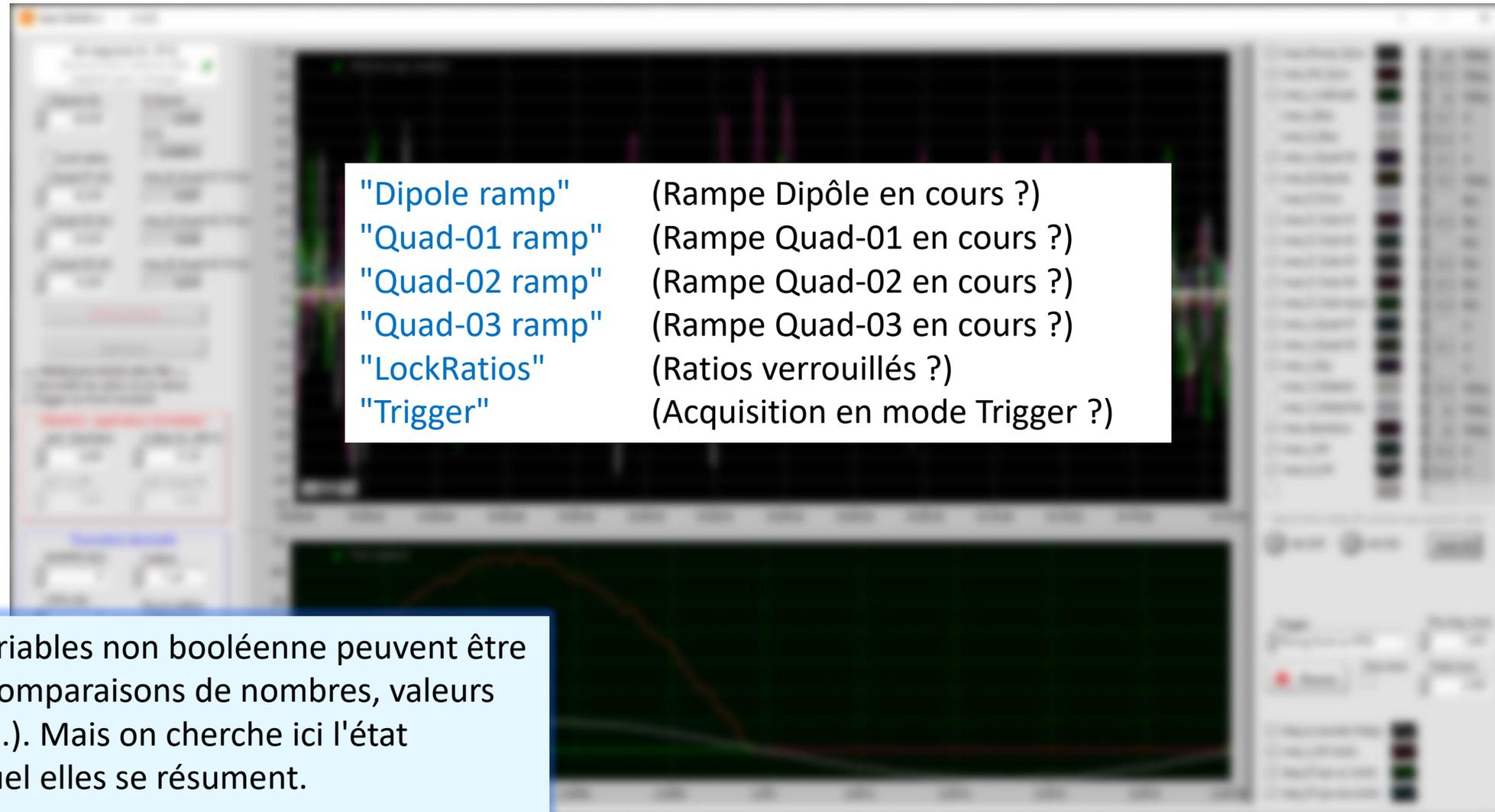
Identification du cahier des charges dans le
programme

2.1 Listez les commandes ou indicateurs qui doivent obéir à des règles de (dé)verrouillage (*états de sortie*).

The screenshot displays the 'Start SEISM.vi' interface (v1.0.0) with several key sections:

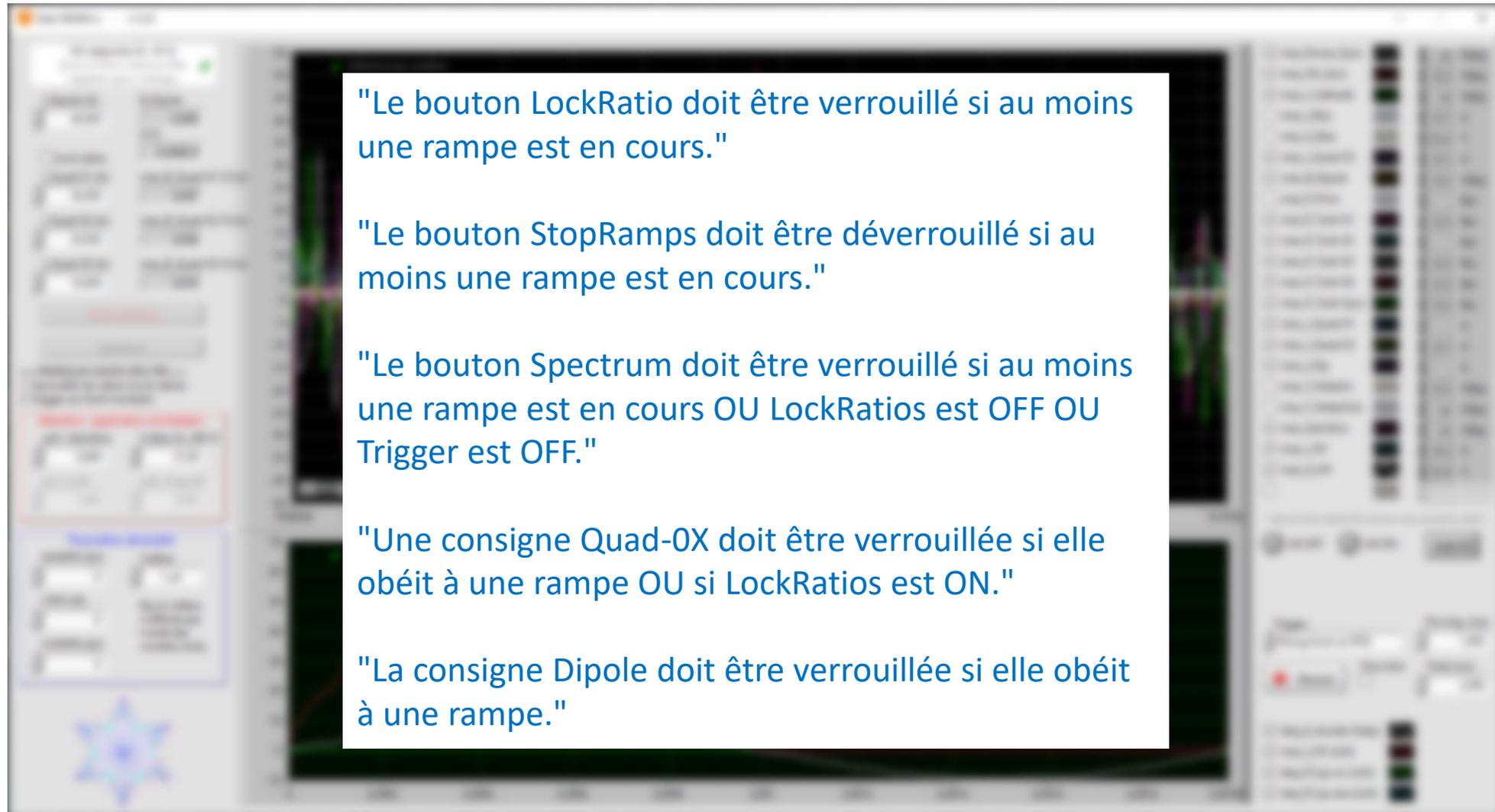
- AO setpoints (0...10 V):** A panel on the left with red arrows pointing to controls for I_{Dipole} (40,000 A), B_{Dipole} (-0,004 T/m), $I_{Quad-01}$ (30,000 A), $I_{Quad-02}$ (20,000 A), $I_{Quad-03}$ (10,000 A), and A/Q (8,564E-6). It includes a 'STOP RAMP(S)' button and a 'Spectrum' button.
- Prerequisites:** A section titled '== PREREQUIS MODE SPECTRE ==' with instructions: '1. Verrouiller les ratios (Lock ratios)' and '2. Trigger sur front montant'.
- Attention: application immédiate!** A red-bordered box containing controls for U_{Bias} (31,00 V), $ao0$: GasValve (4,000), $ao7$: t_HF (7,000), and $ao8$: Freq_HF (8,000).
- Paramètres déclaratifs:** A section with controls for AHVPPS (kV), CFPS (W), CHVPPS (kV), and Calibre (1 μ A).
- Monitoring Graphs:**
 - Top Graph:** 'Monitorings readout' showing multiple colored waveforms (green, blue, red, black) over time from 10:09:40 to 10:10:06. A green checkmark indicates 'Monitorings readout'.
 - Bottom Graph:** 'Fast readout' showing a red curve peaking at approximately 65 and a black curve peaking at approximately 10, over a time range from 0 to 0,00198. A green checkmark indicates 'Fast readout'.
- Right Panel:** A list of monitored variables with checkboxes and line style/color indicators. Checked items include mes_Power_Gyro , mes_HV_Gyro , $mes_I_Cathode$, $mes_I_Quad-03$, mes_B_Dipole , $mes_P_Turb-01$, $mes_P_Turb-02$, $mes_P_Turb-03$, $mes_P_Turb-04$, $mes_P_Turb-Gyro$, $mes_I_Quad-01$, $mes_I_Quad-02$, mes_I_Dip , $mes_GasValve$, mes_I_HV , and mes_U_HV .
- Trigger Section:** Controls for 'Trigger' (Rising front on PFI0), 'Pre-trig. (ms)' (1,000), 'Record' (checked), 'One shot' (unchecked), and 'Total (ms)' (2,000).

2.2 Identifiez et nommez les états booléens dont dépendent les règles de (dé)verrouillage (*états d'entrée*).



Note : Des variables non booléenne peuvent être impliquées (comparaisons de nombres, valeurs particulières...). Mais on cherche ici l'état booléen auquel elles se résument.

2.3 Ecrivez sous forme logique les différentes *règles de verrouillage* reliant les états d'entrée aux états de sortie.



3. Personnalisation des fichiers modèles

3.1 Typedef of Input states list.ctl

Saisissez la liste des états d'entrée.

Boolean input state name

Dipole ramp

Propriétés de l'énum: Boolean input state name

Apparence Type de données Format d'affichage Éditer les éléments

Éléments	Valeurs
Dipole ramp	0
Quad-01 ramp	1
Quad-02 ramp	2
Quad-03 ramp	3
LockRatios	4
Trigger	5

Insérer

Supprimer

Déplacer vers le haut

Déplacer vers le bas

Désactiver l'élément

Autoriser les valeurs non définies à l'exécution

"Dipole ramp"
"Quad-01 ramp"
"Quad-02 ramp"
"Quad-03 ramp"
"LockRatios"
"Trigger"

3.2 Global of Targets references.vi

Créez un indicateur global pour la référence de chaque commande ou indicateur à (dé)verrouiller.

I_Dipole (A)
DBI

I_Dipole (A)
Nombre

[PLACEZ ICI UNE VARIABLE DE REFERENCE POUR CHAQUE COMMANDE OU INDICATEUR FAISANT L'OBJET D'UNE REGLE DE VERROUILLAGE SUR L'INTERFACE GRAPHIQUE UTILISATEUR DU PROGRAMME]

ref "I_Dipole" ref "I_Quad-03" ref "StopRamps"

ref "I_Quad-01" ref "Lock ratios"

ref "I_Quad-02" ref "Spectrum"

Note :

1. A partir du terminal de la commande dans le code du programme, faites "créer une référence".
2. A partir de cette référence, faites "créer une commande".
3. Déplacez cette commande vers le VI global.

3.3 Initialize.vi (face avant)

Répliquez l'ensemble des commandes du VI global, et associez-les aux connexions du VI.

The screenshot displays the LabVIEW environment with two windows. The main window, titled 'GUIInterlocks.Initialize.vi - Face-avant', shows the front panel of the VI. It contains a grid of reference icons for various components: 'ref "I_Dipole"', 'ref "I_Quad-01"', 'ref "I_Quad-02"', 'ref "I_Quad-03"', 'ref "Lock ratios"', and 'ref "Spectrum"'. A red arrow points from the 'ref "Spectrum"' icon in the main window to a callout box on the right. The callout box, titled 'GUIInterlocksCenter.Initialize.vi', contains a list of references: 'ref "I_Dipole"', 'ref "I_Quad-01"', 'ref "I_Quad-02"', 'ref "I_Quad-03"', 'ref "Lock ratios"', 'ref "Spectrum"', and 'ref "StopRamps"'. A second red arrow points from the 'ref "Spectrum"' icon in the callout box back to the main window. A blue box at the bottom right contains a note.

Text in the main window:

Ce VI incorporé (inlined) sert à paramétrer la globale requise par le dispositif "GUI interlocks" (références des commandes faisant l'objet de règles de verrouillage).

Il doit être appelé en pré-initialisation du programme. Sur l'appel à ce VI, décochez "Afficher sous forme d'icône" pour améliorer la lisibilité de son câblage.

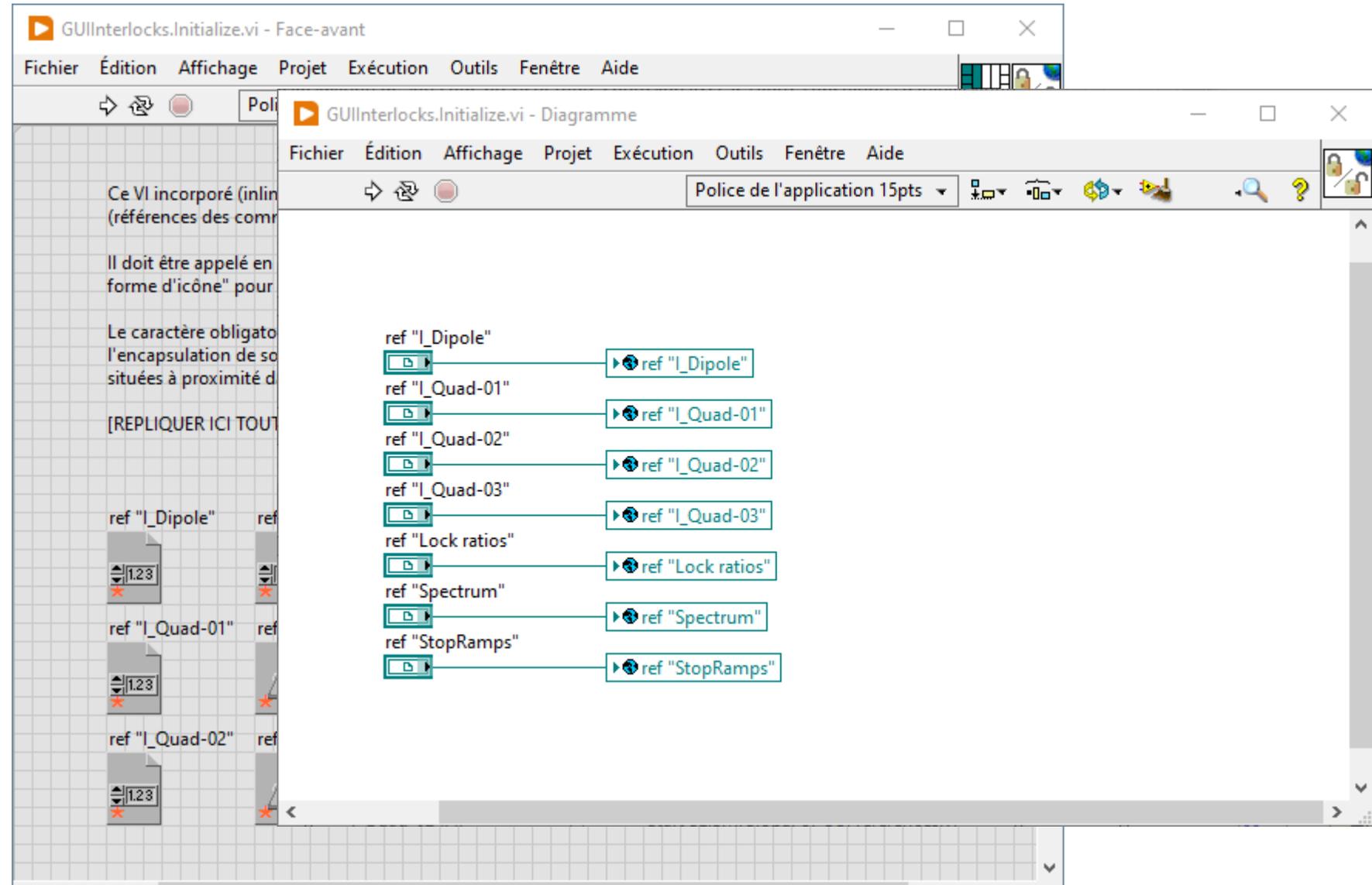
Le caractère obligatoire de ses entrées garantit que chaque variable requise est bien définie, et l'encapsulation de son code prévient toute confusion avec les autres opérations d'initialisation situées à proximité dans le code appelant.

[REPLIQUER ICI TOUTES LES REFERENCES SPECIFIEES DANS LA VARIABLE GLOBALE]

Note : La disposition et le nombre des connexions sont sans importance car on décochera "Visualiser sous la forme d'une icône" dans l'appelant.

3.4 Initialize.vi (diagramme)

Câblez la valeur de chaque entrée vers la globale qui lui correspond.



3.5 GUIInterlocks.vi (face avant)

Supprimez tous les indicateurs booléens du modèle sauf un et donnez-lui le nom du premier état d'entrée.

GUIInterlocks.vi - Face-avant

Fichier Édition Affichage Projet Exécution Outils Fenêtre Aide

Police de l'application 15pts

Rechercher

Ce VI a pour vocation de centraliser la gestion du verrouillage/déverrouillage des éléments d'interface utilisateur ("GUI interlocks") en fonction d'un certain nombre d'états d'entrée caractérisant le programme appelant.

Il permet au programme appelant de déclarer tout changement de valeur (passage à TRUE ou FALSE) d'un état d'entrée impliqué dans les verrouillages de l'interface graphique. A chaque appel, il recalcule les états de sortie et les applique si nécessaire.

Les références des commandes cibles doivent être affectées au moyen de variables globales en pré-initialisation du logiciel appelant. Les états d'entrée, les états de sortie et la logique qui les relie doivent tous être entièrement conçus pour chaque application (se référer aux commentaires dans le code).

Ce VI doit être appelé ponctuellement chaque fois qu'un de ses états d'entrée change de valeur, et à l'initialisation pour tout état d'entrée vrai au démarrage.

Interlock state variable

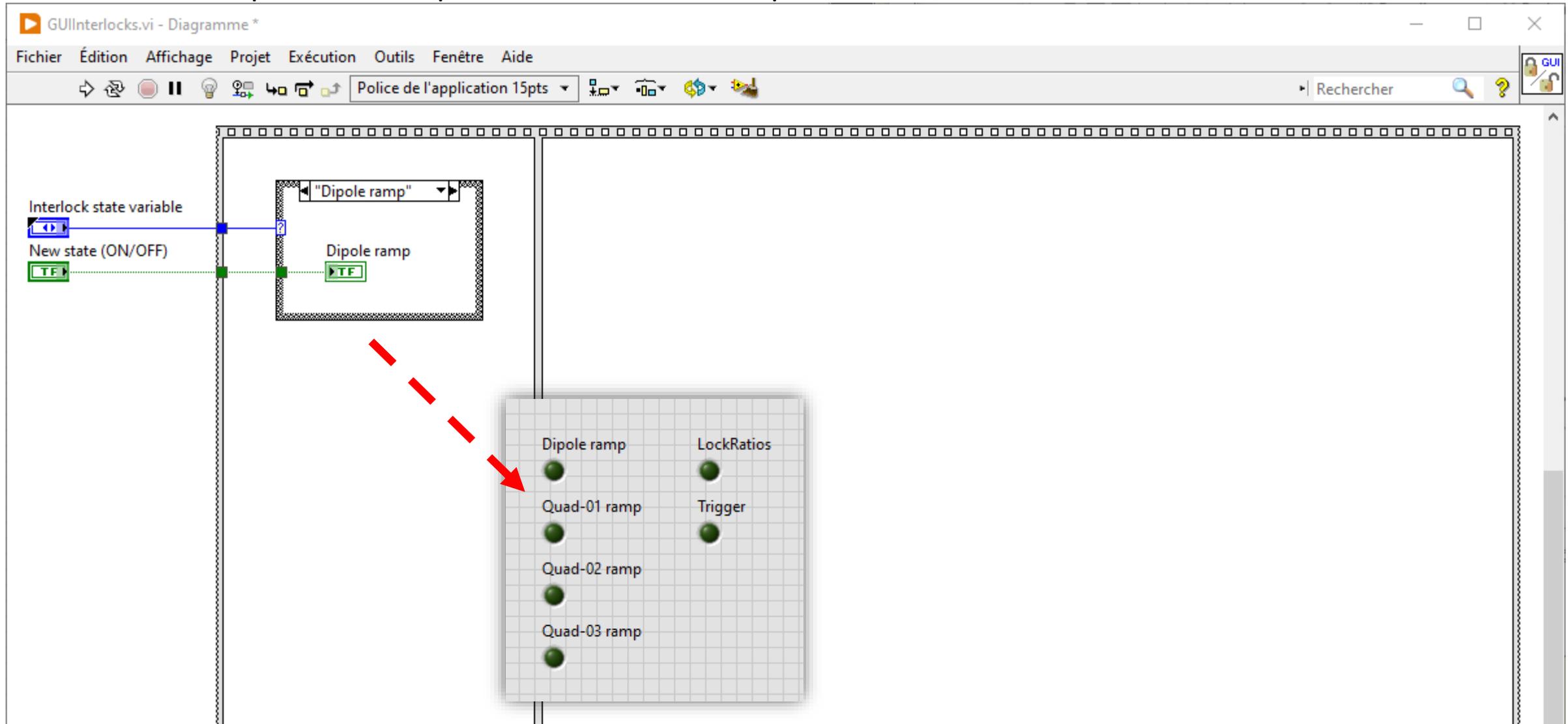
Trigger

New state (ON/OFF)

Dipole ramp

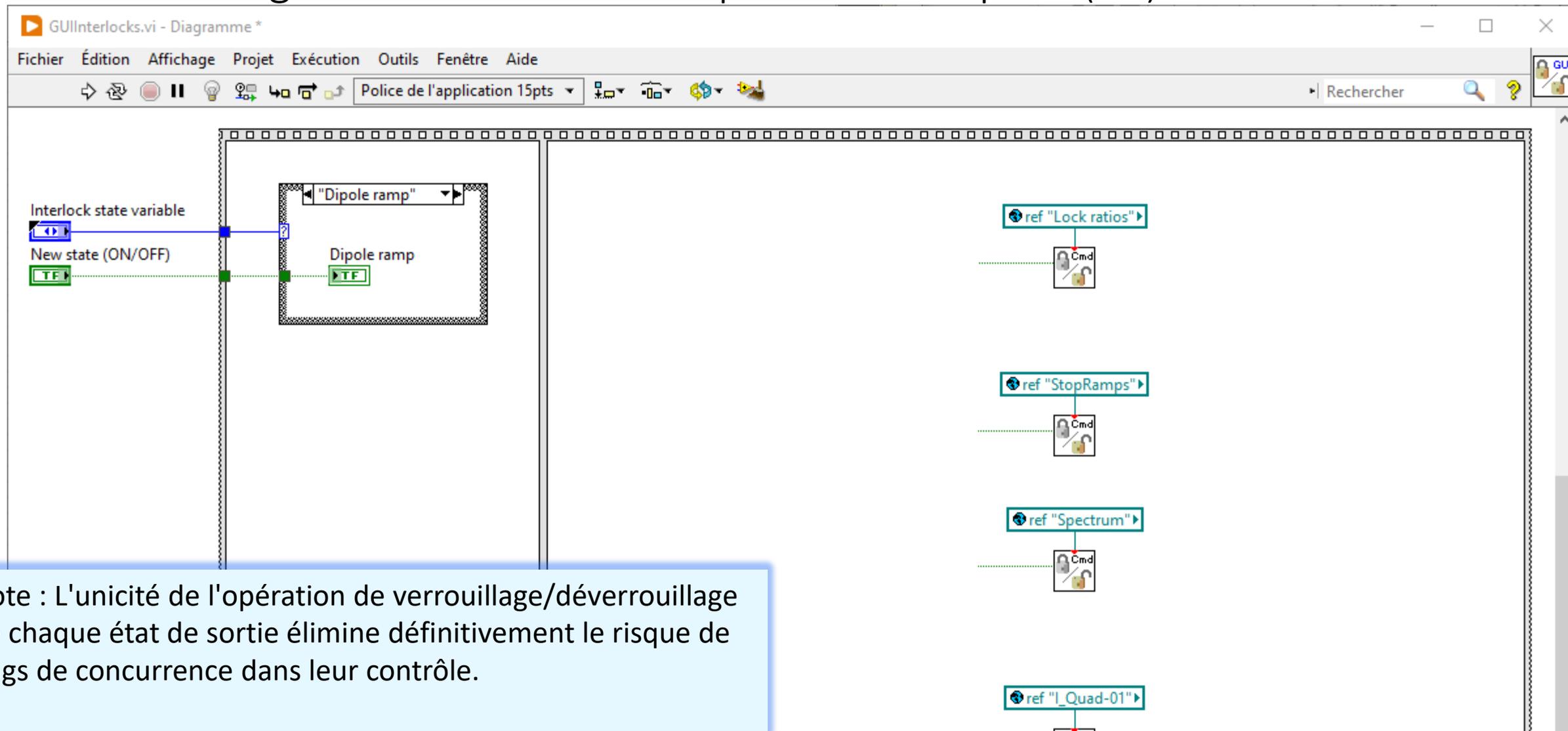
3.6 GUIInterlocks.vi (diagramme, à gauche)

Supprimez les cas vides de la structure conditionnelle et dupliquez le cas restant pour chaque état d'entrée possible. Renommez les indicateurs.



3.7 GUIInterlocks.vi (diagramme, à droite)

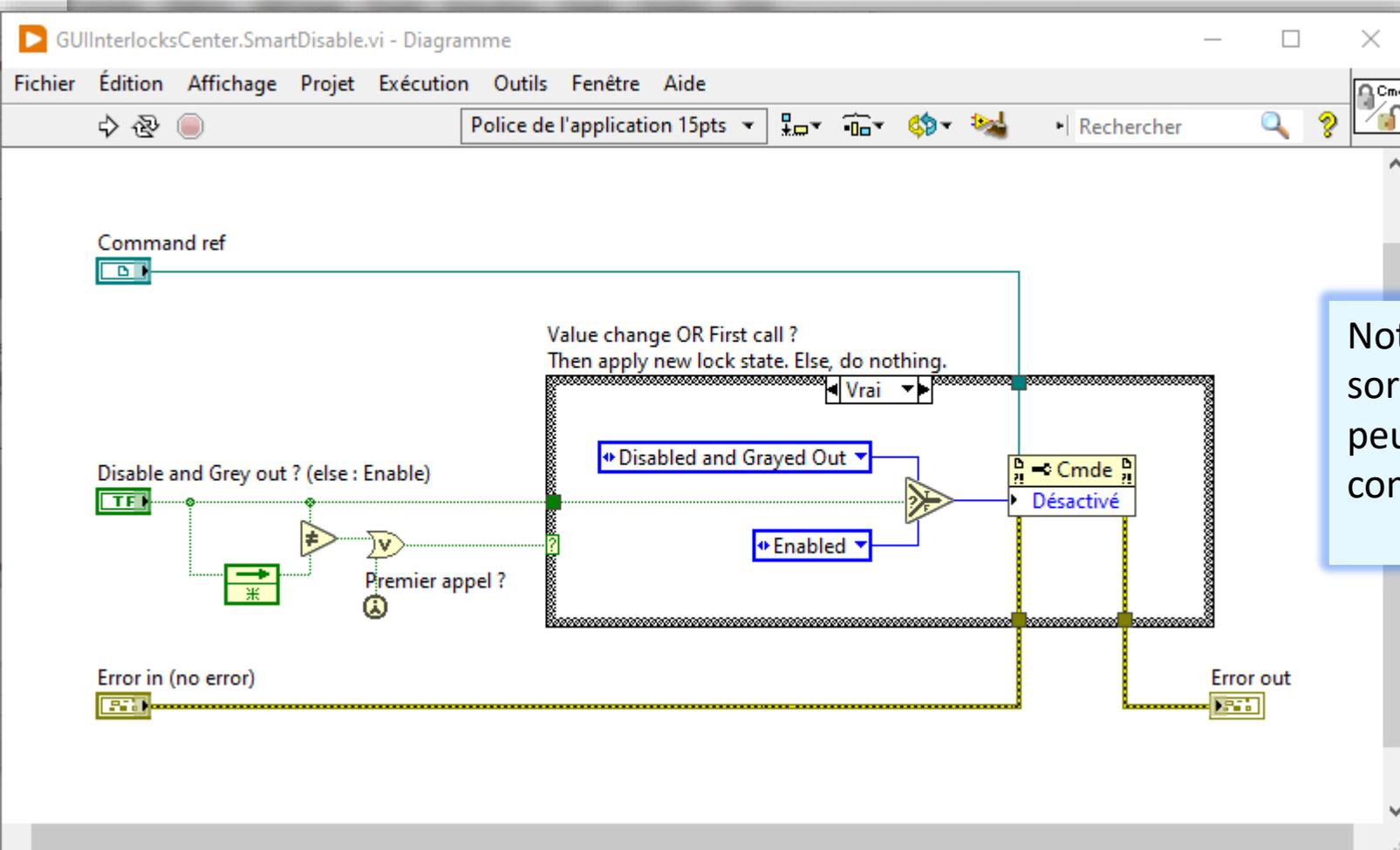
Chaque référence cible doit apparaître ici sous la forme d'une et une seule variable globale en lecture. Adaptez le code qui la (dé)verrouille.



Note : L'unicité de l'opération de verrouillage/déverrouillage de chaque état de sortie élimine définitivement le risque de bugs de concurrence dans leur contrôle.

3.7 GUIInterlocks.vi (diagramme, à droite)

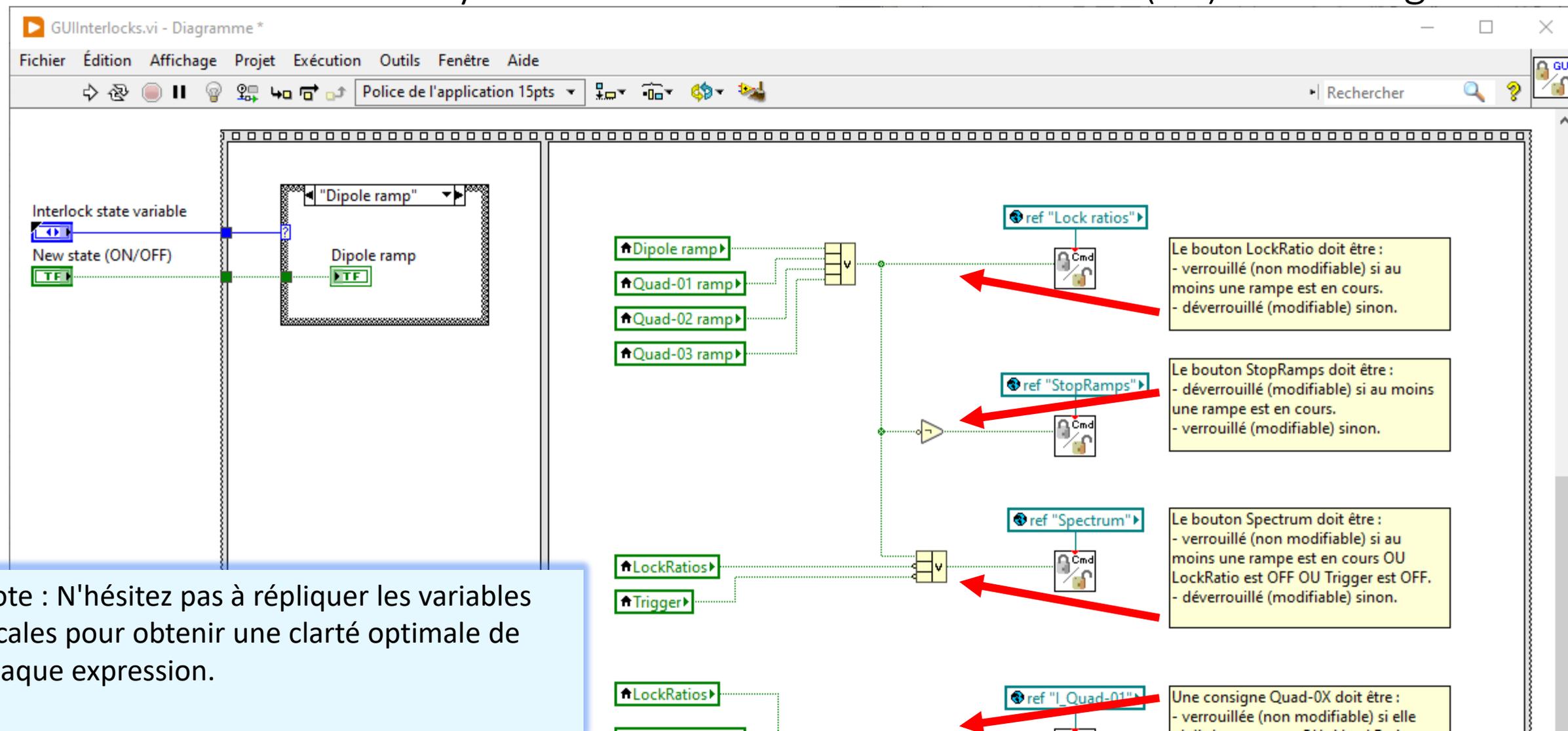
Chaque référence cible doit apparaître ici sous la forme d'une et une seule variable globale en lecture. Adaptez le code qui la (dé)verrouille.



Note : Le risque de concurrence sur les états de sortie étant éliminé, le code de (dé)verrouillage peut être conçu pour n'opérer que si sa consigne change (ici au moyen d'un sous-VI).

3.8 GUIInterlocks.vi (diagramme, au centre)

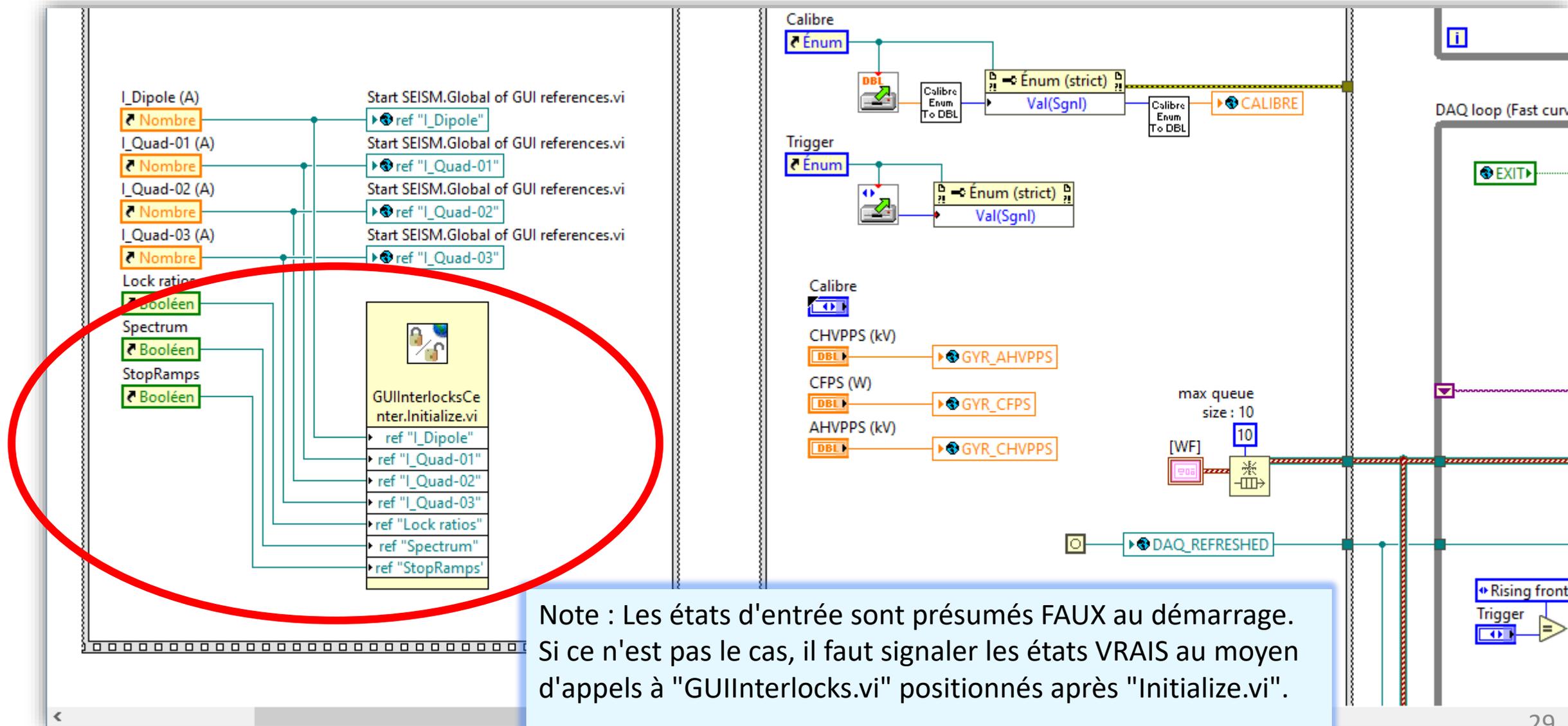
Implémentez les expressions logiques en les alimentant avec des variables locales et en envoyant les résultats sur les actions de (dé)verrouillage.



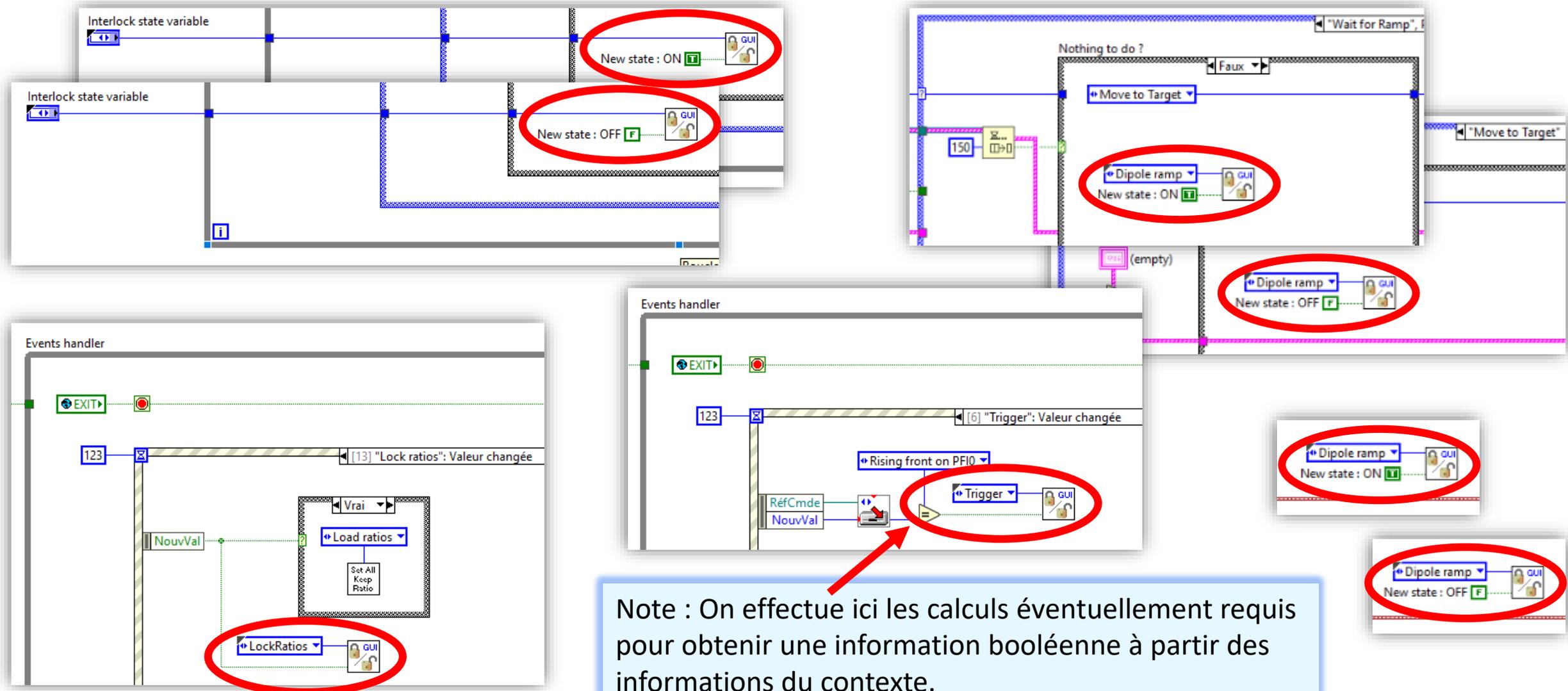
Note : N'hésitez pas à répliquer les variables locales pour obtenir une clarté optimale de chaque expression.

4. Intégration au programme appelant

4.1 En (pré)-initialisation, câblez vers "Initialize.vi" les références (constantes) des commandes et indicateurs cibles.



4.2 A travers le code : partout où un état d'entrée change de valeur, ajoutez un appel à "GUIInterlocks.vi" en spécifiant son nom et sa nouvelle valeur.



Note : On effectue ici les calculs éventuellement requis pour obtenir une information booléenne à partir des informations du contexte.

Conclusion : forces et faiblesses

- ⊕ Un seul VI simple assure la gestion des états et leur mise à jour (code léger).
- ⊕ L'application des états de sortie est immédiate (performance).
- ⊕ Les états de sortie sont tous recalculés à chaque changement d'un état d'entrée et uniquement à ce moment-là (garantie de cohérence, même en présence d'états interdépendants).
- ⊕ Un état de sortie n'est appliqué que s'il a changé (charge minimale).
- ⊕ La règle logique qui gouverne un état de sortie est centralisée, lisible (évolutivité, vérifiabilité).
- ⊕ L'intersection avec le code appelant est minimisée (découplage des facettes logicielles).
- ⊕ Les structures de code mise en œuvres sont plutôt simples (simplicité d'usage).
- ⊖ Nombreuses étapes manuelles pour configurer le code modèle (procédure nécessaire).
- ⊖ Les états d'entrée sont des abstractions : leur pertinence dépend du programmeur.
- ⊖ Recours à des variables globales pour passer les références à "GUIInterlocks.vi".
- ⊖ Mise en œuvre d'astuces peu conventionnelles (VI incorporé pour initialiser les globales, mémorisation des états d'entrée avec des indicateurs, usage de variables locales...).