Introduction à PVSS

Pierre-Yves Duval, *Centre d e Physique des Particules de Marseille, Aix-Marseille Université, Marseille, France*

Notion de SCADA

SCADA: Supervisory Control And Data Acquisition

L'idée générale est celle d'un système de télégestion/téléopération:

- à grande échelle
- réparti des points de vue des mesures et des commandes.

Les SCADA sont employés pour surveiller ou commander:

- des usines de processus chimiques (raffineries, papeteries …)
- des systèmes de transports,
- l'approvisionnement en eau,
- la génération d'énergie électrique,
- la transmission et la distribution de gaz, pétrole, …

Ils appartiennent au monde de l'automatisation industrielle.

PVSS au CERN

PVSS est un SCADA créé par **ETM** une petite société autrichienne rachetée depuis par **SIEMENS.**

- outil d'implantation du contrôle du LHC (accélérateur)
- choisi en 2000 pour implanter le contrôle des quatres expériences LHC
- recommandé depuis 2002 le board du CERN pour implanter tous les systèmes de contrôle de type SCADA des expériences au CERN.

Un groupe de 8 personnes de la **division IT/CO assurent support**, formation et le développement d'un framework spécifique ajoutant à PVSS des fonctionnalités utiles aux expériences LHC.

Ce framework est développé dans le cadre du projet JCOP (Joint Controls Project).

Fonctions de PVSS

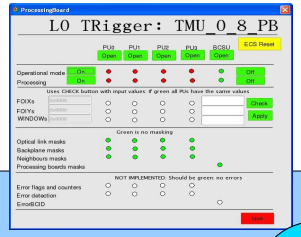
Objectifs:

- Interconnecter des capteurs hardware (capteurs/actionneurs)
- Interconnecter des programmes de calcul et traitement
- Fournir les outils de supervision
- Fournir la trace et l'archivage du fonctionnement d'un système

Moyens:

- Une base de donnée dynamique
- Un outil graphique de création et paramétrisation des objets de la base de données
- Des fonctions d'archivage automatique
- Des production d'alarme et d'association de leur traitements
- Un éditeur graphique de panneaux de supervision (WYSIWYG)
- Un langage interprété de traitement des données (proche du C)
- Des moyens d'intégration de pilotes(drivers) pour les composant hardware (ouverture!)
- Interfaçage avec les protocoles de communication standard (TCP/IP, OPC, Modbus ...)

Architecture de PVSS



Interfaces utilisateurs

UIM

UIM

UIM

Calculs et traitements

Ctrl

API

Communication et mémoire

Dynamic DB

DB manager

Events manager

Pilotes d'appareils

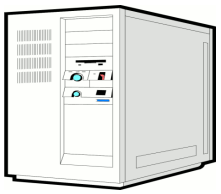
driver

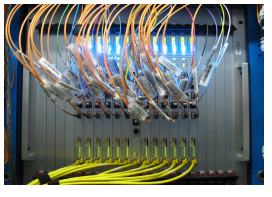
driver

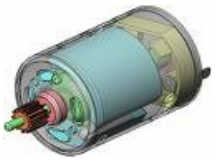
driver

driver

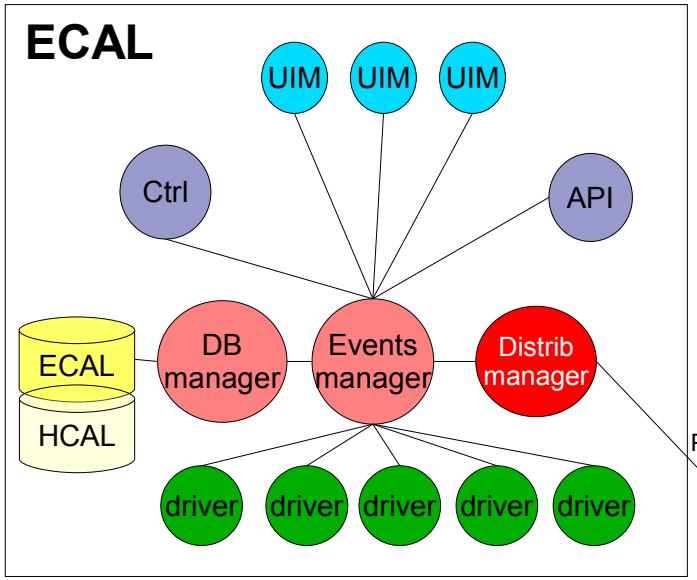
driver







Architecture distribuée

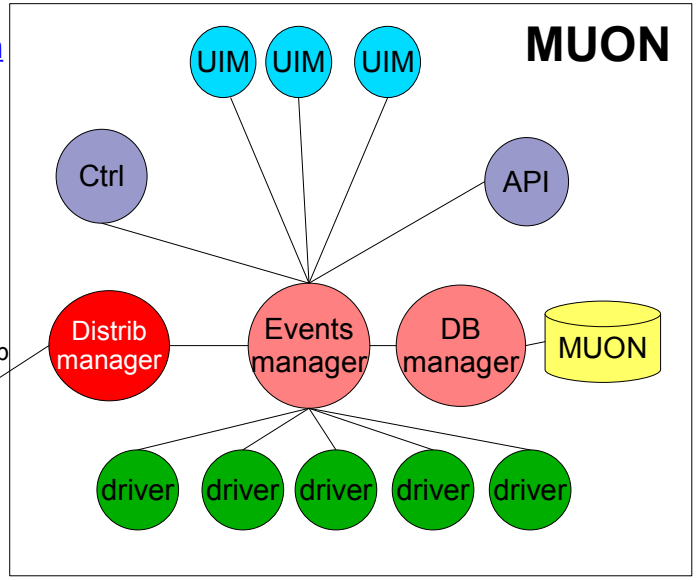


Déclaration dans un fichier de configuration

System name = projet PVSS
Dist Mng port = port Nb (socket)

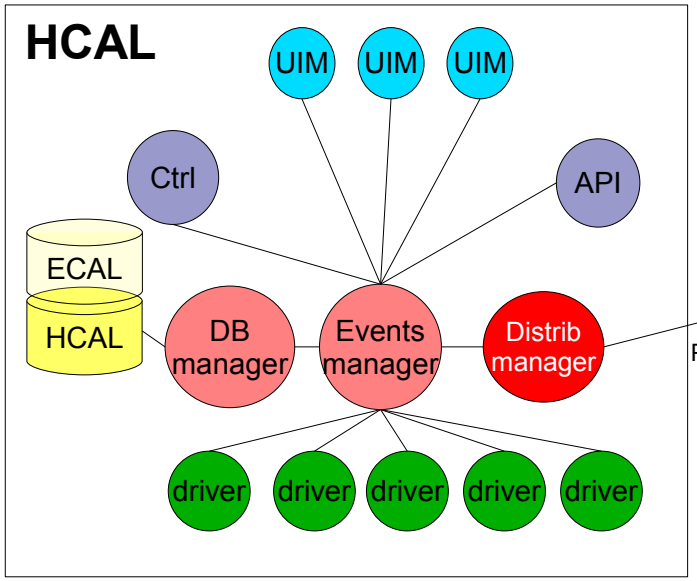
Directive pour ce connecter à un autre projet
DistPeer = « machine » « dist mng port »

Les base de données distantes sont vues comme locales



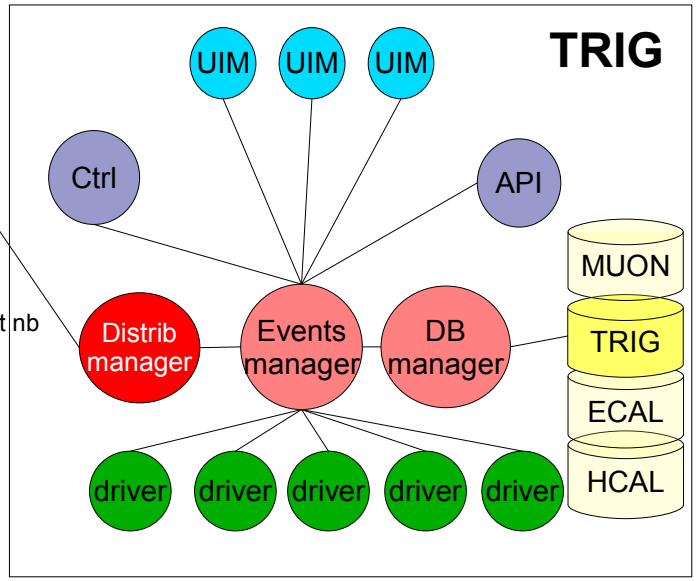
Port nb

Port nb



TCP/IP

Port nb



Datapoints

L'entité de la base de données est un « datapoint » qui est une instance d'un type.

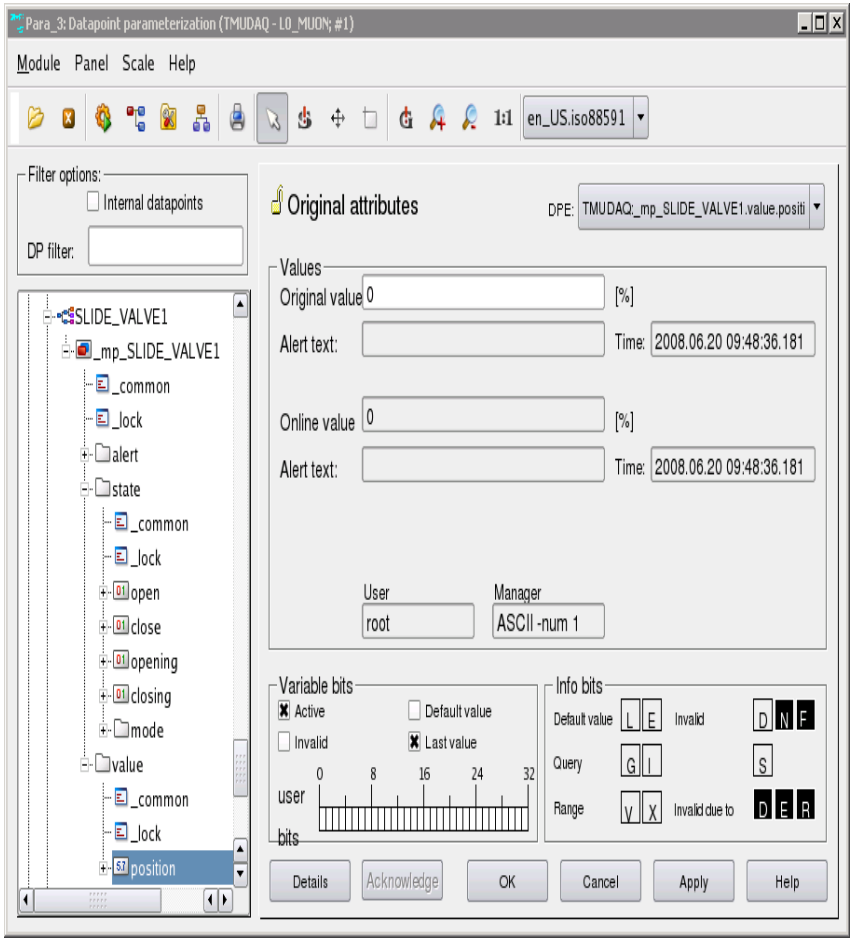
Un datapoint peut être structuré en « datapoint elements » avec des types prédéfinis:
- terminaux(bool, char, uint, int, float ...)
- composites standards (string,blob, dyn_char, dyn_dyn_float, dyn_dyn_anytype ...)
ou de types définis par l'utilisateur.

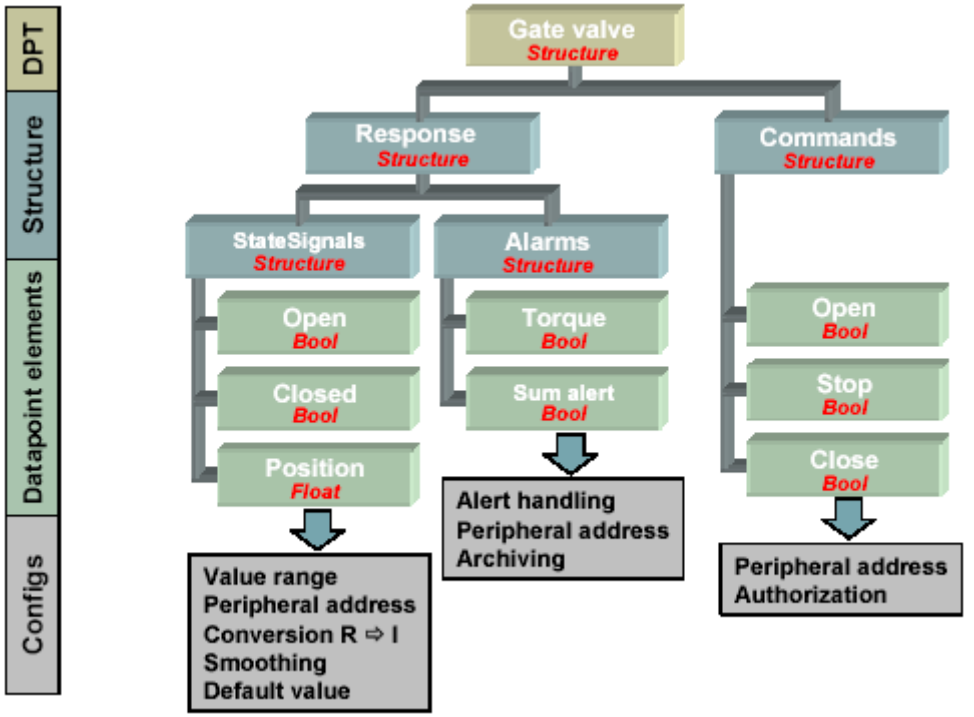
L'identifiant en est unique:

[system:]dpName.[dpElement(s)]:config.[detail].attribute

System: name of the PVSS system (**ECAL, HCAL, TRIG, MUON**)
DpName: name of the data point
DpElement: element name of the DP type
Config: attribute group (default value, **online value**, value range, command conversion ...)
Detail: detail number of the attribute group
Attribute: real value of an attribute

Datapoint: exemple





Example of a DP Structure as device representation

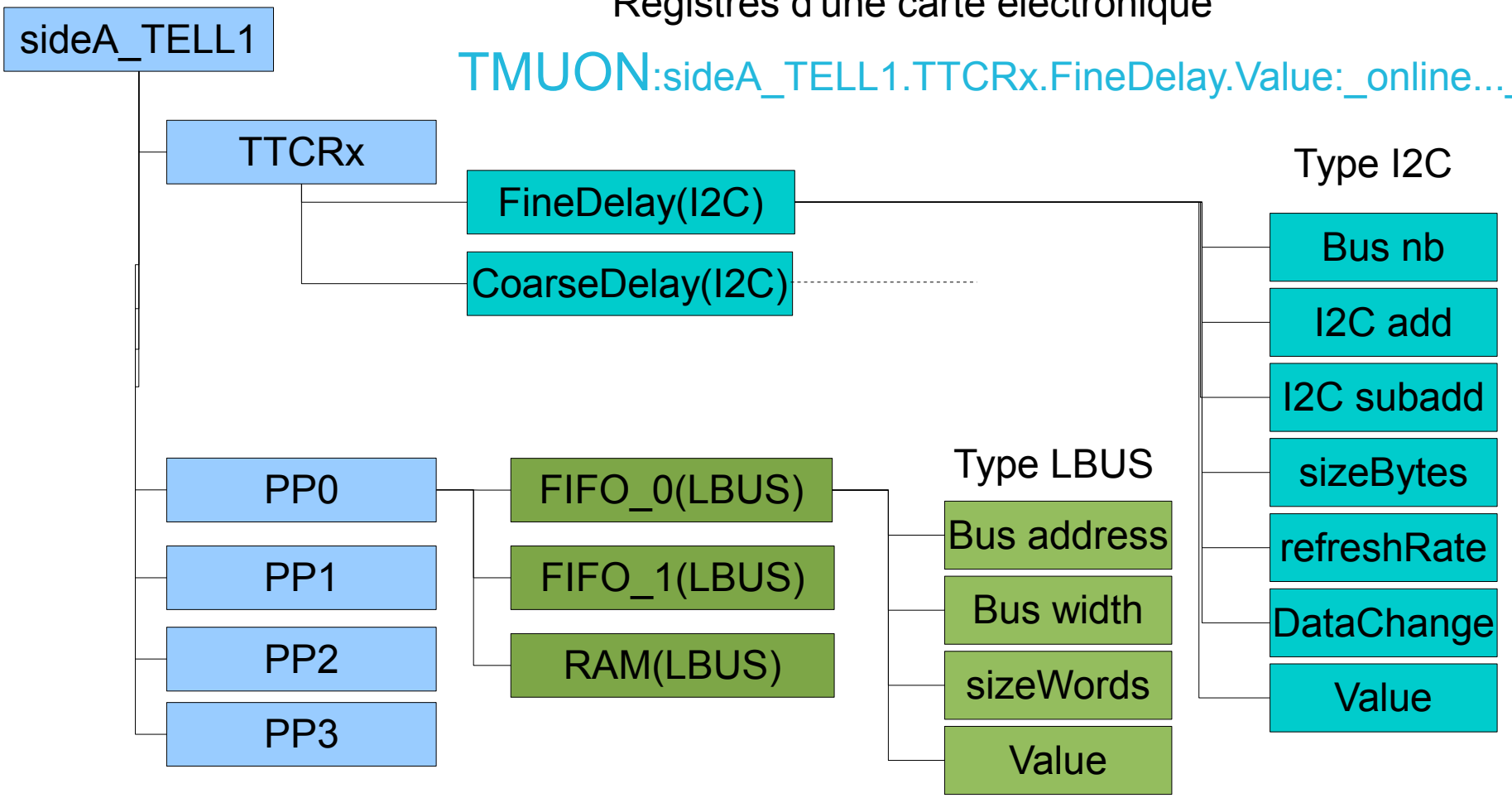
[System:] dpName. [dpElement(s)] : Config. [Detail] .Attribut

valve1.commands.open:_online..._value

Data points: exemple

Registres d'une carte électronique

TMUON:sideA_TELL1.TTCRx.FineDelay.Value:_online..._value



Datapoints: programmation réactive

Dans les panneaux et les scripts on peut associer des fonctions de type callback aux changements de valeur de datapoints elements.

int dpConnect (string work, [bool answer,] string dpe1 [, string dpe2 ...]);

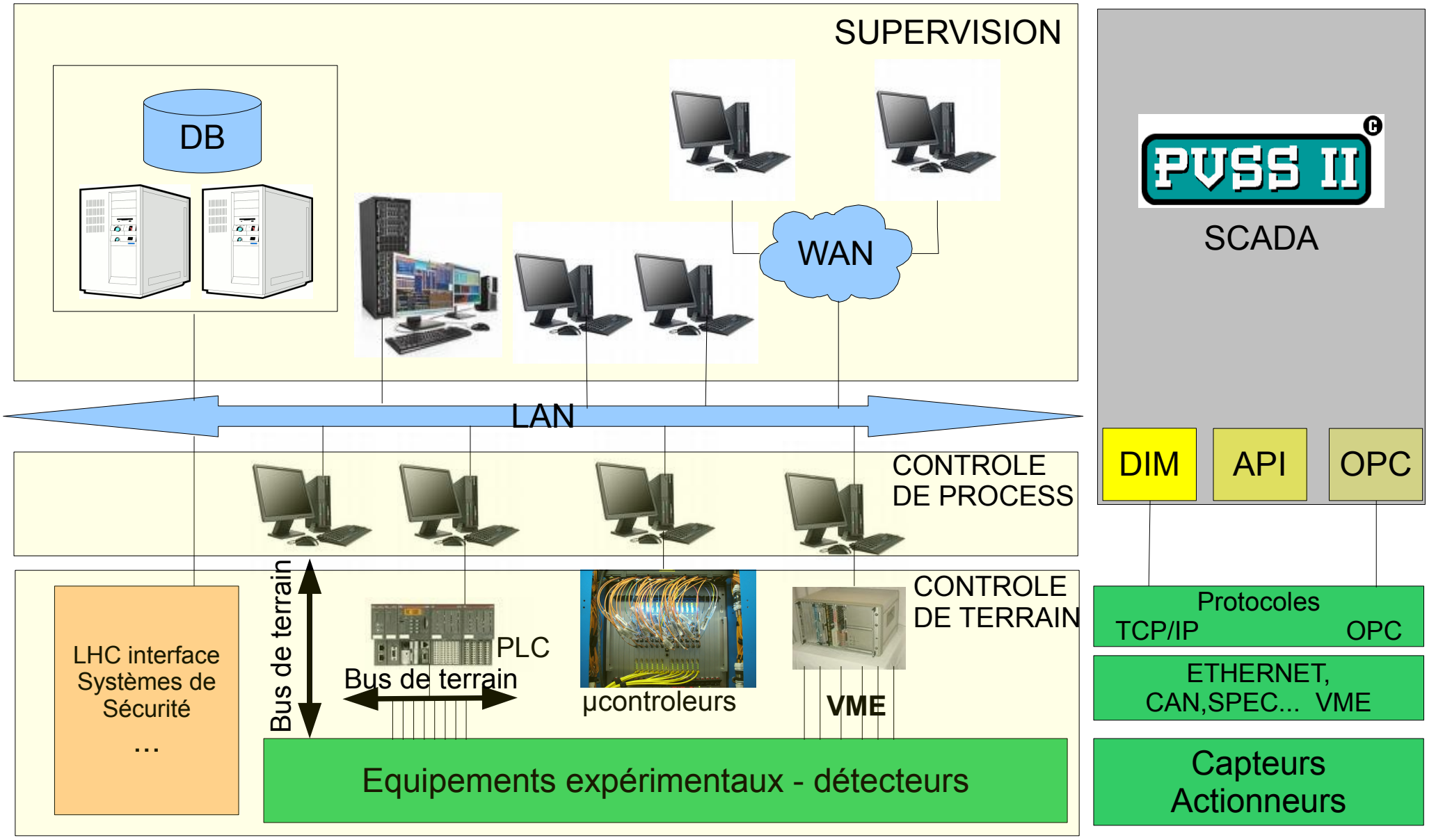
```
dpConnect("workCB",  "valve1.response.opening:_online..._value");
```

La fonction appelée reçoit la nouvelle valeur du DPE.

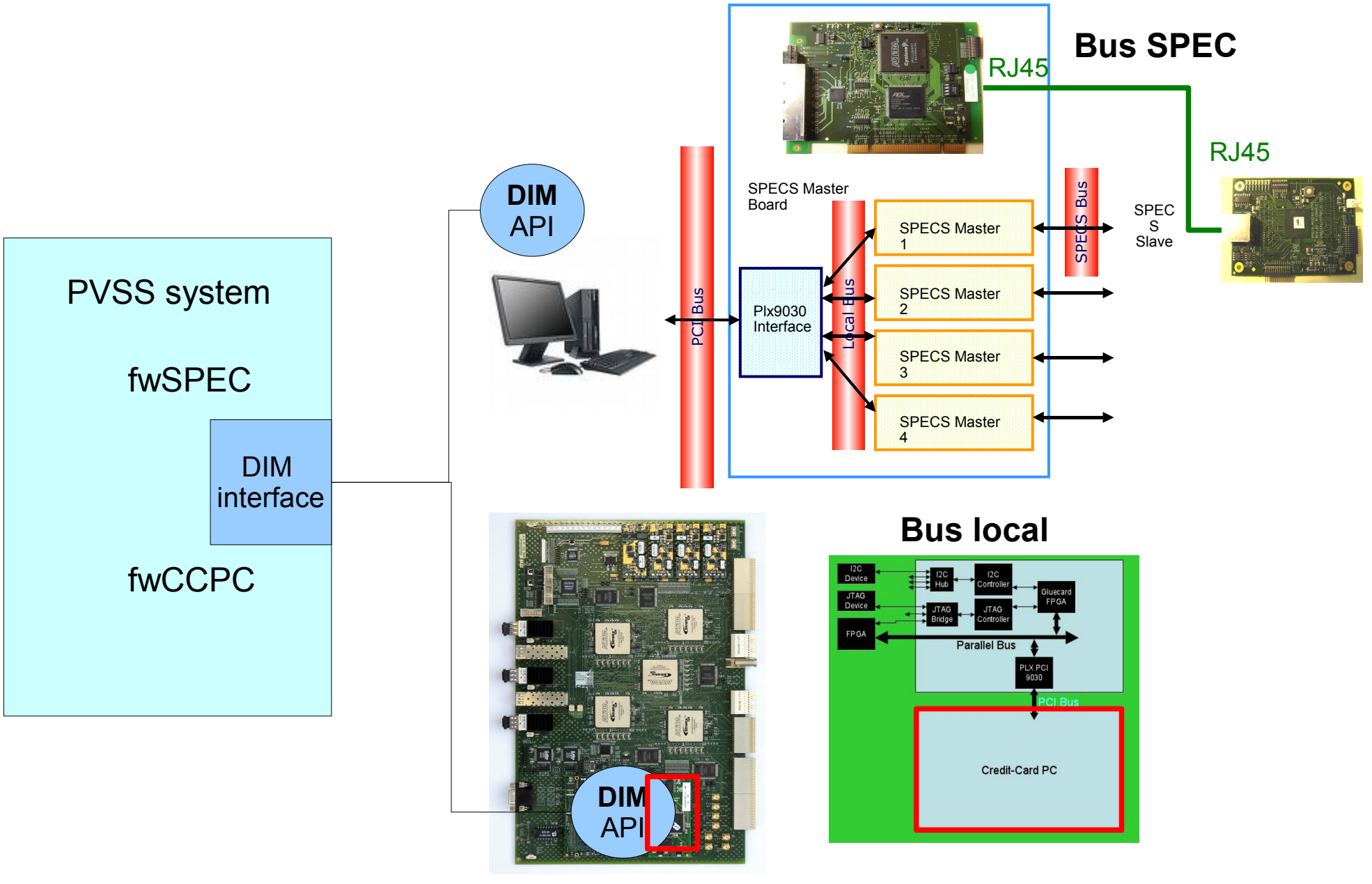
void work(string dpe1, <type1> var1 [, string dpe2, <type2> var2 ...])

```
workCB(string myValveStatus, float angleCourant){
...
}
```

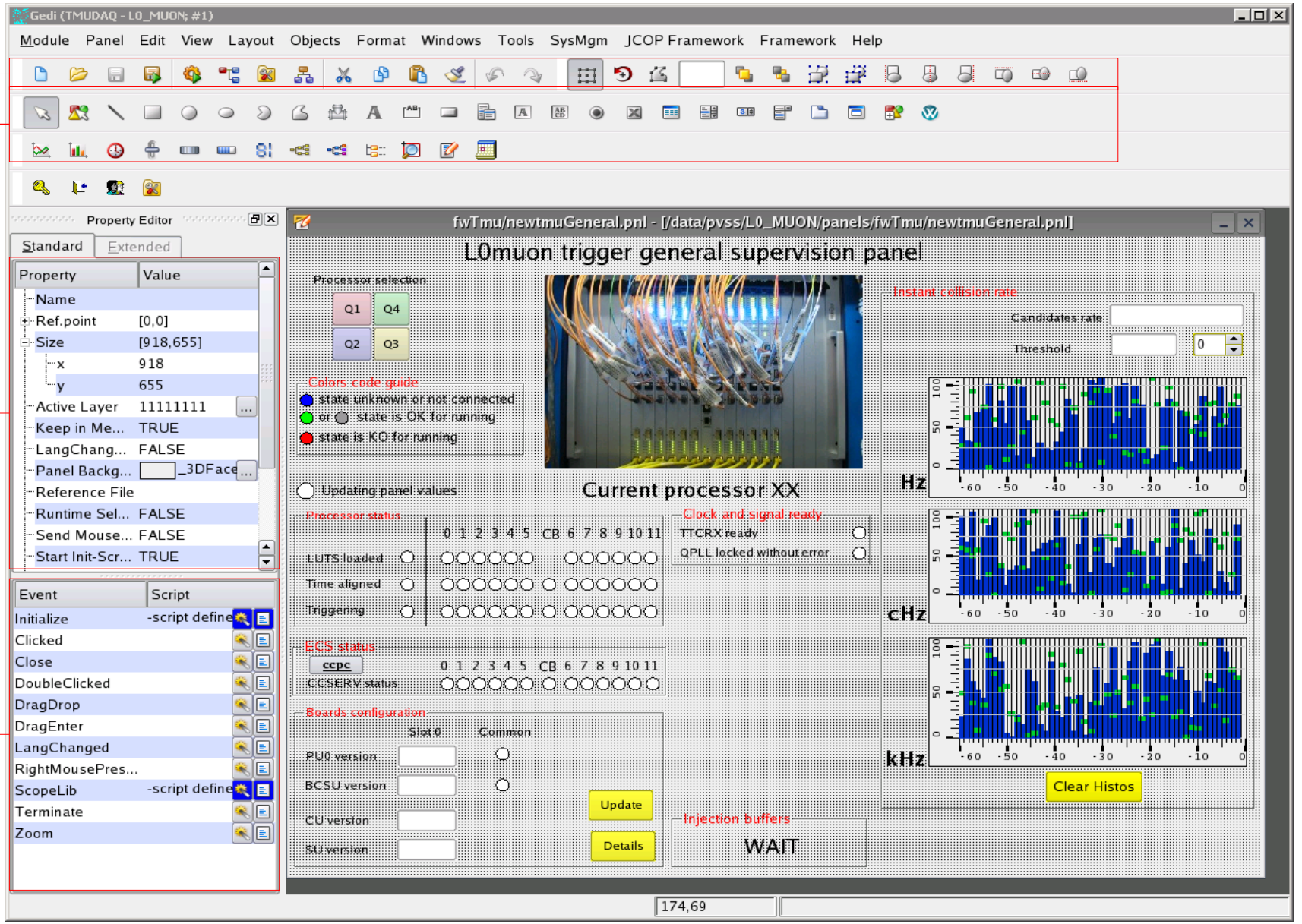
Intégration pilotes



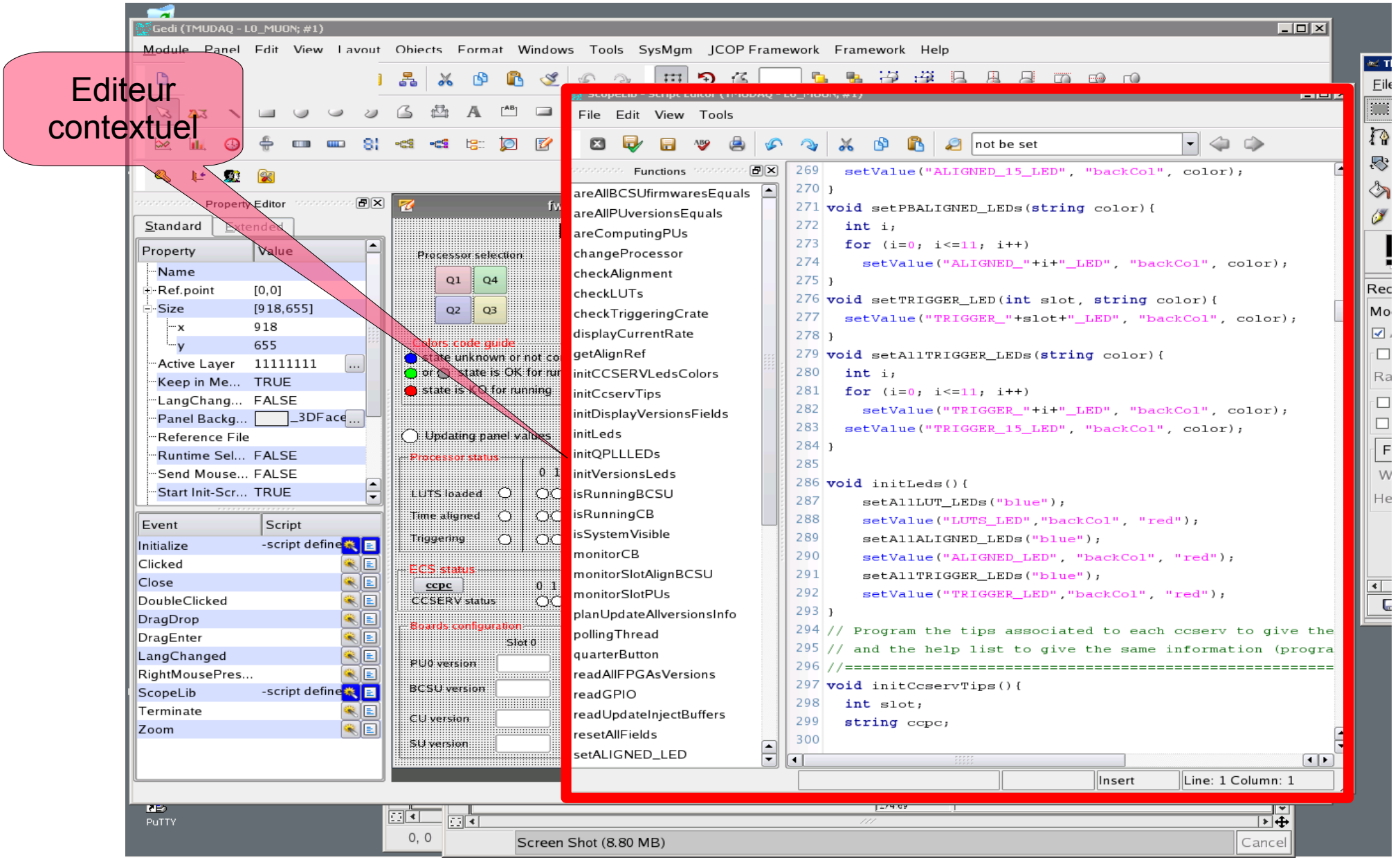
Intégration pilotes: DIM serveurs



Développement d'interfaces



Développement d'interfaces



Scripts et librairies

Le langage de contrôle est **un dialecte de C (simplifié) interprété.**

Ils sert à écrire le code:

- des **activités** associées à des panneaux
- des **scripts** portions de code autonome avec une fonction d'entrée main().
- des **librairies** fonctions réutilisables dans les panneaux et scripts.

Les script peuvent servir à:
- Exécuter du code d'initialisation de projet
- Avoir des tâches de fond développées par l'utilisateur (polling, fonction de calcul, régul ...)

Il existe des fonctions pour:
- manipuler les datapoints
- manipuler les widgets
- gérer des threads
- gérer le temps
- administrer les projets
- s'interfacer (grossièrement) avec l'OS (fichiers et appel de commandes/programmes)

Archivage et Alarmes

Archives:

Si on veut tracer l'historique des valeurs d'un datapoint element il suffit de le déclarer à PVSS. Il archivera les valeurs avec les dates des valeurs (estampilles temporelles).

Le logiciel dispose d'outil pour visualiser graphiquement et administrer les archives PVSS.

PVSS a son propre système de sauvegarde mais il peut aussi être connecté à Oracle.

Alarmes:

On peut associer des alarmes sur des valeurs (ou plages de valeurs) de datapoints élements. Les alarmes sont aussi stockées dans une DB et peuvent être activées/inhibées sélectivement.

Programmation graphique?

PVSS encourage la programmation graphique et l'administration graphique.
Tout (développement et administration) peut se faire par des panneaux (à la WINDOWS).

On peut faire des applications entières presque sans écrire une ligne de code:

- DB création, destruction et paramétrisation des DP et DPE
- description des interfaces et de leurs comportements
- administration des projets

C'est vrai pour des actions simples.

Mon expérience est que plus on avance et moins on utilise ces possibilités pour préserver la réutilisation de code.

Attention: ETM appelle programmation graphique « objet » un modèle qui ne l'est pas.
Il s'agit de définir des objets graphiques paramétrés dont les valeurs des paramètres sont fixées au moment de la construction de l'interface.

Joint COntrol Project

Project mandate:

"To develop a common Framework and components for detector control of the LHC experiments and to define the required long-term support."

http://itcobe.web.cern.ch/itcobe/Projects/Framework/Download/welcome.html

Liste des principaux outils:

Installer, Access Control, Alarm Handling, Central Log Viewer, Trending Tools,
Configuration DB,
Control Hierarchy (Finite State Machine),
DIM (service publish/subscribe au dessus de TCP/IP)

Liste des principaux pilotes:

Alim Wiener, Alim Caen, Alim ISEG,
Rack Control
ELMB (µcontroller CAN pour ATLAS), PC monitoring,

PVSS dans LHCb

LHCb est l'expérience qui utilise le plus PVSS car elle l'utilise pour:

- slow control
- run control
- pilotage des triggers (L0 et ferme HLT)
- TTC et contrôle du timing

Les autres expériences ne l'utilisent que pour le slow control.
Le framework PVSS LHCb est riche fwCCPC, fwSPEC, fwHw ...

Dans LHCb PVSS c'est:

- **~150 machines** (~100 Linux ~50 WINDOWS) en réseau pour l'Electronic Control System
- en moyenne 2000 DP par machine avec en moyenne 40 DPE soit ~**12 000 000 DPE**
- **400 serveurs DIM** fournissant 900000 services
- PVSS en version actuel est limité à 512 systèmes connectés (mais ça doit changer)

PVSS et CERN

Le CERN est habilité a distribuer des licences aux laboratoires d'HEP pour des expériences approuvée ou reconnues par le CERN.

Il y a une équipe de support au CERN (8 personnes) qui gère l'interface avec ETM et les utilisateurs.

http://itcobe.web.cern.ch/itcobe/Services/Pvss/welcome.html

Tarifs indicatifs obtenus début 2009:

Micro Project: 1 Server for max. 1'500 IOs, 3Clients, Para, RDB-Manager,all standard drivers **CHF 2'500.00**

Small Project: 1 Server for max. 8'000 IOs, 5Clients, Para, RDB-Manager,all standard drivers **CHF 7'900.00**

Medium Project: 2 Servers for total 40'000 IOs, 10Clients, 2 Distribution Managers, Para, RDB-Manager, all standard drivers **CHF 25'300.00**

Large Project: 3 to 10 Servers unlimited number of IOs, 15 Clients, 3 to 10 Distribution Managers, Para, RDB-Manager, all standard drivers **CHF 61'000.00**

Extra Large Project: on quote « **list -60%** »