



*Institut  
Supérieur  
d'Informatique de  
Modélisation et de leurs  
Applications*

*Complexe des Cézeaux  
BP 125 63173 Aubière  
Cedex*

*Organisation  
Européenne pour la  
Recherche Nucléaire*

*CERN - CH1211 Genève  
23  
Switzerland*



## *Rapport de Stage 2<sup>ème</sup> année Filière 1*

CERN-THESIS-2006-043  
21/09/2006  


---

# Développement d'une interface graphique et implémentation d'une procédure de tests automatisés

---

Présenté par : Maha LAHJOUJI

Tuteur entreprise : Niko NEUFELD  
Tuteur ISIMA : Emmanuel MESNARD

Avril – Septembre 2006

# Remerciements

Tout d'abord je tiens à remercier mon tuteur au CERN, Niko NEUFELD, pour les outils qu'il m'a fournis ainsi que pour son écoute et ses suggestions. Je le remercie de m'avoir expliquée le cadre technique dans lequel se situe mon travail et de m'avoir intégrée dans son équipe.

Je tiens également à remercier mon responsable à l'ISIMA, Emmanuel MESNARD, pour avoir accordé une grande attention au déroulement de mon stage et pour être venu me rendre visite au CERN.

Je tiens à remercier Richard JACOBSSON pour sa coopération dans la réalisation de la deuxième partie de mon travail et Clara GASPAR pour son aide technique.

Enfin je remercie toutes les personnes qui m'ont aidée dans l'avancement de mon stage, je remercie en particulier Lana ABADIE pour ses conseils techniques et Beat JOST pour l'intérêt qu'il a porté à mon travail.

# Résumé

Dans le cadre de l'expérience LHCb menée par le CERN, des **particules** sont accélérées dans un environnement magnétique très intense. L'acquisition de certaines grandeurs physiques se fait grâce au composant **ELMB**.

Les particules accélérées rentrent en phase de collision et génèrent un flux énorme de données. Après avoir été filtrées ces données sont traitées par des cartes **TELL1** faisant partie de l'électronique du **Font-End** et enfin acheminées vers le système de stockage **DAQ**.

Une première partie de mon travail consiste à créer un serveur pour **calibrer** les valeurs magnétiques et permettre leur visualisation sur une **interface graphique**. La deuxième partie porte sur l'implémentation d'une **procédure automatisée** pour tester le fonctionnement interne des cartes TELL1 d'une part et de vérifier leur communication avec les autres composants du système d'autre part. La supervision des cartes est assurée grâce à un logiciel de type **SCADA**.

Les outils réalisés ont été testés et répondent aux attentes de mon stage. Ils peuvent toutefois être améliorés au niveau de leurs temps de réponse. Quant à la procédure de vérification des cartes TELL1, elle pourrait s'adapter aux futures exigences de l'expérience en incluant de nouveaux tests.

## Mots clés:

Particules, ELMB, TELL1, Front-End, DAQ, calibrer, interface graphique, procédure automatisée, SCADA.

# Abstract

For the LHCb experiment done at CERN, **particles** are accelerated and their collision products are separated in a very intense magnetic field. Acquisition of some physical variables is realised by an **ELMB** component.

The measurements of the particles in the detector create a great amount of data. This data are filtered and processed by **TELL1** cards which are part of the **Front-End** electronics. Finally it is given to the acquisition system **DAQ**.

The first part of my work consists in creating a **calibration server** for the magnetic field values and displaying them in a **graphical interface**. The second one concerns the development of a TELL1's automated **test procedure** and to check their communication with the environment. The cards are supervised by a **SCADA** software.

The components I have created have been tested and fulfil all requirements. They can be improved concerning the execution time. The TELL1's test processing could integrate other tests in the future.

## Key words:

Particles, ELMB, TELL1, Front-End, DAQ, calibration server, graphical interface, automatic test procedure, SCADA.

# Table des matières

Remerciements .....	2
Résumé .....	3
Abstract .....	3
Table des matières .....	4
Table des figures .....	6
Introduction .....	7
1. Présentation générale.....	8
1.1 CERN .....	8
1.2 LHCb .....	9
1.2.1 Objectif.....	9
1.2.2 Réalisation technique de l'expérience LHCb .....	9
1.3 Description du matériel utilisé .....	12
1.3.1 Composant d'acquisition du champ magnétique : ELMB .....	12
1.3.2 Carte de traitement des données : TELL1 .....	13
2. Réalisation d'une interface graphique pour la calibration et la visualisation du champ magnétique .....	15
2.1 Prise en main des outils logiciels .....	16
2.1.1 PVSS .....	16
2.1.2 DIM .....	19
2.2 Stratégie de résolution.....	21
2.2.1 Configuration du système.....	21
2.2.2 Démarche suivie et développement de l'interface graphique .....	24
3. Implémentation d'une procédure automatisée de test et de supervision des cartes TELL1 .....	30
3.1 Présentation du cadre technique du projet et étude de l'existant .....	32
3.1.1 Le système TFC et le framework lbTFC.....	32
3.1.2 Le framework fwCcpc.....	35
3.2 Planification et réalisation des tests .....	36
3.2.1 Conception des étapes de la procédure de test automatisé.....	36
3.2.2 Développement d'une librairie de fonctions .....	37
3.3 Développement de l'interface graphique .....	38
3.3.1 Initialisation et configuration des cartes TELL1 .....	38
3.3.2 Configuration de HUGIN.....	41
4. Résultats et discussion.....	42
4.1 Résultats de l'outil de calibration et de visualisation du champ magnétique.....	42
4.1.1 Test du serveur de calibration .....	42
4.1.2 Test de l'interface graphique.....	43
4.1.3 Test de la procédure de sauvegarde.....	44
4.1.4 Création du framework.....	45
4.2 Résultats de la procédure de test des cartes TELL1 .....	45
4.2.1 Communication avec les cartes TELL1 .....	45
4.2.2 Interface graphique.....	46
4.2.3 Enregistrement des résultats finaux.....	49
4.3 Discussion .....	50
4.3.1 Avancement du projet .....	50
4.3.2 Enrichissement personnel.....	50
Conclusion.....	51
Glossaire.....	52

Références bibliographiques .....	54
<i>Annexe A</i> .....	56
A.1 Boucle while du serveur de calibration .....	56
A.2 Description du composant ELMB .....	57
<i>Annexe B</i> .....	58
B.1 Détails de l'architecture du système TFC.....	58
B.2 Prototypes des fonctions de la librairie tell1.ctf .....	59
B.3 Principales fonctions pour la communication avec HUGIN .....	60
B.4 Adresses utiles pour la procédure de test des cartes .....	61
B.4 .1 Accès au registre d'horloge sur le bus I2C .....	61
B.4.2 Registre throttle pour communiquer avec HUGIN.....	61
B.4.3 Registre utilisé pour vérifier les quatre ports Ethernet des TELL1 .....	62

# Table des figures

Figure 1. 1: Photographie aérienne du CERN .....	8
Figure 1. 2: Le détecteur du LHCb .....	10
Figure 1. 3: Face intérieure de la carte mère du ELMB .....	12
Figure 1. 4: Photographie de la carte CCPC .....	13
Figure 1. 5: Photographie de la carte TELL1 .....	14
Figure 2. 6: Dimensions de l'aimant de séparation des particules. ....	15
Figure 2. 7: La console de PVSS.....	17
Figure 2. 8: Hiérarchie des managers de PVSS .....	18
Figure 2. 9: Principe d'interconnexion avec DIM.....	19
Figure 2. 10: Architecture software du protocole CANopen .....	22
Figure 2. 11: Fichier de configuration du OPC .....	23
Figure 2. 12: Commande de DIM .....	25
Figure 2. 13: Processus de communication des données par DIM.....	26
Figure 2. 14: Principe d'exécution des commandes par le serveur.....	27
Figure 2. 15: Algorithme du serveur de calibration .....	28
Figure 2. 16: DataPoint de fixation des instants de sauvegarde .....	29
Figure 3. 17: Structure du système de synchronisation du LHCb .....	31
Figure 3. 18: Fonctionnement du système TFC .....	33
Figure 3. 19: Interface graphique du système TFC .....	34
Figure 3. 20: Interface graphique de la carte CCPC .....	35
Figure 3. 21: Description de la procédure de test des TELL1 .....	36
Figure 3. 22: Ordonnancement des TELL1 .....	40
Figure 3. 23: Configuration des entrées de HUGIN.....	41
Figure 3. 24: Trace du traitement du serveur de calibration .....	42
Figure 3. 25: Fenêtre de visualisation des valeurs calibrées .....	43
Figure 3. 26: Fichier de sauvegarde des données calibrées.....	44
Figure 3. 27: Communication avec la carte TELL1 .....	46
Figure 3. 28: Interface graphique de test des cartes TELL1 .....	48
Figure 3. 29: Trace des résultats de la procédure de test.....	49
Figure A. 30: Code de la boucle while du serveur de calibration .....	56
Figure A. 31: Fiche de description du ELMB .....	57
Figure B. 32: Architecture du système TFC .....	58
Figure B. 33: Prototypes des fonctions de la librairie tell1.ctl.....	59
Figure B. 34: Prototypes des fonctions de communication avec HUGIN .....	60

# Introduction

La physique des particules est une science qui a donné lieu à des découvertes de très grande utilité. Cette science continue à explorer les mystères des plus petits constituants de la matière. Dans ce cadre le CERN (Organisation européenne pour la Recherche Nucléaire) mène des expériences d'ampleur internationale dans l'objectif de donner naissance à des nouvelles substances dans des conditions qui défient les normes de la physique classique. Les résultats de ces expériences devraient avoir lieu en 2007.

Des particules sont accélérées et séparées dans un champ magnétique pour subir des collisions aux points où se déroulent les quatre expériences principales du CERN. Les valeurs du champ magnétique sont mesurées grâce au composant ELMB qui supporte des conditions extrêmes de radiation et de variations du champ magnétique.

Le ELMB envoie les valeurs mesurées à un serveur « OPC » qui gère la communication entre le Hardware et le logiciel de supervision PVSS adopté par le CERN. Ces valeurs dépendent de certains paramètres physiques et doivent subir une calibration.

La première partie de mon travail consiste à faire l'acquisition, la calibration et la sauvegarde de ces données en utilisant un protocole d'interconnexion développé par le CERN basé sur le concept client serveur.

Après la collision des particules, les détecteurs de l'expérience LHCb effectuent les mesures des paramètres fixés par les chercheurs et envoient les données à un système de filtrage qui réalise une première réduction du flux reçu. Les événements acceptés par ce niveau sont alors traités par des cartes TELL1.

Les cartes TELL1 qui fonctionnent avec la même fréquence d'horloge que l'ensemble des composants électroniques de l'expérience reçoivent les décisions du système central TFC d'une part et envoient les données au système d'acquisition DAQ sur un réseau Ethernet d'autre part.

A ce niveau intervient la seconde partie de mon travail, il s'agit de réaliser une procédure pour tester le fonctionnement des cartes TELL1 en lisant les données dans les registres correspondant aux différentes étapes du test et vérifier la circulation des données sur leurs ports de sortie.



# 1. Présentation générale

## 1.1 CERN

Fondé en 1954 par 20 états européens, le CERN (Conseil Européen pour la Recherche Nucléaire) actuellement nommé Organisation Européenne pour la Recherche Nucléaire est le plus grand laboratoire des particules au monde. Il est situé sur la frontière franco-suisse aux environs de Genève [CERN].

Bien qu'il soit créé par 20 pays, le CERN est la destination des passionnés de la physique des particules à l'échelle mondiale. Plus de 80 nationalités sont présentes au sein du CERN en tant que chercheurs, ingénieurs, techniciens, ouvriers et étudiants. 2500 personnes sont mobilisées en alliance avec 6500 scientifiques dans un objectif purement scientifique, ce chiffre représente la moitié des physiciens des particules au monde. 500 universités et instituts coopèrent avec le CERN.

Le but des expériences menées par les physiciens du CERN est de dévoiler les mystères des plus petites substances dans l'espoir de découvrir les composants fondamentaux de la matière à une échelle encore plus petite que celle connue jusqu'à présent. Le moyen est d'accélérer des particules chargées au point d'atteindre une vitesse proche de celle de la lumière grâce à trois accélérateurs gigantesques. Une collision entre ces particules accélérées pourrait donner naissance à des nouvelles particules.

Quatre grandes expériences LHCb, Atlas, Alice et CMS se préparent simultanément chacune étudie un aspect précis des particules lors de l'accélération et de la collision. Elles ont lieu en France et en Suisse sur un périphérique circulaire d'un diamètre de 27 Km.

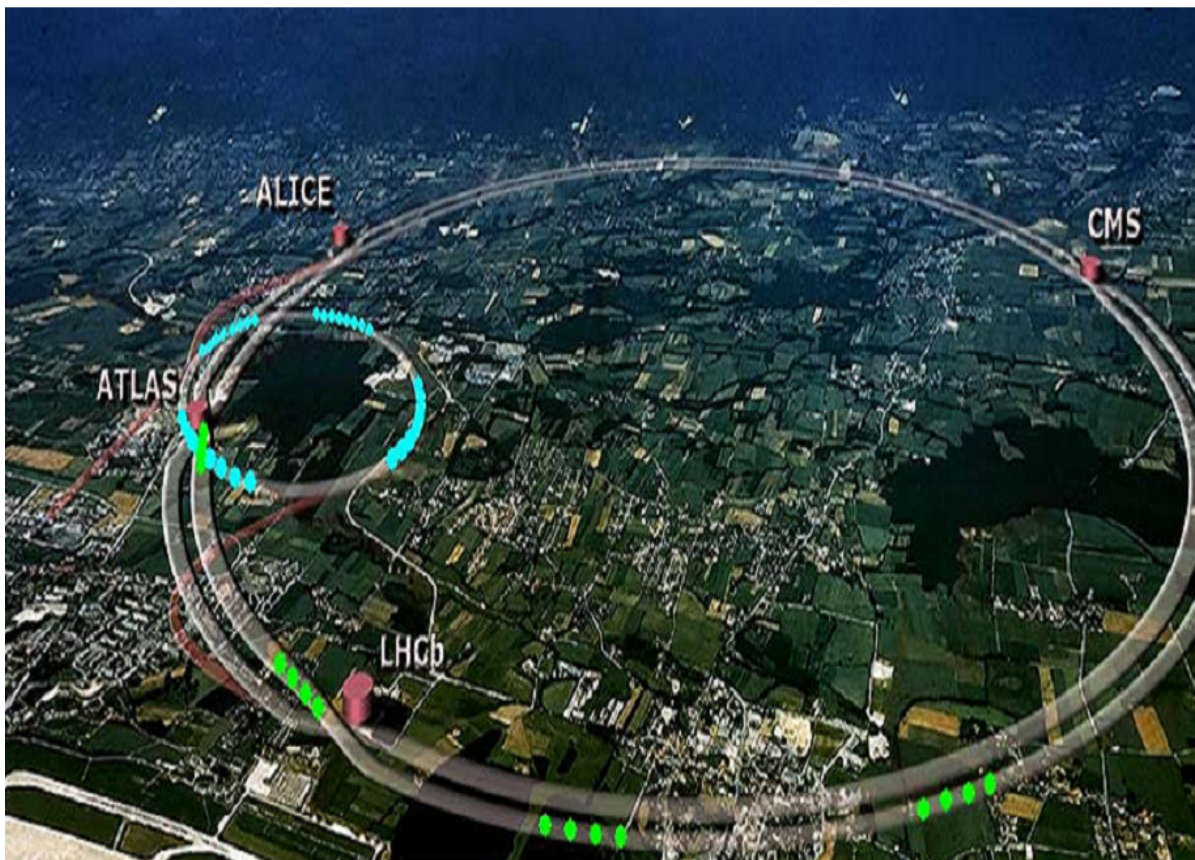


Figure 1. 1: Photographie aérienne du CERN



## 1.2 LHCb

### 1.2.1 Objectif

Le LHC (Large Hadron Collider) est le plus grand accélérateur des particules au monde (27 Km de diamètre), il sera mis en marche en 2007 [LHCb].

Cet accélérateur a pour but de répondre à une grande question sur l'origine de la création de l'Univers en essayant de faire apparaître le « Boson de Higgs », cette particule décrite par le Modèle Standard de la physique et dont la découverte pourra dévoiler l'origine de la matière.

D'après la théorie, l'énergie extraordinaire du Big Bang s'est condensée en quantité équivalente de particules et d'antiparticules et par conséquent l'Univers est formé à la base d'un équilibre de matière et d'antimatière mais suite à une violation de cette symétrie (CP : Charge and Parity symmetry) la matière a complètement dominé l'antimatière. Quelle est la cause de cette violation ?

Certains chercheurs ont développé l'hypothèse de l'existence de l'antimatière avec la même quantité que la matière mais quelque part dans l'univers. Une autre hypothèse consiste à dire que la matière et l'antimatière existent avec des quantités différentes et ne sont pas complètement symétriques, en effet la transformation de l'énergie en matière et en antimatière aurait favorisé la matière avec un pourcentage très faible de 1 proton de plus pour chaque milliard d'antiprotons ce qui a donné lieu à la domination de celle-ci.

Les chercheurs du CERN ont adopté l'hypothèse de « l'énergie faible » due aux quarks. De nombreuses recherches sont menées pour éclaircir la théorie mais sans aboutir à des résultats satisfaisants.

L'expérience LHCb a pour objectif de créer des nouvelles particules. Toutes les 25 ns une collision entre deux protons aura lieu à des emplacements fixes dans l'accélérateur. Cette collision devra générer 14 TeV d'énergie qui pourrait donner naissance aux quarks.

Les expériences voisines du LHCb à savoir Atlas, Alice et CMS étudient d'autres aspects de la physique des particules. Ces expériences disposent de détecteurs d'une très haute précision, le détecteur du LHCb doit capter et envoyer les données avec une fréquence de 40 MHz. Statistiquement 4 Téraoctets de données doivent être traités chaque seconde ce qui présente un véritable défi technique.

### 1.2.2 Réalisation technique de l'expérience LHCb

#### a. Accélération des particules

Afin de créer les collisions Proton-Proton qui pourront générer une très grande énergie, les particules doivent être accélérées au point d'atteindre une vitesse proche de celle de la lumière.

Les particules subissent une première accélération classique permettant de créer un choc contre la matière pour produire des électrons et des protons. Ensuite ces particules sont séparées par un aimant, les charges positives et les charges négatives prennent des trajectoires de sens différents lors de la séparation. Par la suite les protons sont accumulés dans un accumulateur qui les éjecte dans le premier accélérateur.

Le CERN possède deux grands accélérateurs intermédiaires:

- **PS** : (Proton Synchrotron) mis en service en 1959, c'est le premier accélérateur, il reçoit les particules accumulées dans l'accumulateur.
- **SPS** : (Super Proton Synchrotron) mis en service en 1976, il produit des particules portant le nom « premiers bosons de jauge massifs ».

Ces deux accélérateurs permettent d'atteindre une énergie de l'ordre de 440 GeV. Le but de la nouvelle physique est d'atteindre une énergie à l'échelle du TeV. Dans cette optique le **LHC** (Large Hadron Collider) fut construit, sous forme d'un anneau circulaire de 27 Km de circonférence, il permet de créer des collisions en quatre points où se déroulent les expériences du LHC (LHCb, Atlas, Alice et CMS).

### **b. Acquisition des données**

4 Téraoctets de données doivent être traitées par seconde, ce flux énorme de données envoyées par le détecteur n'est pas entièrement utile car il contient beaucoup de parasites dus à des collisions inintéressantes dans le sens de ce que les chercheurs veulent prouver d'où la nécessité de filtrer les données.

En effet les collisions qui seront exploitées par l'expérience ne se produiront qu'avec une fréquence de 15 Hz par rapport à 40 MHz de fréquence des collisions réellement produites.

La figure 1.2 montre le détecteur utilisé par le LHCb, il fait 20 m de longueur et 10 m de largeur.

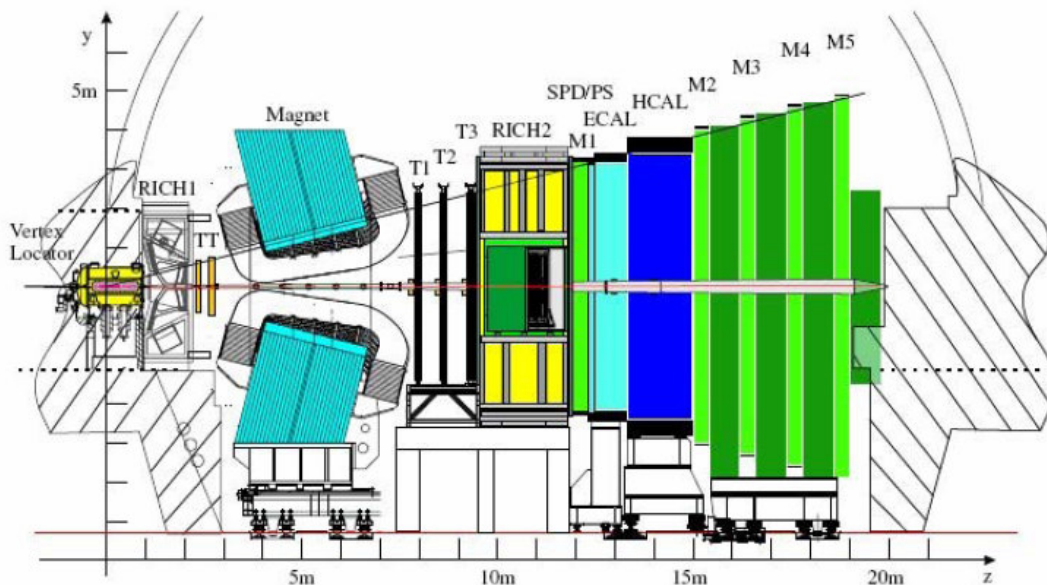


Figure 1. 2: Le détecteur du LHCb

Afin d'étudier la violation du CP, le LHCb intègre des systèmes de contrôle et de détection très précis. Le détecteur du LHCb est constitué de plusieurs sous détecteurs qui peuvent tourner indépendamment, nous exposons les différentes parties de ce détecteur :

- **VELO** : (Vertex Locator) ce détecteur permet de déterminer l'emplacement des collisions dans le LHC et les événements qui peuvent causer la violation du CP.
- **PUS** : (Pile-Up system) il se situe au dessus du VELO, son rôle est de préciser où se déroulent les collisions simultanées Proton-Proton.
- **RICH** : (Ring Imaging Cherenkov) il est composé de deux parties, RICH1 se trouve avant l'aimant et RICH2 après l'aimant. Son rôle est d'identifier les particules chargées ainsi que leur quantité de mouvement.
- **TRACKERS** : est composé de trois sous détecteurs. Ils sont situés entre l'aimant et le RICH2. Il est utilisé pour fournir des mesures précises sur les quantités de mouvement des particules chargées.
- **CALORIMETERS** : permettent l'identification des électrons, hadrons et photons. Ils fournissent aussi des mesures sur leur énergie et leur position. Ils sont en nombre de quatre.
- **MUON** : son objectif est de permettre l'identification des muons.

Le système de supervision de l'expérience est composé de trois sous-systèmes fondamentaux :

- **ECS** : contrôle et surveille le détecteur du LHCb.
- **TFC** : est le système central dont le rôle fondamental est de synchroniser toutes les composants électroniques de l'expérience.
- **DAQ** : ce système sélectionne les données qui seront envoyées et stockées dans les serveurs de stockage.

## 1.3 Description du matériel utilisé

### 1.3.1 Composant d'acquisition du champ magnétique : ELMB

Dans la partie de l'accélérateur LHC où se trouve l'aimant de séparation des particules, les physiciens ont besoin d'un outil de mesure du champ magnétique, de la température et d'autres grandeurs physiques. Le ELMB (Embedded Local Monitor Board) est un composant de mesure qui contient un processeur embarqué sur une carte interne [ELMB].

Le ELMB est conçu spécialement pour le CERN, grâce à sa grande résistance aux radiations et aux fortes variations du champ magnétique il permet de fournir des mesures précises sur l'environnement où sont accélérées les particules. Chaque ELMB peut comporter 1 à 4 capteurs magnétiques.

Le ELMB peut supporter une radiation allant jusqu'à 5 Gy et  $3.10^{10}$  neutrons/cm<sup>2</sup>, sa durée de vie est d'environ 10 ans dans un champ magnétique pouvant s'élever à 1.5 T. Cette valeur est équivalente à 30000 fois la valeur du champ magnétique terrestre.

#### Outils d'installation et de configuration du ELMB :

**National Instruments PCI-CAN/2 Interface :** cette interface sert d'intermédiaire entre le ELMB et le serveur qui reçoit les données, La carte PCI-CAN-2 est alimentée grâce au câble CAN qui est relié aux composants ELMB128.

**Kvaser PCI CAN Card Interface:** La configuration de la carte PCI CAN se fait via le panneau de control "control panel" sous l'entité « CAN Hardware ».

**Carte mère du ELMB:** cette carte permet de tester le ELMB, elle contient une centaine de points de connexion avec le ELMB et 64 chaînes ADC pour les adaptateurs. Elle peut être logée sur une surface de 80x180 mm.

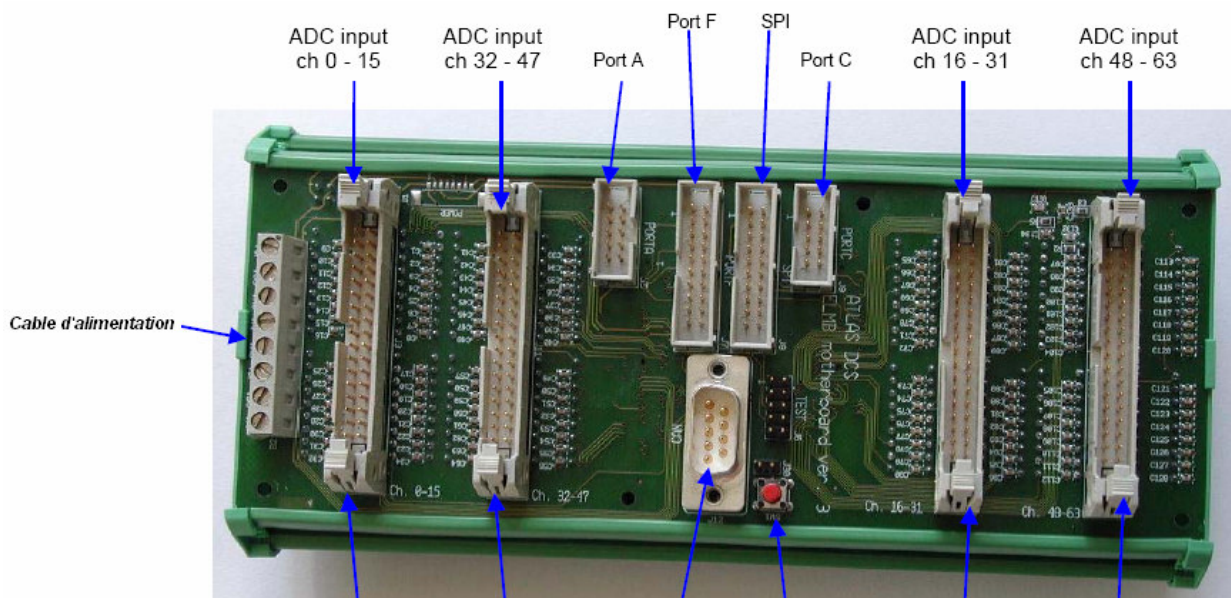


Figure 1. 3: Face intérieure de la carte mère du ELMB



### 1.3.2 Carte de traitement des données : TELL1

La carte TELL1 (Trigger Electronics and L1 board) est utilisée dans l'expérience LHCb pour le traitement des données qui proviennent des collisions (Proton-Proton) [TELL1]. Cette carte intervient au second niveau du traitement. En effet, après la collision les données contiennent beaucoup d'informations inutiles. Le LHCb procède donc à un filtrage selon des niveaux.

Le L0 trigger est un premier niveau de filtrage qualifié de matériel, il permet une réduction de la fréquence des données de 40 MHz à 1 MHz.

Le L1 trigger zero suppression est un niveau de filtrage intermédiaire qui comme son nom l'indique il supprime les zéros inutiles.

Le HLT (High Level Trigger) est le deuxième niveau de filtrage qualifié de logiciel, il réduit la fréquence de 1 MHz à quelques KHz. Les cartes TELL1 appartiennent à ce niveau de filtrage, elles effectuent les derniers traitements avant d'envoyer les données au système DAQ (système de stockage des données) via des switches. L'électronique utilisée dans cette partie s'appelle « Front-End ».

La carte TELL1 possède comme la plupart des cartes utilisées dans le LHCb une petite carte CCPC (Credit Card Personal Computer) qui fournit aux utilisateurs une facilité d'accès aux différents composants des TELL1 [CCPC].

Les CCPC sont de la taille d'une carte de crédit et communiquent grâce aux signaux envoyés et reçus par l'intermédiaire des quatre bus : I<sup>2</sup>C, JTAG, LBUS et GBE.

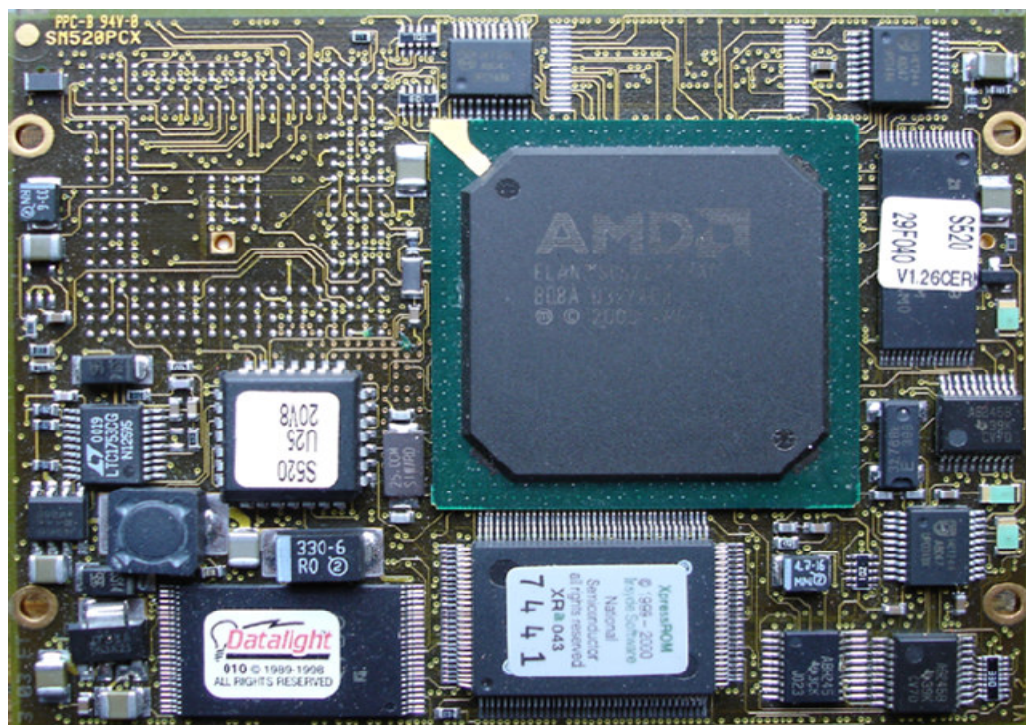


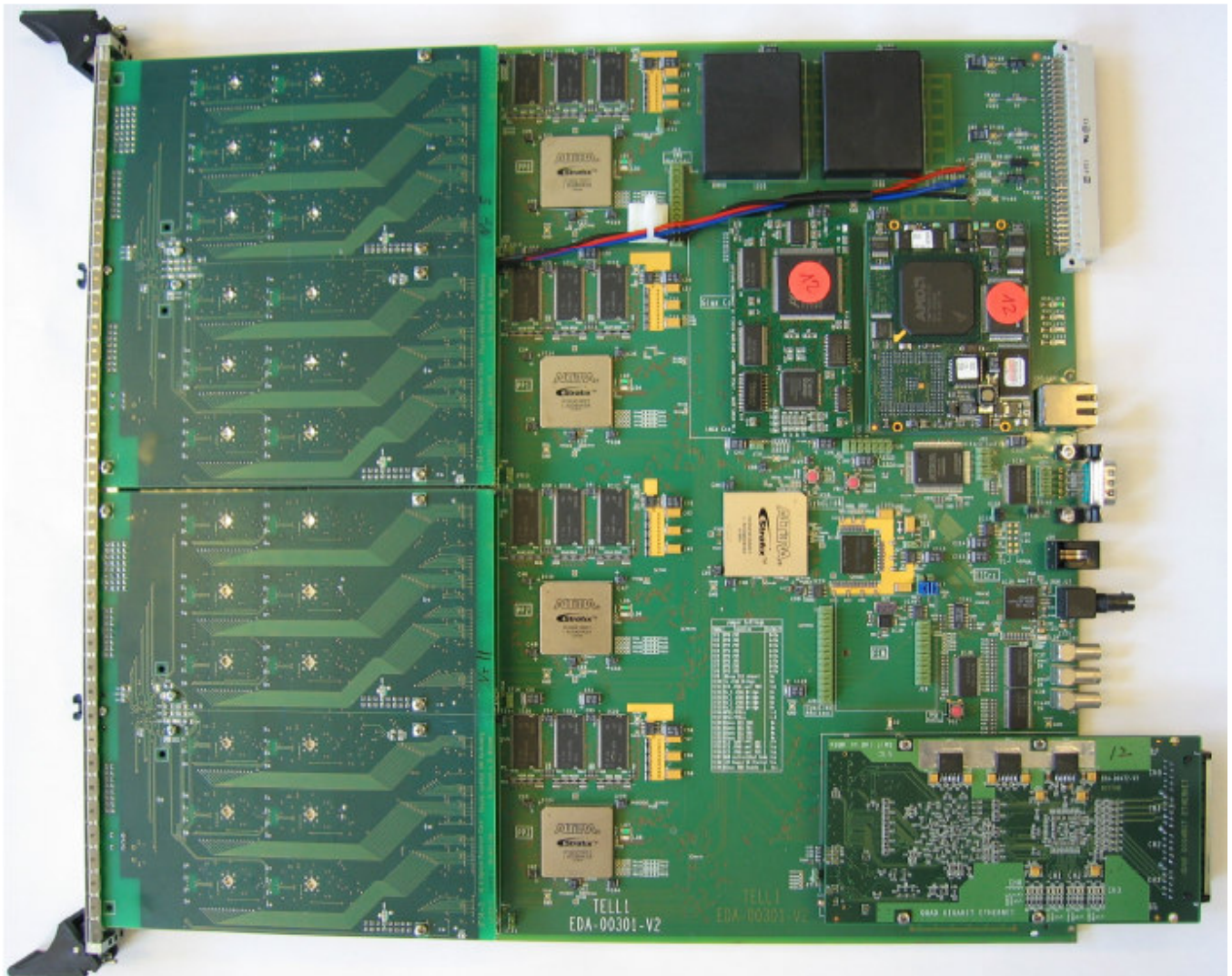
Figure 1. 4: Photographie de la carte CCPC

La connexion entre les TELL1 et le système de stockage est établie grâce à deux types de switches et routeurs :

Un grand switch qui comporte 1200 ports est connecté à la sortie des cartes TELL1.

Des switches de distribution au nombre de 50, leurs entrées sont reliées au grand switch et leurs sorties aux 40 ordinateurs qui traiteront les données par la suite.

L'acheminement de ces données fait appel à un protocole de transfert créé par le CERN.



*Figure 1. 5: Photographie de la carte TELL1*



## 2. Réalisation d'une interface graphique pour la calibration et la visualisation du champ magnétique

Dans l'accélérateur LHC, les particules sont séparées par un aimant. Les charges positives et les charges négatives sont déviées dans deux sens opposés. Le rayon de courbure de chaque particule est proportionnel à son énergie. Pour avoir une idée sur cette énergie il faut mesurer la valeur du champ magnétique qui est à l'origine d'une telle déviation. Une mesure doit avoir lieu toutes les 20 s.

Chacune des quatre sondes portées par le ELMB fournit la mesure de 4 variables : les valeurs du champ suivant les trois directions de l'espace et la valeur de la température au même point. En effet, la valeur du champ magnétique au même point de mesure dépend de la température et de l'effet Hall.

Expérimentalement d'autres facteurs physiques interviennent pour que deux mesures au même point et à la même température donnent des valeurs de champ magnétique différentes. Pour améliorer la précision des résultats nous avons recours à une référence de valeurs calibrées. Une étude expérimentale sur des centaines de sondes prouve que les mesures sont différentes selon le type de la sonde utilisée.

Mathématiquement la valeur absolue du champ magnétique se calcule à partir de  $B_x$ ,  $B_y$  et  $B_z$  avec la formule suivante :

$$|B| = (B_x^2 + B_y^2 + B_z^2)^{1/2}$$

Grâce aux outils Graphiques de PVSS, nous pouvons afficher la courbe du champ magnétique avant et après la calibration.

*Figure 2. 6: Dimensions de l'aimant de séparation des particules.*

## 2.1 Prise en main des outils logiciels

### 2.1.1 PVSS

Créé par la société autrichienne ETM, PVSS est un outil de contrôle et d'acquisition de données, c'est le logiciel adopté par le CERN pour la supervision des différents systèmes dans toutes les expériences [PVSS].

La structure de données dans PVSS est modélisée par des DataPointsTypes (DPT), nous pouvons traduire les DPT en langage orienté objet par des Classes. Les instances de ces classes s'appellent des DataPoints (DP), chaque DPT contient des DataPoints Elements (DPE) ce sont les données membres.

Un DPE peut en contenir d'autre ceci revient à considérer le DPE comme une structure.

Le JCOP Frame Work (Joint Control Project) est le moyen d'écrire et de lire les données du hardware [JCOP]. Le CERN utilise un produit commercialisé pour établir la communication entre le hardware et PVSS, nous parlons des serveurs OPC que nous allons étudier plus tard et pour les équipements propres au CERN comme la carte TELL1, la communication se fait grâce à CAN-OPC, SPECS ou CCPC qui sont des protocoles développés par le LHCb.

PVSS offre des moyens pour modéliser la hiérarchie d'un système, HARDWARE et FSM sont les plus importants :

**HARDWARE**: permet de représenter la hiérarchie sous forme d'un arbre de DataPoints.

**FSM**: (Finit State Machine) est une machine d'états intégrée dans PVSS pour permettre de définir les états des DataPoints : opérationnel, prêt, en erreur ... ainsi que les actions associées à ces états.

Les principaux outils de travail sous PVSS sont :

- **Editeur graphique** : (Graphical editor) il permet de créer des Panels, ce sont des interfaces graphiques propres à l'utilisateur. Il offre le moyen d'utiliser les outils graphiques pour exécuter le code qui est implémenté derrière cette interface. Le code peut aussi être généré automatiquement grâce à une librairie de PVSS.
- **Control Script** : permet à l'utilisateur d'exécuter des codes en C plus certaines fonctions relatives à la SCADA (Supervisory Control And Data Acquisition) propre à PVSS, sans avoir à utiliser l'interface graphique du Graphical Editor.
- **Drivers** : ce sont des pilotes qui permettent à PVSS de communiquer avec les composants Hardware ou Software du système.
- **Para** : (Graphical Parameterization tool) est une schématisation de la base de donnée du projet, elle permet de créer, éditer, supprimer des DPTs et des DPs. Elle offre aussi la possibilité de relier les DataPoints par des fonctions. Le para propose à l'utilisateur

plusieurs possibilités pour configurer un DataPointElement, nous retenons les plus intéressantes entre elles :

- *Archiving* : c'est le système de sauvegarde des valeurs d'un DPE qui varient dans le temps, cela se fait via des classes d'archivage, ces classes peuvent effectuer la sauvegarde de 100 DPE simultanément depuis la création du projet. La sauvegarde peut être interrompue si le Archive Manager correspondant à cette classe est arrêté par l'utilisateur.
- *Alarm Activation and Handling* : c'est le système de gestion de l'alarme, le déclenchement se fait selon les paramètres fixés par l'utilisateur.
- *DataPoint Function* : cette fonction permet de relier les DataPoints en effectuant des calculs mathématiques.

PVSS est un logiciel très riche et parfaitement adapté aux systèmes faisant appel à beaucoup de contrôle et de connexions.

L'élément supérieur de contrôle d'un projet sous PVSS est la *console* qui est un ensemble de processus et d'applications que l'utilisateur peut configurer, arrêter, ou mettre en marche.

On peut aussi faire apparaître de nouveaux processus, ces processus portent le nom de *Managers*.

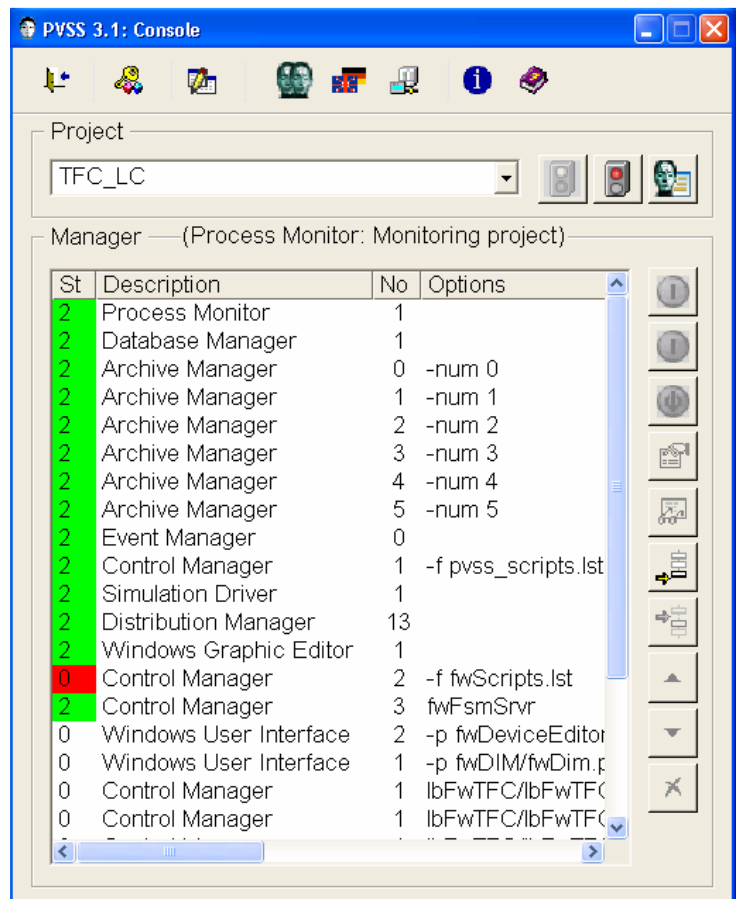


Figure 2. 7: La console de PVSS

Le manager principal est le event manager il joue le rôle du cœur du projet, il est directement relié aux pilotes (drivers).

La figure 2.8 montre la structure des managers sous PVSS, le Event manager (EVT) est le responsable de toutes les communications, il reçoit les données des drivers et les envoie à la base de données. Les UIM (managers d'interface utilisateur), communiquent avec la base de données sous le contrôle de l'utilisateur. Le Ctrl (manager de contrôle) exécute les scripts intégrés dans PVSS et le API manager(Application Programming Interface) permet à l'utilisateur d'exécuter ses propres codes C++ en utilisant une interface PVSS.

*Figure 2. 8: Hiérarchie des managers de PVSS*

### 2.1.2 DIM

Le système distribué de gestion de l'information DIM (Distributed Information Management) est un protocole de communication basé sur le concept client-serveur [DIM]. Il est spécialement conçu par le CERN pour gérer les échanges de données entre des systèmes distants qui interviennent dans ses différentes expériences.

*Figure 2. 9: Principe d'interconnexion avec DIM*

La communication entre les processus de DIM se fait via des DataPoints. PVSS représente le client qui envoie des commandes. Chaque commande est reliée à un DataPoint.

Le changement de l'une des DataPoints est pris en compte par DIM qui fait automatiquement sa mise à jour.

Une fois un DataPoint change de valeur, DIM exécute un envoi de la nouvelle commande vers le serveur qui peut tourner sur une machine distante. La connexion est établie grâce à la variable d'environnement DIM\_DNS\_NODE, cette variable est fixée par l'utilisateur qui doit fournir au client et au serveur le nom de la machine sur laquelle tourne DIM, la communication est dès lors établie. Il faut noter que sur une machine on ne peut avoir qu'un seul DIM manager.

Au niveau de la configuration de DIM il faut aussi affecter chaque service à un DataPoint. Dès que le serveur reçoit les commandes de DIM il les traite et renvoie les résultats de ce traitement à DIM qui les publie dans le DataPoints correspondant à chaque service. PVSS prend ensuite ces services pour effectuer un traitement.

Il existe des fonctions de DIM qui permettent d'exécuter des scripts dès le changement des valeurs de certaines DataPoints.

DIM offre un moyen de diagnostic sur une interface graphique indépendante de PVSS, nous parlons ici de DID (Distributed Information Display) c'est un outil qui permet de

visualiser la liste des serveurs mis en marche ainsi que la liste des services disponibles. Il donne aussi à l'utilisateur la possibilité de sélectionner un serveur particulier et d'accéder à sa liste de services ou de commandes.

DID est aussi un moyen pour tester le fonctionnement du serveur seul en permettant d'envoyer des commandes indépendamment de PVSS via son interface graphique et de lire le contenu du service où DIM publie les résultats.

DIM offre une librairie de fonctions qui peuvent être intégrées dans un code C++ ou Java, cette librairie est principalement composée de deux types de classes :

- *Classes du serveur* : à utiliser par les serveurs de DIM, elles implémentent des méthodes statiques relatives au serveurs et permettent de créer des serveurs, des services et des commandes ainsi qu'effectuer la mise à jours des services à partir d'un objet serveur.
- *Classes du client* : implémentent des méthodes relatives aux clients, contient plusieurs méthodes statiques, la principale est l'envoi d'une commande par un client à un serveur ou la demande d'informations à un serveur et leur mise à jour.



## 2.2 Stratégie de résolution

### 2.2.1 Configuration du système

Un projet sous PVSS peut recevoir ses données de plusieurs composants ELMB à la fois. Afin de parvenir à communiquer avec tous ces composants, un protocole dit « CANopen » est adopté pour brancher jusqu'à 127 ELMB sur le même bus « CAN-bus ». La connexion entre le bus et le serveur se fait via l'interface de la carte CAN.

#### a. Le protocole CANBus et CANopen

Pour établir la communication entre PVSS et le ELMB, un serveur « CANopen OPC server » traite les requêtes envoyées par PVSS qui joue dans ce cas le rôle d'un client OPC. Ce serveur est d'autre part relié au bus sur lequel il lit les données envoyées par le ELMB.

Le serveur OPC peut communiquer à la fois avec plusieurs clients.

Chaque nœud représentant un ELMB au niveau software envoie et reçoit des trames (n° de la chaîne, adresse, type de données, etc...) sur le bus CAN.

Une trame CAN est un nombre d'octets de données, un entête et un identifiant de l'objet qui communique COB-ID. Il existe 4 types de messages :

- **PDO** : *Process Data Objects* est utilisé pour des transfert de données avec haute priorité en temps réel.
- **SDO** : *Service Data Objects* ce sont des messages envoyés par l'entité, ils contiennent les données.
- **NMT** : *Network Management* ces messages sont envoyés par les services d'administration du projet dans le but de contrôler le réseau en entier et toutes ses entités. Ils ont la plus haute priorité puisqu'ils permettent de lancer et d'arrêter le fonctionnement sur le réseau du CANopen.

La figure suivante montre un système software géré par le protocole de communication CANopen, le cœur de ce système est le serveur CANopen OPC. D'une part, il a trois clients différents (un client PVSS, une application, et un autre serveur) et d'autre part il reçoit les données par une carte CAN qui communique directement au bus CAN sur lequel est branché un ELMB. Cette communication est surveillée par un système interactif de contrôle qui peut l'interrompre et la relancer à tout moment.

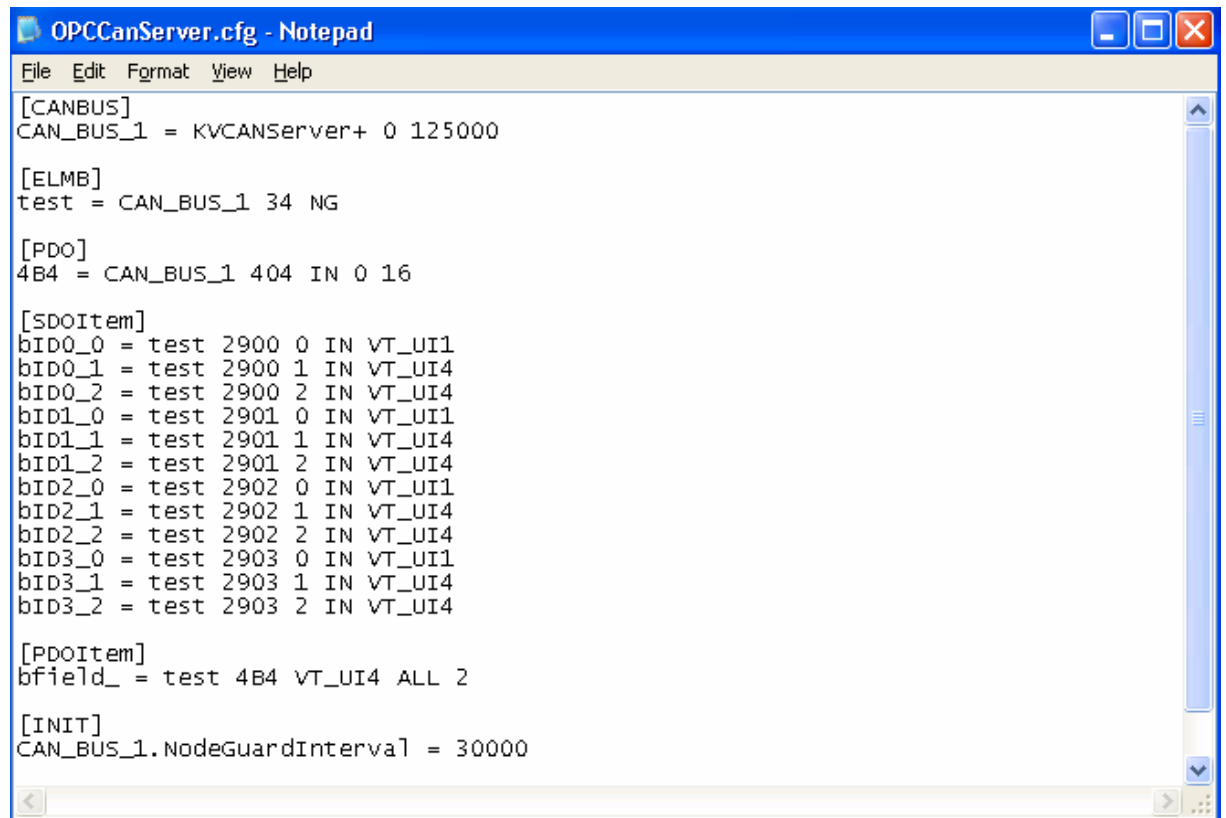
Il faut noter que le serveur CANopen OPC fonctionne uniquement sous le système d'exploitation Windows.

Figure 2. 10: Architecture software du protocole CANopen

**b. Configuration de l'espace d'adressage du serveur OPC**

Le serveur CANopen OPC lit toutes les informations nécessaires pour définir son espace d'adresses dans un fichier de configuration nommé OPCCanServer.cfg qui comporte six sections :

- [CANBUS] : décrit tous les bus CAN utilisés par le système.
- [DEVICE] : liste les ELMB avec leurs identifiants sur chacun des bus.
- [ELMB] : décrit les ELMBs telles qu'elles seront vues par PVSS. Le serveur en crée une par défaut.
- [SDOItem] : décrit la structure des messages de type SDO.
- [PDO] : définit le COB-ID des PDO.
- [PDOItem] : décrit les données envoyées par le ELMB en temps réel.
- [INIT] : définit les valeurs qui seront utilisées lors de l'initialisation dans l'espace d'adresses du OPC.



```
OPCCanServer.cfg - Notepad
File Edit Format View Help

[CANBUS]
CAN_BUS_1 = KVCANServer+ 0 125000

[ELMB]
test = CAN_BUS_1 34 NG

[PDO]
4B4 = CAN_BUS_1 404 IN 0 16

[SDOItem]
bID0_0 = test 2900 0 IN VT_UI1
bID0_1 = test 2900 1 IN VT_UI4
bID0_2 = test 2900 2 IN VT_UI4
bID1_0 = test 2901 0 IN VT_UI1
bID1_1 = test 2901 1 IN VT_UI4
bID1_2 = test 2901 2 IN VT_UI4
bID2_0 = test 2902 0 IN VT_UI1
bID2_1 = test 2902 1 IN VT_UI4
bID2_2 = test 2902 2 IN VT_UI4
bID3_0 = test 2903 0 IN VT_UI1
bID3_1 = test 2903 1 IN VT_UI4
bID3_2 = test 2903 2 IN VT_UI4

[PDOItem]
bfield_ = test 4B4 VT_UI4 ALL 2

[INIT]
CAN_BUS_1.NodeGuardInterval = 30000
```

Figure 2. 11: Fichier de configuration du OPC

**c. Définition des variable dans fwELMB**

fwELMB est un composant logiciel « framework component » créé par les concepteurs du ELMB. Il constitue la partie logicielle utilisée par PVSS pour communiquer avec le serveur OPC.

fwELMB permet de définir la structure de données relative aux composants matériels. Nous pouvons ainsi créer les variables qui reçoivent les données et les stockent dans la base de données du projet. Nous obtenons à la fin une structure arborescente identique à l'architecture matérielle du système.

## 2.2.2 Démarche suivie et développement de l'interface graphique

### a. Algorithme général

J'ai réalisé le travail demandé en suivant l'algorithme suivant :

- Un premier traitement a lieu dès la lecture par PVSS des valeurs envoyées par une sonde. La température et la valeur absolue calculées dans les *DataPoints* seront affichées dans l'interface graphique, l'identifiant de la sonde qui envoie les données sera déterminé grâce à la section SDOItem.
- PVSS, envoie 5 données (3 valeurs du champ, la température et l'identifiant de la sonde) sous forme d'une commande à DIM.
- Le serveur de calibration reçoit ces valeurs, fait appel à la fonction de calibration et envoie les trois valeurs calibrées du champ magnétique à DIM.
- DIM publie ces valeurs sous forme de services. Il met les valeurs dans des *DataPoints* de PVSS.
- PVSS calcule la valeur finale du champ magnétique, l'affiche, trace la courbe et enregistre les trois valeurs calibrées ainsi que la température dans une classe d'archivage.

### b. Processus de communication client/serveur

DIM établit l'interconnexion entre le client (PVSS) et le serveur distant de calibration. La communication des valeurs entre DIM et PVSS se fait via des DataPoints.

Le client enregistre les valeurs qu'il veut traiter dans l'une des quatre DataPoints suivantes :

- *commandDp1* : si les données sont envoyées par la première sonde.
- *commandDp2* : si les données sont envoyées par la deuxième sonde.
- *commandDp3* : si les données sont envoyées par la troisième sonde.
- *commandDp4* : si les données sont envoyées par la quatrième sonde.

DIM traduit les valeurs de ces DataPoints en commandes car dans la configuration de DIM, nous relierons chaque DataPoint à une commande. Par exemple, sur la figure suivante, la commande qui porte le nom CmndServMany/Cmnd\_Calibration est reliée à quatre DataPoints différents, dès que l'une des quatre change de valeur, ce changement est pris en compte par la commande.

Chaque commande contient la structure suivante :

- *inBx* : valeur non calibrée du champ suivant la direction x.
- *inBy* : valeur non calibrée du champ suivant la direction y.
- *inBz* : valeur non calibrée du champ suivant la direction z.
- *inTemp* : valeur de la température.
- *stringInfo* : détails utiles pour le serveur.
- *ident* : identifiant de la sonde qui envoie la commande.

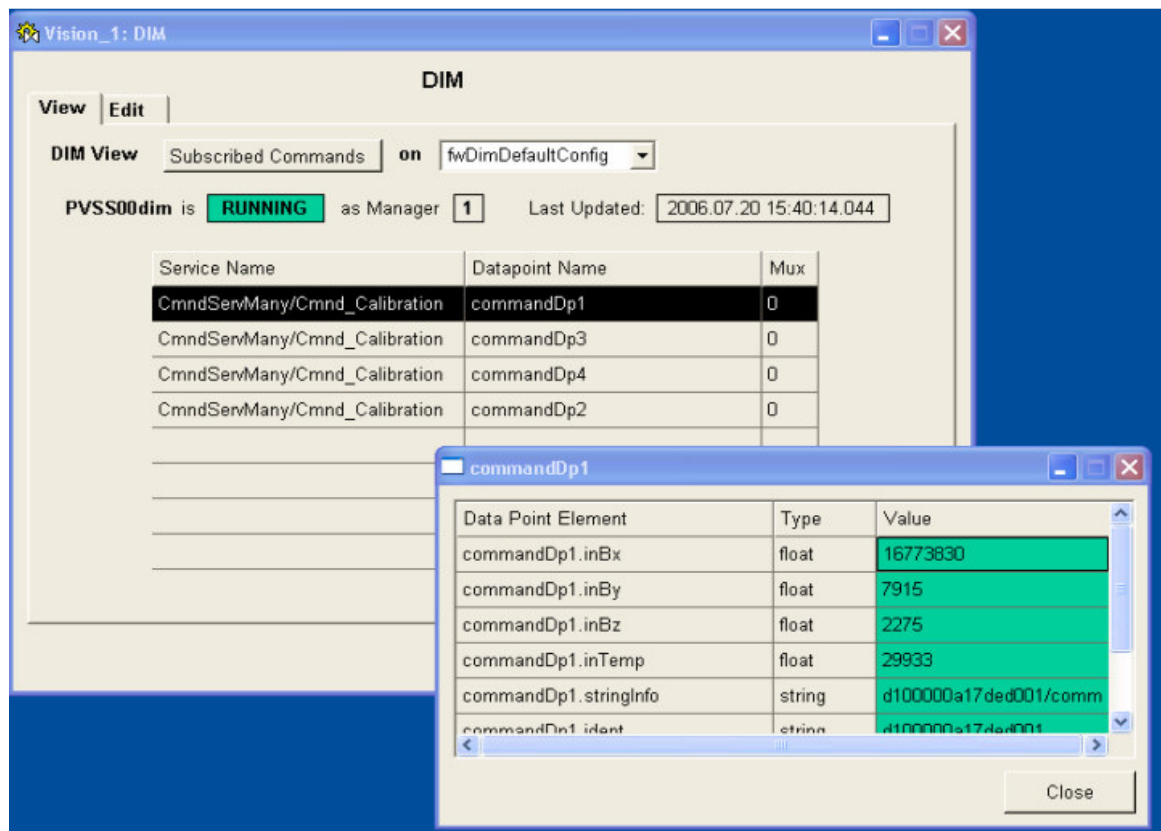


Figure 2. 12: Commande de DIM

Une fois les données traitées par le serveur, DIM les enregistre dans les DataPoints de PVSS. Ces requêtes s'appellent des services.

Les services sont au nombre de quatre et chacun communique avec un seul DataPoint. En effet, un service correspond à une seule sonde et dès que les valeurs de cette sonde sont traitées par le serveur celles-ci sont prises en compte par le service correspondant.

La figure 2.13 schématise l'architecture globale du processus de communication via DIM.

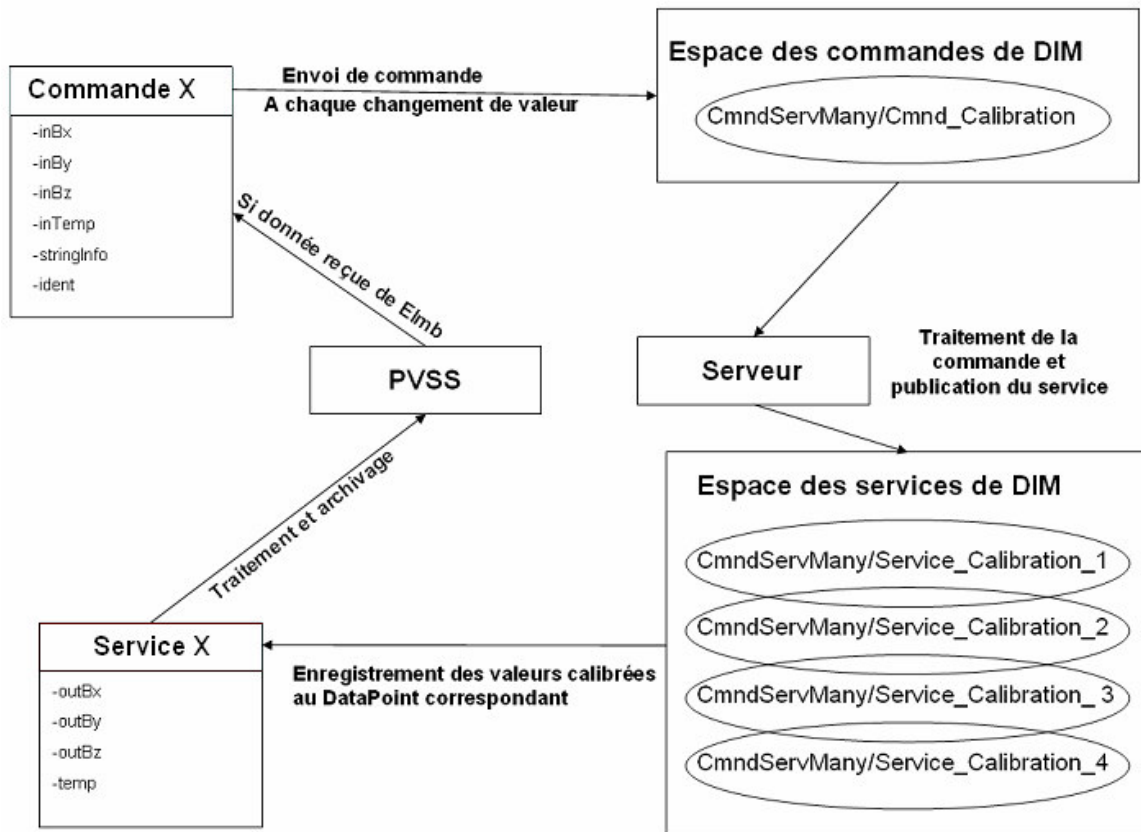


Figure 2. 13: Processus de communication des données par DIM



*c. Traitement des données par le serveur de calibration*

Le serveur traite les commandes envoyées par Dim et publie les résultats sous forme de services. Pour établir la communication entre Dim et le serveur j'ai utilisé une librairie de fonctions conçue pour Dim.

*Figure 2. 14: Principe d'exécution des commandes par le serveur*

La commande qui reçoit les données est une instance de la classe CmndCal qui hérite d'une classe définie dans la librairie de Dim. L'appel du constructeur de cette classe est fait automatiquement à la réception de la première commande et le call-back de la classe se fait avec une fréquence de 20 s.

Le serveur utilise l'algorithme développé sur la figure 2.15. Lors de son exécution Dim détecte de nouveaux services et les publie à PVSS. Dès que le composant ELMB effectue de nouvelles mesures du champ magnétique et de la température, PVSS envoie les données à Dim. Le constructeur de la classe commande est alors appelé avec les valeurs reçues.

La fonction de calibration appelée par le serveur, est le résultat de plusieurs centaines de tests sur des sondes. Cette fonction prend les paramètres suivant : les trois valeurs du champ mesurées, la température et le code identifiant de la sonde utilisée. Elle retourne les trois valeurs calibrées.

Il faut noter que cette fonction ne peut être utilisée que sous Linux alors que le serveur OPC tourne uniquement sous Windows. Cette contrainte explique l'intérêt de DIM.

*Figure 2. 15: Algorithme du serveur de calibration*

#### d. Enregistrement des données

L'affichage des valeurs calibrées permet à l'utilisateur de visualiser leur évolution dans le temps. Dès que l'interface graphique est fermée, il n'est plus possible de lire les valeurs antécédentes. L'origine du repère du temps étant située au moment du lancement de l'application, il est nécessaire d'enregistrer les événements depuis la création du projet.

PVSS offre la possibilité de créer une sauvegarde grâce à des classes internes qui permettent de gérer 100 *DataPoints* simultanément. Chaque classe est supervisée par un *manager* « archive manager ».

La lecture de l'enregistrement se fait via une fonction de PVSS qui affiche le contenu historique du *DataPoint*. En moyenne 1000 valeurs sont sauvegardées toutes les heures.

Pour offrir à l'utilisateur le choix de lire les données aux instants voulus. J'ai créé un *DataPoint* de type temps où on peut fixer les instants de début et de fin de sauvegarde. Une fois les instants du début et de fin sont fixés, l'utilisateur pour lancer la lecture des données entre ces deux instants. Un fichier Excel est alors mis à jour avec ces données.

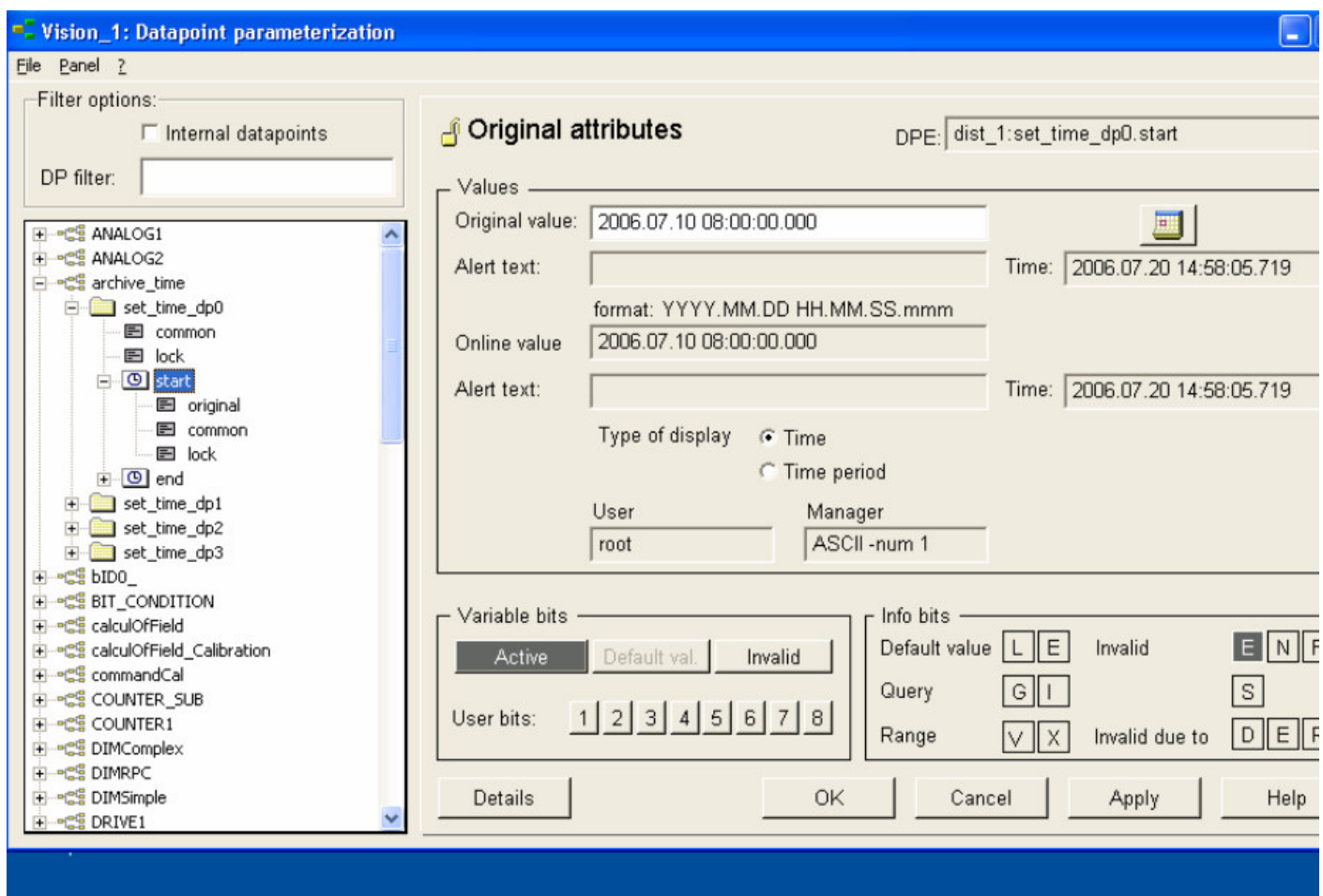


Figure 2. 16: *DataPoint* de fixation des instants de sauvegarde

### 3. Implémentation d'une procédure automatisée de test et de supervision des cartes TELL1

L'expérience LHCb utilise un détecteur composé de 6 sous détecteurs. Chaque système de détection est un ensemble de composants électroniques qui fonctionnent avec la même fréquence d'horloge. Les données envoyées par ces systèmes et qui sont acceptées par le « L0-trigger » sont reçues par les cartes TELL1.

Les cartes TELL1 utilisées fonctionnent avec la même fréquence d'horloge que le système de détection. A fin de parvenir à réaliser une telle synchronisation, le LHCb a créé des cartes qui supervisent chaque système. Ces cartes jouent le rôle du chef d'orchestre, leur objectif principal est d'envoyer le même signal d'horloge à tous les éléments faisant partie du système en question, elles portent le nom « **ODIN** » [ODIN]. ODIN fournit le signal d'horloge via une fibre optique aux registres « TTCrx » des cartes électroniques.

Si une carte TELL1 rencontre un problème, elle envoie un signal de freinage « throttle » à une carte nommée « **HUGIN** » qui bloque les autres cartes pour assurer la synchronisation des événements. Dès que le signal est remis à zéro le fonctionnement reprend de nouveau.

Une carte HUGIN peut superviser le fonctionnement de 20 cartes TELL1. Elle effectue un « OU » logique de tous les signaux de freinage reçues par les TELL1 et donne en sortie le résultat du OU qui sera envoyé à la carte ODIN gérant le système.

*Figure 3. 17: Structure du système de synchronisation du LHCb*

Les cartes TELL1 possèdent des cartes CCPC que j'ai décrites précédemment, ces cartes publient des services via Dim pour permettre de lire ou d'écrire des données dans les registres des cartes TELL1.

Toutes les cartes TELL1 possèdent un registre nommé TTCrx, où on peut lire le signal d'horloge envoyé par ODIN. L'accès à ce registre peut se faire grâce au bus I<sup>2</sup>C, cet accès est offert par l'un des services du serveur de la carte CCPC embarquée sur la TELL1.

Le signal de freinage « throttle » peut être envoyé grâce à un registre interne à la carte TELL1 en mettant l'un de ses bits à 1, l'écriture se fait via le bus local LBUS du CCPC.

Enfin les cartes TELL1, envoie des paquets de données dur quatre ports de sortie au système d'acquisition selon le protocole TCP/IP.

### Travail demandé

Mon projet consiste à créer une interface graphique avec PVSS pour permettre au personnel de l'installation des cartes TELL1 dans la partie de traitement des données envoyées par les détecteurs de vérifier que :

- les 20 cartes TELL1 connectées à un Hugin sont branchées correctement de point de vue Hardware.
- les serveurs des cartes CCPC répondent sur le même DNS et publient correctement leurs services.
- les cartes reçoivent toutes le même signal d'horloge, si ce n'est pas le cas il faut déterminer la nature du problème.
- les cartes peuvent envoyer un signal de freinage en cas de problème.
- Hugin prend en compte le signal de freinage, et réagit bien à ce signal.
- En cas de résolution de problème, les cartes TELL1 remettent à zéro le signal de freinage.
- Hugin prend en compte ce signal et le fonctionnement reprend normalement.
- Enfin, les cartes doivent être testés au niveau de l'envoi et de la réception des données sur leurs 4 ports Gigabit Ethernet.

## **3.1 Présentation du cadre technique du projet et étude de l'existant**

### **3.1.1 Le système TFC et le framework lbTFC**

Le TFC «Timing and Fast Control» est le système responsable de la synchronisation de tous les éléments intervenant dans l'électronique du LHCb [TFC]. Il distribue l'horloge du LHC de fréquence fixe égale à 40.8 MHz et la décision des filtres de niveaux 0 et 1 (*L0 et L1 trigger*), il assure la synchronisation des reset et le contrôle de l'électronique dite Front-End.

ODIN est connecté via un switch **THOR** au système qu'il supervise. L'ensemble de ce système reçoit l'horloge par une fibre optique reliée au fournisseur d'horloge finale (TTCoc). Le TTCoc fournit le signal en fonction du résultat d'un multiplexeur (TTCvx) qui prend en entrée la décision de ODIN par l'intermédiaire d'un convertisseur électrique-optique (TTCtx).

Le signal d'horloge peut être lu dans les registres internes des cartes TELL1, précisément le registre TTCrx contient des informations relatives à la synchronisation de la carte.



Le schéma suivant simplifie le fonctionnement du système TFC et la distribution d'horloge :

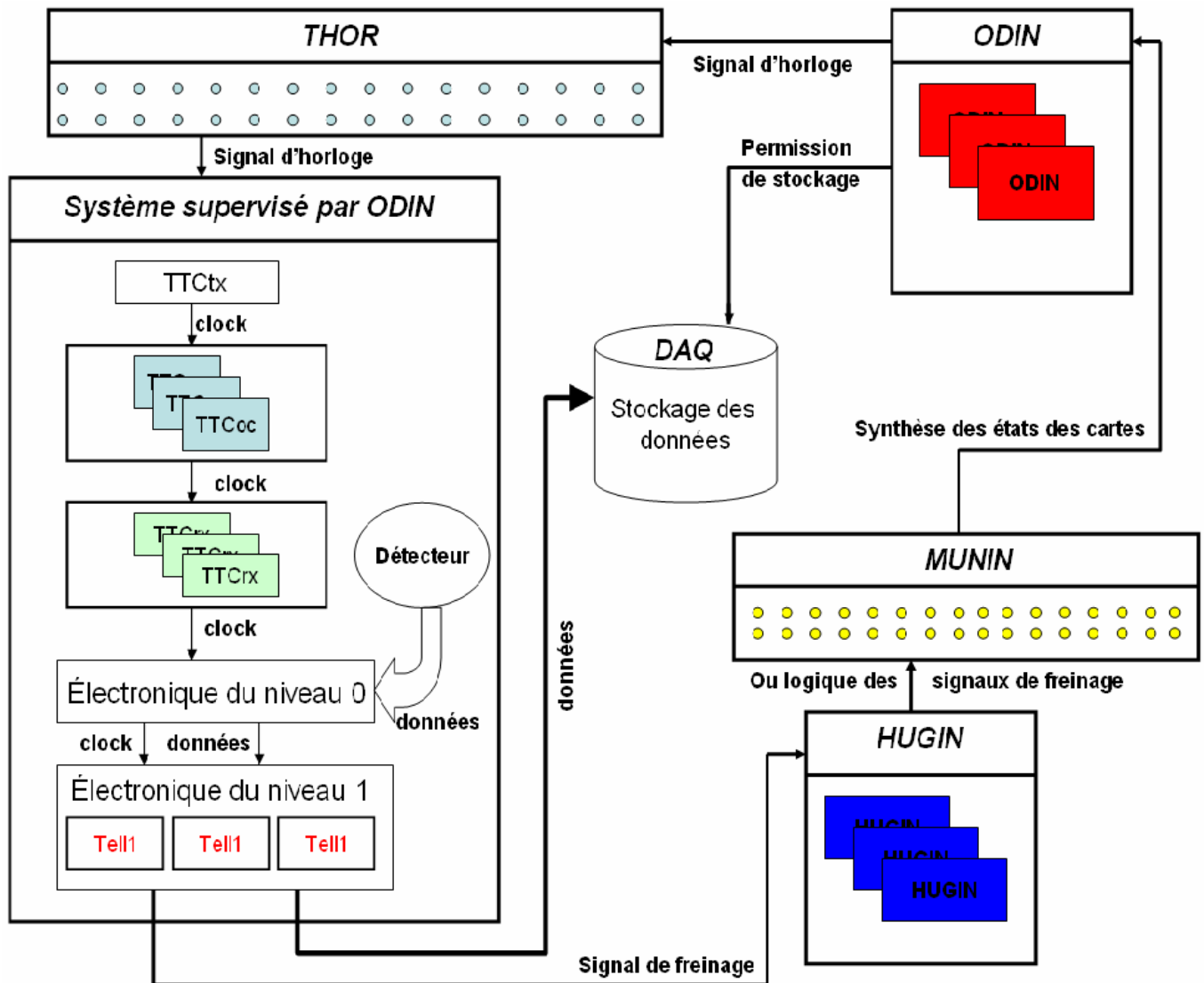


Figure 3. 18: Fonctionnement du système TFC

Une partie du travail demandé est reliée au système TFC, il s'agit de la vérification du signal d'horloge dans les cartes TELL1 et de l'envoi du signal throttle par les cartes vers le système TFC.

Nous avons donc besoin d'un moyen pour vérifier l'arrivée de ces signaux au niveau du TFC. L'outil logiciel existe déjà sous le nom de **lbTFC**.

Le composant framework lbTFC, permet d'effectuer un diagnostic de l'état de ODIN, il suffit de sélectionner le ODIN en question pour obtenir sa connectivité à travers le switch THOR avec l'ensemble des cartes étudiées. Dès lors nous pouvons visualiser l'état du ODIN. S'il est en mode de fonctionnement sans erreurs le test peut être lancé.

Ce framework nous permet de configurer HUGIN, la carte qui reçoit les signaux de freinage de la part des cartes TELL1. Les 20 entrées de cette carte peuvent être activées ou désactivées. A savoir qu'une entrée activée non branchée met automatiquement le signal throttle à 1 et par conséquent bloque entièrement le fonctionnement du système.

Une librairie de fonctions est intégrée dans lbTFC, elle permet de lire et d'écrire dans les registres des cartes HUGIN, nous utilisons certaines fonctions pour la procédure de test des cartes TELL1.

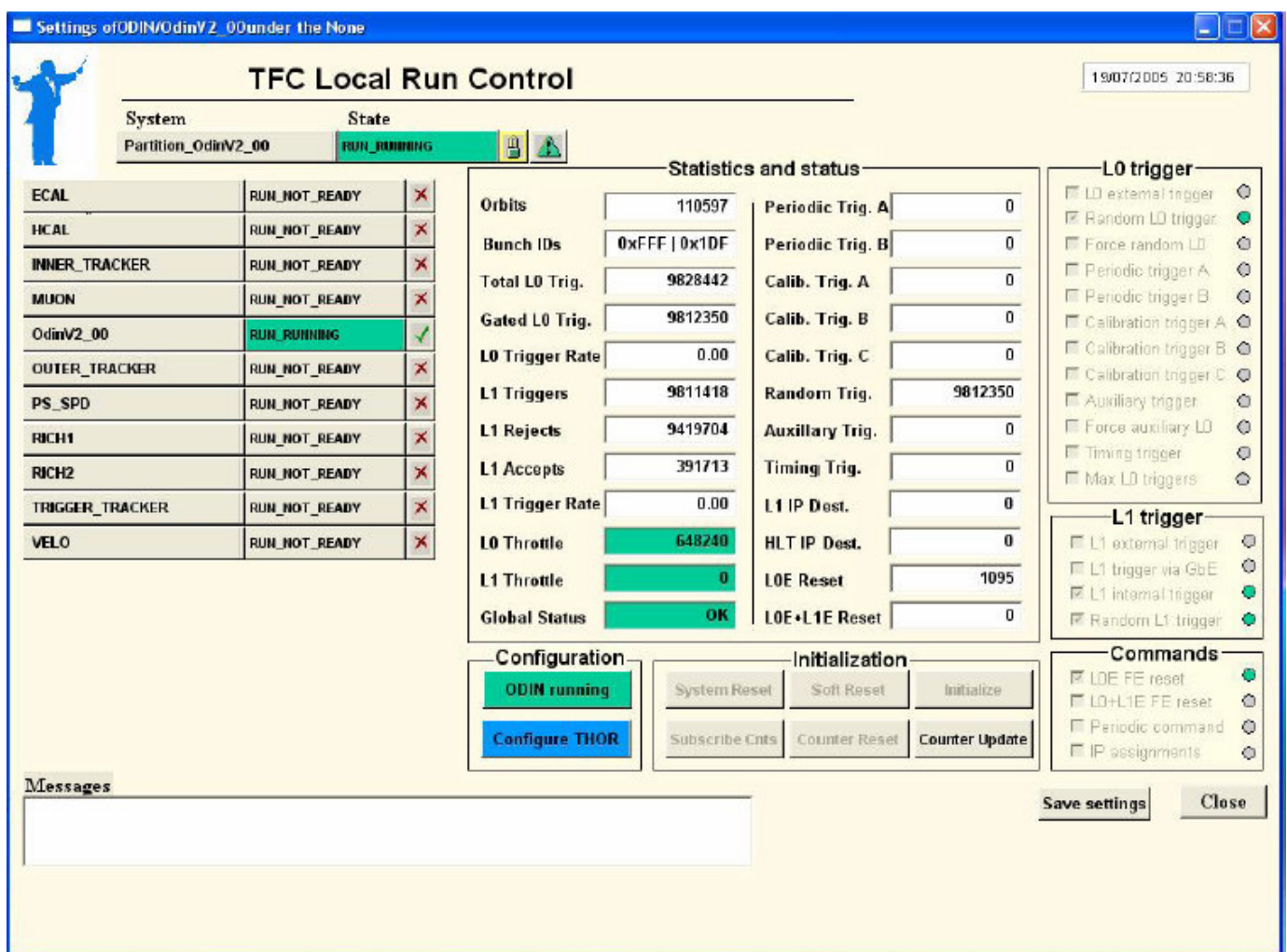


Figure 3. 19: Interface graphique du système TFC

Le serveur TFC est central, tous les projets doivent être distribués dans le sens de PVSS de manière à ce qu'ils puissent communiquer avec les autres projets distribués. En particulier le projet portant le nom TFC est le seul à fournir les données sur toutes les cartes de type : ODIN et HUGIN. Il détermine aussi le chemin à travers les switches pour connecter les systèmes maîtres aux systèmes esclaves.

### 3.1.2 Le framework fwCcpc

Ce composant permet d'avoir accès aux services de la carte CCPC grâce à DIM [fwCCPC].

La carte TELL1 possède une petite carte CCPC qui facilite la lecture et l'écriture des données sur certaines zones de la mémoire de la TELL1. Ces accès sont gérés par les bus : I2C, LBUS et JTAG. Chacun de ces bus est responsable d'un type de données sur la carte. Par exemple nous pouvons lire le signal d'horloge sur le bus I2C et le signal throttle sur le bus LBUS.

L'interface graphique proposée par ce framework permet aussi de lire le contenu de la mémoire de la CCPC entre deux adresses données, d'écrire, de créer des registres et d'exécuter des scripts.

La librairie offerte par le fwCcpc est riche de fonctions dites bas-niveau, elle permet de superviser les composants de la carte d'une manière logicielle.

Nous montrons sur la figure 3.20 l'interface graphique du fwCcpc, sur cet exemple nous pouvons lire sur le LBUS à l'adresse 0x112018 le mot 0x811C80B2 qui s'écrit en binaire sur 32 bits correspondant au type choisi.

*Figure 3. 20: Interface graphique de la carte CCPC*

Le serveur de la carte CCPC *ccserv* tourne sur la carte TELL1. Pour avoir accès à ses services il faut spécifier sur la ligne de commande la variable d'environnement DIM\_DNS\_NODE qui doit être la même que le Dns de la machine sur laquelle tourne le projet de test.

### **3.2 Planification et réalisation des tests**

#### **3.2.1 Conception des étapes de la procédure de test automatisé**

L'objectif du cahier des charges du projet consiste à développer un outil de test des cartes TELL1 avant de les mettre en service. Ces tests doivent être automatisés et permettre à l'utilisateur de localiser les dysfonctionnements dus à des raisons hardware ou software au niveau de leur environnement.

Pour créer cet outil, j'ai développé un algorithme qui permet d'effectuer des tests individuels sur chacune des cartes, des tests de groupe sur chacun des éléments à vérifier et des séries de tests entiers sur le système.

J'ai aussi développé un outil de test des 20 entrées de la carte HUGIN sur lesquelles sont branchées les cartes TELL1 en intégrant les codes du framework TFC. Cet outil permet à l'utilisateur de configurer les entrées de la carte Hugin grâce à l'interface graphique que j'ai développée.

Le traitement des tests se fait suivant les étapes suivantes:

*Figure 3. 21: Description de la procédure de test des TELL1*

J'ai réalisé cinq types de tests différents mais complémentaires. Au cas où un test qualifié de général détecte une erreur, un test sur la carte à problème et un autres sur le détail de disfonctionnement en question s'avèrent nécessaires. En fin la procédure de test finale donne une vue de dessus du système pour valider toutes les étapes de vérification.

### 3.2.2 Développement d'une librairie de fonctions

Afin d'effectuer l'ensemble de ces tests, j'ai créé une librairie de 27 fonctions pour lire et écrire les données dans les registres des cartes CCPC embarquées sur les cartes TELL1 ainsi que dans les registres de Hugin.

J'ai fait appel à des scripts intégrés dans le composant logiciel TFC et j'ai utilisé des fonctions définies dans la librairie du composant fwCcpc. Cette librairie dispose de certaines fonctions nécessaires pour la communication avec le serveur lancé sous TELL1. Elle propose aussi des fonctions pour convertir les données en hexadécimal, binaire au décimal.

J'ai développé dans ce même contexte des fonctions à usage interne qui permettent d'effectuer des opérations logiques binaires. Ces fonctions servent à écrire dans un bit spécifique sans modifier le reste du contenu du registre. Nous pouvons distinguer 5 types de fonctions :

#### a) Fonctions du CCPC

Ces fonctions accèdent aux registres internes de carte CCPC et renvoie des informations sur le serveur *ccserv*. Nous pouvons ainsi afficher le nombre de serveurs en fonctionnement instantanément dans le but de détecter une déconnexion dès sa production. Au cas où la connexion s'interrompt avec tous les serveurs un message d'erreur affiche la cause.

Nous pouvons aussi lire la liste des serveurs en fonctionnement et la liste de serveurs déconnectés.

#### b) Fonctions d'horloge

Ces fonctions ont accès au registre TTCrx qui contient le signal d'horloge envoyé par le TFC. Ce signal ne peut être reçu qu'au cas où ODIN l'autorise autrement dit si aucune des cartes n'a envoyé un signal de freinage à ODIN.

La lecture de ce registre permet d'une part de déterminer au cas de refus à son accès la nature du problème : dû à PVSS ou au serveur CCPC, et d'autre part la lecture de la donnée dans le registre de l'horloge. Cette donnée, une fois comparée à celle du TFC permet d'affirmer si le signal est bien reçu.

Le registre TTCrx circule ses données sur le bus I<sup>2</sup>C, il suffit de donnée à la fonction de lecture 0x58 comme adresse et 0x22 comme sous adresse.

#### c) Fonctions du throttle : TELL1

A fin de communiquer avec la carte HUGIN, deux fonctions permettent de provoquer l'envoi d'un signal de freinage et sa remise à zéro. L'envoi de ces signaux se fait grâce à l'écriture dans l'adresse 0x1000018 de la mémoire de la carte CCPC, cette données doit s'écrire sur le bus local LBUS.

#### d) Fonctions du throttle : HUGIN

Ces fonctions permettent de lire les données sur la carte HUGIN, à chaque appel d'une fonction d'écriture sur le bit correspondant au throttle correspond une fonction de vérification de la réception de cette donnée. Nous utilisons des script associé aux framework lbTFC pour analyser et configurer les états des entrées sur lesquelles sont branchées les cartes TELL1.

#### ***e) Fonctions de test des ports de Ethernet des TELL1***

Les cartes TELL1 font partie de l'électronique du Front-End, elles participent au traitement des données envoyées par les détecteurs et les acheminent au système de stockage DAQ. Des paquets de données circulent sur les ports des TELL1, leur test fait partie de la procédure de test que j'ai développée.

Un serveur DIM tourne sur une machine distante permet de répondre à cette question. En lui fournissant l'adresse IP de la carte ainsi que les données d'une section de la mémoire de la carte, il effectue le test et retourne le résultat sous forme de service DIM.

J'ai créé des fonctions de définition et de lecture de ces sections. Les accès se font sur le bus LBUS, respectivement aux adresses 0x112004 et 0x112018. La section IP doit être configurée avec un identifiant de la carte testée parmi 20. Pour distinguer les cartes j'ai donné à chaque carte sa position par rapport à HUGIN comme identifiant.

Ces données sont ensuite envoyées sous forme de commandes au serveur qui effectue leur test.

### **3.3 Développement de l'interface graphique**

Une interface graphique sous PVSS est un ensemble de boutons, tableaux et zones de texte. Derrière ces objets se trouvent des scripts qui s'exécutent en fonction de l'événement se produisant sur l'objet.

J'ai utilisé l'ensemble de ces objets pour créer des fenêtres de configuration et de visualisation des tests. Les tests étant de trois types : positifs, négatifs ou inactifs. J'ai utilisé trois couleurs pour représenter les résultats des deux séries de tests sur les entrées de HUGIN d'une part et les cartes TELL1 d'autre part.

Quant aux trois autres types de tests individuels j'ai utilisé des boutons dont les codes lancent une procédure de test portant sur la partie concernée. Les résultats sont affichés dans une table et des zones de textes.

Deux types de messages s'affichent, des messages d'erreurs au cas où l'utilisateur ne respecte pas les consignes des tests et des messages d'avertissement si la connexion se coupe avec le système supervisé à cause du Dns.

#### **3.3.1 Initialisation et configuration des cartes TELL1**

Cette étape consiste à lancer un premier *Panel* pour imposer à l'utilisateur une convention d'identification des cartes selon un ordre qui doit être respecté lors de leur branchement sur HUGIN.

Un problème se pose à cause de l'absence d'un signal d'adressage entre les cartes TELL1 et la carte HUGIN. Le technicien responsable du branchement est le seul à savoir l'ordre dans lequel elles sont installées.

Le seul moyen pour reconnaître les cartes TELL1 est leurs noms, ces noms ne respectent pas un ordre relatif à leur branchement. Il est donc nécessaire d'imposer un ordre à l'utilisateur qui doit fournir en entrée les noms des cartes TELL1. Ces noms seront utilisés pour appeler les fonctions de lecture des registres internes.

Au niveau de Hugin ce sont les numéros d'entrées qui sont utilisés pour appeler les fonctions de la librairie de HUGIN.

Pour ces raisons nous imposant que les cartes soient reliées à HUGIN dans le même ordre que leur noms dans le fichier d'initialisation.

Avant de lancer les tests une interface graphique lit les données du fichier texte dans lequel sont stockés les noms des cartes et l'utilisateur confirme l'ordre. Une fois validée, la structure est copiée dans la base de données du projet. Les tests prendront ensuite cette structure comme bijection entre les cartes et les entrées de HUGIN.

Comme le montre la figure suivante, le lancement de cette interface permet de télécharger le fichier des 20 cartes et d'établir une correspondance entre les cartes et les entrées de HUGIN. Par exemple la carte *pctell39* correspond à l'entrée 00 de HUGIN et l'utilisateur doit la brancher ainsi.

Nous remarquons sur cet exemple la répétition de certain nom de serveurs Tell1. Ceci est dû au nombre limité des cartes dont je disposais pour effectuer les tests, en pratique les noms sont strictement différents. Le cas échéant un test permet d'afficher les répétitions.

Le bouton *apply* permet de stocker les données dans la base de données du projet d'une part et de lancer l'interface graphique de test et de configuration.

Figure 3. 22: Ordonnancement des TELL1



### 3.3.2 Configuration de HUGIN

Après avoir lancé l'application de test, il faut configurer les entrées de la carte HUGIN grâce à une nouvelle interface graphique intégrée dans l'application de test. Il faut noter qu'à ce stade les données sont échangées avec le projet central TFC. Le projet que j'ai créé est distribué et peut communiquer avec d'autres projets.

Toutes les cartes de supervision du Cern sont contrôlées uniquement par le projet TFC et tout accès à leur configuration passe strictement par ce projet. Je dois donc gérer deux systèmes, le premier est en relations directe avec HUGIN et l'autre avec les cartes TELL1.

La fenêtre de configuration des entrées de HUGIN exécute des codes qui utilisent des fonctions de la librairie *fwTFC* qui permettent d'autoriser ou de bloquer les 20 entrées.

Sur la figure suivante la carte que j'utilise s'appelle *HUGIN/HuginV1\_0*, ce nom permet d'identifier la carte car les noms sont affectés selon une convention d'appellation créée par les concepteurs des cartes.

Dans cet exemple les entrées cochées sont toutes autorisées. Uniquement 2 cartes Tell1 sont branchées sur les entrées 0 et 1 du HUGIN. Et comme les autres sont autorisées et non branchées le signal throttle se met par défaut à un 1. D'où leurs couleur rouge représentant le freinage. La sortie est le résultat du OU logique de toutes les entrées.

Les entrées 0 et 1 sont branchées et n'envoient aucun throttle leur couleur est alors verte. Les autres sont grises car les entrées ne sont pas autorisées. Une fois la configuration est validée le test peut être lancé.

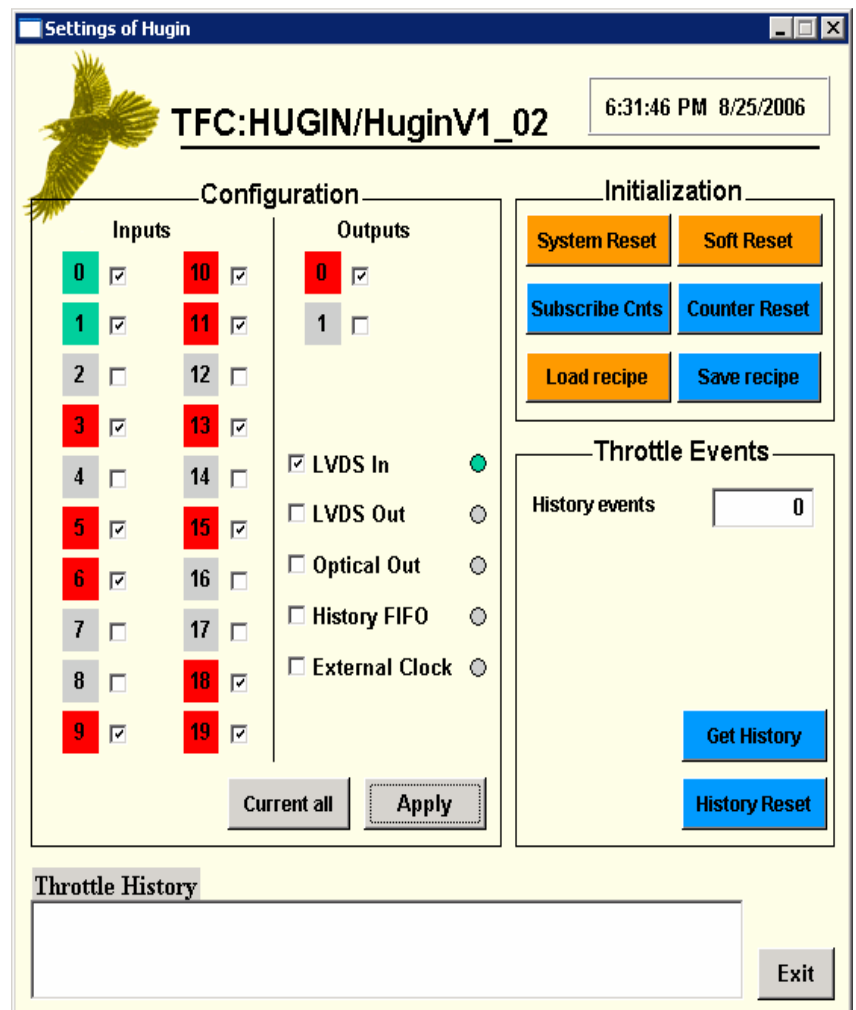


Figure 3. 23: Configuration des entrées de HUGIN

## 4. Résultats et discussion

Mon travail porte sur deux projets différents qui s'inscrivent dans le cadre de l'expérience LHCb. Le premier fait partie des outils de détection dans la partie de l'accélérateur où se situe l'aimant de séparation des particules chargées, il permet de visualiser et d'enregistrer certaines grandeurs physiques. Le deuxième fait partie du système de traitement des données en provenance des détecteurs, c'est un outil de vérification du fonctionnement des cartes utilisées.

### 4.1 Résultats de l'outil de calibration et de visualisation du champ magnétique

L'outil de traitement des valeurs du champ magnétique envoyées par le composant ELMB a été testé dans les conditions du champ magnétique terrestre. En pratique les valeurs qui devront être mesurées sont 30000 fois plus grandes que celles que j'ai testées.

#### 4.1.1 Test du serveur de calibration

Le serveur de calibration reçoit les commandes envoyées par PVSS, appelle la fonction de calibration et publie les valeurs calibrées si les données ne sont pas erronées sinon il retourne un message d'erreur que PVSS prend en compte pour les traitements ultérieurs.

Sur la figure suivante nous avons la trace du fonctionnement du serveur.

```

Terminal
File Edit View Terminal Go Help
in Bz: 2201 =====> out Bz: 4.60048e-05
stringInfo: d10000a17ded001/commandDp1/connected

*****INFORMATION*****INFORMATION*****INFORMATION*****
info : 3
connected
temperature : 29.99
data point name : commandDp1
identifier used for calibration: 0xd1 0x00 0x00 0x0a 0x17 0xde 0xd0 0x01
-----
in Bx: 1.67739e+07 =====> out Bx: -5.45332e-05
in By: 7881 =====> out By: 4.34888e-05
in Bz: 2201 =====> out Bz: 4.59997e-05
stringInfo: d10000a17ded001/commandDp1/connected

*****INFORMATION*****INFORMATION*****INFORMATION*****
info : 3
connected
temperature : 29.949
data point name : commandDp1
identifier used for calibration: 0xd1 0x00 0x00 0x0a 0x17 0xde 0xd0 0x01
-----
in Bx: 1.67738e+07 =====> out Bx: -5.48225e-05
in By: 7881 =====> out By: 6.20274e-05
in Bz: 2305 =====> out Bz: 2.99257e-05

```

Figure 3. 24: Trace du traitement du serveur de calibration

### 4.1.2 Test de l'interface graphique

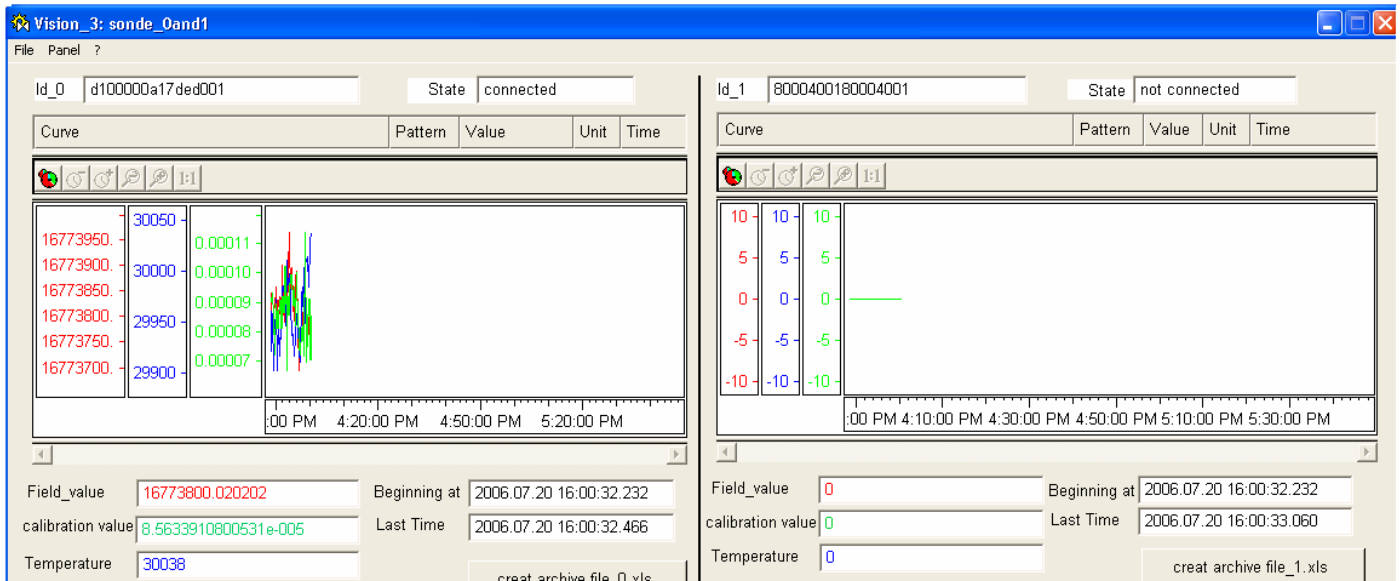


Figure 3. 25: Fenêtre de visualisation des valeurs calibrées

Comme le test s'effectue sur un composant ELMB qui porte quatre capteurs, j'ai développé une interface de quatre parties symétrique. Chacune des parties de l'interface concerne un capteur et donne les informations suivantes :

- L'identifiant de la sonde qui a effectué l'acquisition ainsi que son état.  
L'identifiant d'une sonde qui n'est pas opérationnelle est par défaut égal à 8000400180004001. L'état de la sonde s'affiche en ligne, il change dès que la sonde change d'état.
- La valeur du champ magnétique avant la calibration cette donnée est mise à jour toutes les 20 s.
- La valeur calibrée du champ magnétique envoyée par le serveur DIM distant.
- La valeur de la température à chaque instant.

- L'instant de début du test et celui d'enregistrement de la dernière valeur reçue.
- Trois courbes relatives aux valeurs calibrées et non calibrées ainsi que la valeur de la température correspondant à la mesure.

#### 4.1.3 Test de la procédure de sauvegarde

J'ai créé un *DataPoint* où l'utilisateur peut fixer lui-même les instants de début et de fin d'enregistrement des valeurs envoyées par le serveur. Les résultats sont ensuite stockés dans un fichier. Grâce à ce *DataPoint* on peut avoir accès aux données enregistrées entre deux instants très proche. Ce fichier peut être considéré comme une loupe dont la précision dépend de la durée de sauvegarde. On peut lire les données depuis la création du projet avec une fréquence de 5 s.

Chacun des quatre capteurs enregistre son historique dans un fichier suite à la demande de l'utilisateur.

	A	B	C	D	E	F	G	H	I	J	K	L
196	20/7/2006	15:43:13	0	-0.000059	0.000046	0.000027	29.865999					
197	20/7/2006	15:43:18	0	-0.000059	0.000046	0.000027	29.865999					
198	20/7/2006	15:43:23	0	-0.000059	0.000046	0.000027	29.865999					
199	20/7/2006	15:43:28	0	-0.000059	0.000046	0.000027	29.865999					
200	20/7/2006	15:43:33	0	-0.000061	0.000062	0.000050	29.857000					
201	20/7/2006	15:43:38	0	-0.000061	0.000062	0.000050	29.857000					
202	20/7/2006	15:43:43	0	-0.000061	0.000062	0.000050	29.857000					
203	20/7/2006	15:43:48	0	-0.000061	0.000062	0.000050	29.857000					
204	20/7/2006	15:43:53	0	-0.000034	0.000032	0.000018	29.882999					
205	20/7/2006	15:43:58	0	-0.000034	0.000032	0.000018	29.882999					
206	20/7/2006	15:44:3	0	-0.000034	0.000032	0.000018	29.882999					
207	20/7/2006	15:44:8	0	-0.000034	0.000032	0.000018	29.882999					
208	20/7/2006	15:44:13	0	-0.000052	0.000063	0.000031	29.871000					
209	20/7/2006	15:44:18	0	-0.000052	0.000063	0.000031	29.871000					
210	20/7/2006	15:44:23	0	-0.000052	0.000063	0.000031	29.871000					
211	20/7/2006	15:44:28	0	-0.000052	0.000063	0.000031	29.871000					
212	20/7/2006	15:44:33	0	-0.000059	0.000048	0.000023	29.899000					
213	20/7/2006	15:44:38	0	-0.000059	0.000048	0.000023	29.899000					
214	20/7/2006	15:44:43	0	-0.000059	0.000048	0.000023	29.899000					
215	20/7/2006	15:44:48	0	-0.000059	0.000048	0.000023	29.899000					
216	20/7/2006	15:44:53	0	-0.000044	0.000067	0.000031	29.916000					
217	20/7/2006	15:44:58	0	-0.000044	0.000067	0.000031	29.916000					
218	20/7/2006	15:45:3	0	-0.000044	0.000067	0.000031	29.916000					
219	20/7/2006	15:45:8	0	-0.000044	0.000067	0.000031	29.916000					
220	20/7/2006	15:45:13	0	-0.000084	0.000055	0.000040	29.879000					
221	20/7/2006	15:45:18	0	-0.000084	0.000055	0.000040	29.879000					
222	20/7/2006	15:45:23	0	-0.000084	0.000055	0.000040	29.879000					
223	20/7/2006	15:45:28	0	-0.000084	0.000055	0.000040	29.879000					
224	20/7/2006	15:45:33	0	-0.000045	0.000072	0.000028	29.886999					
225	20/7/2006	15:45:39	0	-0.000045	0.000072	0.000028	29.886999					
226	20/7/2006	15:45:43	0	-0.000045	0.000072	0.000028	29.886999					
227	20/7/2006	15:45:48	0	-0.000045	0.000072	0.000028	29.886999					
228	20/7/2006	15:45:54	0	-0.000051	0.000052	0.000042	29.934000					

Figure 3. 26: Fichier de sauvegarde des données calibrées

#### 4.1.4 Création du framework

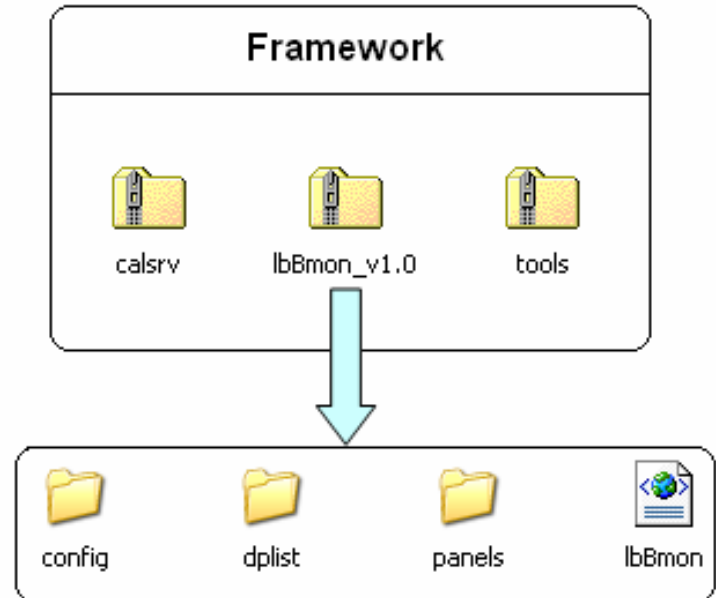
Afin de mettre en service le composant que j'ai créé, je l'ai mis sous forme d'un framework téléchargeable sur le serveur du CERN.

Le framework du projet *lbBmon* contient le serveur de calibration avec les librairies nécessaires et la fonction de calibration, le composant *lbBmon\_v1.0* à installer sous PVSS et les outils qui permettent d'utiliser le projet.

Le composant que j'ai créé a été testé sous une autre machine. Après l'avoir téléchargé dans un nouveau projet nous avons effectué des tests et les résultats étaient identiques aux résultats sous le projet original.

Le contenu de *lbBmon* comporte :

- un fichier portant le même nom que le composant décrit sa structure.
- L'ensemble des DataPoints du projet et des classes d'archivage est traduit sous forme de fichier contenu dans *dplist*.
- Le fichier de configuration du serveur OPC et celui du projet sont copiés dans le framework.
- L'interface graphique est contenue dans *panel*, elle peut être lancée dès l'installation du composant. Elle est dès lors opérationnelle.



#### 4.2 Résultats de la procédure de test des cartes TELL1

L'outil de vérification des cartes TELL1 que j'ai créé peut être qualifié de complexe, il fait appel à plusieurs scripts et utilise 7 composants PVSS développés au CERN. 3 types de serveurs DIM fournissent en permanence des données au projet : le serveur TFC, 20 serveurs CCPC et le serveur de test des ports des cartes TELL1.

La vérification du fonctionnement de ce projet consiste à vérifier : la communication avec les serveurs des cartes TELL1, la carte HUGIN, le système distribué TFC et le serveur DIM de test d'échange de données sur les ports des TELL1.

##### 4.2.1 Communication avec les cartes TELL1

J'ai effectué le test de lecture et d'écriture des registres des cartes TELL1, en utilisant les bus I<sup>2</sup>C et LBUS. Sur la figure suivante nous lisons les opérations qu'effectue le projet PVSS sur la carte tell034. Le registre d'horloge TTCrx fournit la donnée du signal reçu par le système de synchronisation TFC.

Sur le bus LBUS, nous avons accès à l'adresse 0x1000018 au registre throttle sur lequel nous pouvons envoyer le signal à HUGIN ou le mettre à zéro.

*Figure 3. 27: Communication avec la carte TELL1*

#### 4.2.2 Interface graphique

La procédure de test des cartes a été lancée avec 2 cartes TELL1 branchée sur les entrées 0 et 1 de HUGIN, les autres entrées ne sont pas branchées.

A partir de l'interface graphique que j'ai créé la fenêtre de configuration de HUGIN peut être lancée. Une fois configurées, les entrées de HUGIN qui ne sont pas activées ne sont pas prises en compte par ma procédure de test, les afficheurs sont donc grisés. Les autres entrées sont testées.

Quant aux cartes TELL1, le test s'effectue uniquement sur celles dont le serveur CCPC répond sur le même Dns du projet. Je procède alors aux tests suivants :

##### **a. Test « All servers »**

Nous avons deux serveurs CCPC accessibles par le projet. 8 boutons permettent d'exécuter des tests sur : les serveurs connectés, les serveurs non connectés, la vérification du fichier contenant tous les serveurs, la lecture des données du registre TTCrx, la détermination de l'origine des erreurs d'horloge, l'envoi et l'arrêt du throttle, la lecture de l'états du throttle dans HUGIN.



**b. Test « TELL1 individual test »**

Lors du lancement du projet la liste des cartes TELL1 à tester est téléchargée dans une liste de sélection disponible. Pour tester une carte spécifique, il suffit de sélectionner la carte à tester pour qu'un script s'exécute et fournit l'état du CCPC, de l'horloge et la donnée du registre TTCrx.

**c. Test « Control Hugin input»**

Avant de lancer les tests l'utilisateur doit fournir au projet le même nom de la carte HUGIN utilisée que celui qui a été utilisé lors de sa configuration. Pour éviter qu'une erreur se produise cette donnée est copiée grâce à l'activation du bouton *set hugin name*. La liste des entrées de HUGIN permet d'effectuer un test sur chacune d'elles et de visualiser son état actuel.

**d. Test « Inputs Diagnostic»**

Ce test dure plus d'une minute il effectue toute une série de tests sur chacune des entrées de HUGIN en impliquant les cartes TELL1 dans cette procédure. Le résultat du test que nous avons sur la figure suivante montre que les deux premières entrées fonctionnent correctement. Les entrées grisées sont désactivées donc aucun test n'a été effectué et les entrées rouges sont activées mais aucun serveur n'est détecté ou le serveur n'arrive pas à communiquer avec HUGIN, le résultat est donc négatif.

**e. Test « Global test»**

Ce test représente la procédure de test automatisée. C'est l'exécution automatique de tous les tests précédents. Le bilan final de ce test est le résultat effectif à prendre en compte pour confirmer que l'installation est validée et que les cartes peuvent être intégrées dans l'électronique du Front-End.

L'exécution de ce test dure 3 à 5 minutes et tout test lancé en parallèle avec le «Global test » peut falsifier les résultats de celui-ci.

Dans cette procédure nous testons aussi les ports des cartes TELL1.

*Figure 3. 28: Interface graphique de test des cartes TELL1*



### 4.2.3 Enregistrement des résultats finaux

Les résultats du test global effectué sur les 20 cartes TELL1, sont d'une part affichés sur l'interface graphique et d'autre part leur trace est sauvegardée dans un fichier. Nous retrouvons pour chaque carte l'ensemble de ses résultats. Voici un extrait de ce fichier :

*Figure 3. 29: Trace des résultats de la procédure de test*

## 4.3 Discussion

### 4.3.1 Avancement du projet

Durant mon stage j'ai travaillé sur deux sujets différents, chacun intervient dans une partie de l'expérience LHCb. PVSS étant un outil logiciel commun aux deux sujets, je l'ai étudié durant la première période de mon stage et en fonction de l'avancement de mon travail de nouveaux besoins sont apparus dont l'apprentissage de DIM.

Comme j'ai utilisé deux composants matériels, ELMB pour la première partie et TELL1 pour la deuxième, l'étude des documents descriptifs de ces composants a constitué une étape considérable du déroulement de mon stage.

Le développement de l'outil de calibration du champ magnétique fait appel à une fonction qu'il fallait intégrer dans le serveur qui reçoit les commandes de PVSS. Cette fonction a besoin de certains paramètres ne correspondant pas aux formats des données fournies par PVSS. La conversion des données fait partie des principales étapes du traitement des valeurs brutes du champ magnétique.

Les cartes TELL1 se situent au cœur de l'expérience LHCb et fonctionnent mutuellement avec les autres composants du système électronique, il était donc important pour moi d'avoir une vue d'ensemble et de comprendre les éléments qui sont en interaction avec ces cartes.

Afin de réaliser la procédure automatisée de test des TELL1, j'ai utilisé 7 projets élaborés sous PVSS qui portent sur le système de synchronisation, la configuration de la base de données, les cartes CCPC, l'échange des données sur les ports des TELL1 et la communication avec les cartes HUGIN. Après avoir maîtrisé les traitements et les particularités de chacun de ces projets j'ai intégré les fonctions qu'ils proposent dans l'ensemble des tests que j'ai développés.

J'ai créé une librairie de fonctions pour rendre mon projet générique et accessible par les personnes qui pourraient reprendre mon travail. Au lieu de définir des scripts exécutables lors de la manipulation des objets graphique, j'ai écrit des codes faisant appel aux fonctions prédéfinies dans la librairie.

Enfin j'ai conçu l'interface graphique d'une manière à rendre son utilisation accessible à toute personne ayant une légère connaissance de PVSS. J'ai utilisé des afficheurs en couleurs et des messages pour communiquer les résultats de chaque test.

### 4.3.2 Enrichissement personnel

Ce stage a été pour moi la première expérience professionnelle en dehors des projets réalisés au sein de mon école. J'ai appris à collaborer avec mon équipe de travail et à chercher l'information, non seulement sur des documents, mais aussi en contactant les personnes concernées.

J'ai appris à gérer les problèmes techniques d'une manière autonome et j'ai réussi à développer une capacité à synthétiser les points communs de différents projets afin de les intégrer dans un travail à part entière.

L'un des plus grands profits que j'ai tiré de cette expérience est la technique de la gestion d'un système complexe en tenant compte de chacun de ses composants, de la synchronisation des événements et des éventuelles erreurs au cas d'une fausse manipulation par l'utilisateur. Au fur et à mesure de sa réalisation j'ai eu des réflexions sur les perspectives et l'avenir de mon travail.

Enfin la période que j'ai vécu CERN était pour moi une opportunité d'intégration dans un environnement universel où plusieurs nationalités sont présentes.

## Conclusion

Durant mon stage j'ai vécu des étapes allant de l'apprentissage à la concrétisation. Cette expérience m'a prouvée l'importance d'analyser tous les détails qui peuvent intervenir dans un projet et de tester avec une grande précision chaque élément avant de l'intégrer dans un système complexe.

J'ai découvert que l'efficacité d'un travail dépend essentiellement de l'uniformité des outils adoptés. Dans ce sens la cohérence et la complémentarité des projets menés par le CERN sont les fruits des outils développés pour son usage interne. Grâce à ces outils j'ai réalisé mon travail en intégrant facilement plusieurs projets existants.

Dans un système aussi complexe que l'expérience LHCb, la supervision et le contrôle à distance est une clé pour la simplification et l'accélération des traitements. Dans cette optique j'ai traduit un processus de test matériel en une procédure automatisée basée sur la communication par signaux avec le Hardware.

Afin de localiser les points de dysfonctionnement j'ai créé des tests supplémentaires qui permettent à l'utilisateur de vérifier indépendamment les différentes parties pouvant être à l'origine de l'erreur. La simplicité de la visualisation des résultats est une contrainte que j'ai prise en compte en basant mon interface graphique sur des afficheurs pouvant représenter tous les cas possibles.

Une éventuelle amélioration de ce projet consiste à tester plusieurs groupes de cartes TELL1 grâce à la duplication des structures que j'ai créées. Pour faciliter la reprise de mon travail j'ai développé une librairie commentée à laquelle pourraient s'ajouter de nouvelles fonctions.

Quant à l'outil de calibration et de visualisation du champ magnétique, son intégration dans un autre projet est assurée grâce au framework. Le serveur Dim que j'ai créé pour la calibration permet des traitements à distance ce qui rend son utilisation flexible.

En fin le travail que j'ai réalisé pendant mon stage sera intégré dans l'expérience LHCb lors de sa mise en marche de l'accélérateur LHC prévue officiellement au cours de l'année prochaine.

# Glossaire

**Accélérateur :** anneau circulaire dans lequel les particules sont soumises à un champ électrique permettant d'augmenter leur énergie et leur vitesse. Elles sont ensuite séparées dans un champ magnétique très intense.

**Antimatière :** terme créé en 1928 par le physicien Paul Dirac, c'est l'image miroir de la matière. La théorie montre que lors de la création de l'univers la matière et l'antimatière formaient un équilibre, elles sont dotées des propriétés exactement opposées.

**Antiparticule:** c'est l'opposé de la particule dans le contexte de la notion de matière-antimatière. Elle a la même masse que la particule correspondante mais ses nombres quantiques sont opposés et en particulier sa charge électrique.

**Big Bang:** il y a environ 33,7 milliards d'années l'univers a vécu une époque dense et chaude, cette phase marque le début de la dilatation et de l'expansion de l'univers. Comparée à une explosion cet événement a été désigné sous le terme de Big Bang.

**Boson de Higgs:** cette particule élémentaire est supposée expliquer l'origine de la masse de toutes les particules de l'univers. Sa découverte serait une confirmation du modèle standard.

**CCPC :** Credit Card Personal Computer est une carte de la taille d'une carte de crédit embarquée dans certaines cartes électroniques afin de faciliter l'accès aux registres de celles-ci. Elles utilisent le protocole DIM développé par le CERN.

**DAQ :** système d'acquisition des données provenant des détecteurs de l'expérience LHCb.

**DNS :** Domain Name System est un système permettant d'établir une correspondance entre une adresse IP et un nom de domaine dans le but de trouver une information à partir du nom de domaine.

**Effet Hall :** concept découvert en 1879 par Edwin Herbert Hall, il consiste à dire qu'un courant électrique traversant un matériau qui baigne dans un champ magnétique engendre une tension perpendiculaire à ceux-ci.

**ELMB :** composant d'acquisition des valeurs du champ magnétique développé par le CERN. Il supporte des radiations très fortes, sa durée de vie est de 10 ans dans un champ magnétique 30000 fois plus grand que le champ magnétique terrestre.

**Fonction de calibration :** fonction créée par les physiciens du CERN, elle permet de corriger les valeurs du champ magnétique mesurées grâce à un nombre considérable de tests effectués sur les capteurs du ELMB.

**Force faible:** est une des quatre forces fondamentales de la nature. Elle est responsable de la désintégration radioactive qui conduit à l'émission d'une particule bêta (électron ou positron).

**Gigabit Ethernet:** terme utilisé pour décrire une variété de technologies utilisées pour implémenter le standard Ethernet à des taux de transfert de données de un gigabit par seconde. Ces technologies reposent sur de la paire torsadée de câbles de cuivre ou sur de la fibre optique.

**JCOP Frame Work:** Joint COntrol Project est une structure logicielle développée au CERN qui contient tous les composants disponibles et qui permet de les intégrer dans d'autres projets.

**L0/L1 trigger:** le niveau 0 et le niveau 1 de filtrage des données en provenance des détecteurs du LHCb. Ces données arrivent avec une fréquence de 4 Téraoctets par seconde dont un très petit pourcentage peut être considéré comme utile, le L0 et le L1 trigger sélectionnent les événements intéressants.

**LHC :** Large Hadron Collider est le plus grand accélérateur des particules au monde sous forme d'un anneau circulaire de 27 Km de diamètre. Situé sur la frontière franco-suisse aux environs de Genève.

**Modèle standard :** c'est une représentation qui s'applique aux objets quantiques et qui tente d'expliquer leurs interactions. Elle distingue des familles de particules par les forces auxquelles elles sont sensibles.

**Particule :** est l'un des composants élémentaires de la matière étudiée par la physique des particules. Nous retrouvons deux types de particules, les particules chargées (électrons, positrons, muons, protons, alpha, pions) et les particules non chargées (photons, neutrons, neutrinos).

**Physique classique :** l'ensemble des théories physiques validées jusqu'à la fin du XIX<sup>ème</sup> siècle à savoir : la mécanique newtonienne, la théorie du champ électromagnétique et la thermodynamique. La dénomination « physique classique » a été introduite par opposition à la physique quantique.

**Protocole TCP/IP:** c'est la pile des protocoles utilisée par Internet. Son nom est la combinaison de deux de ses protocoles TCP (Transmission Control Protocol) et IP (Internet Protocol). Le modèle OSI définit les couche de cette pile, chaque couche résout un certains nombres de problèmes relatifs à la transmission de données et fournit les services bien définis aux couches supérieures.

**Quarks:** ce sont les particules fondamentales qui forment la matière nucléaire. Ils ne peuvent exister qu'en groupe et contiennent les Hadrons. Les Hadrons les plus connus sont le proton et le neutron.

**Radiation :** ou Rayonnement est le processus de transfert d'énergie sans matière. Le transfert se fait par rayonnement électromagnétique. Il peut en effet se réaliser dans le vide. L'ultime exemple de ce transfert est le rayonnement du soleil dans l'espace.

**TFC:** c'est le système central de synchronisation de tous les composants électronique de l'expérience LHCb.

**Violation du CP :** la violation de symétrie CP est l'une des trois conditions nécessaires pour expliquer l'asymétrie matière-antimatière que nous observons dans l'univers. D'après le modèle standard de la physique cette violation s'explique à partir d'un mécanisme basé sur l'existence de trois familles de quarks.

# Références bibliographiques

- [CCPC] LHCb Credit-Card PCs  
*<http://lhcb-online.web.cern.ch/lhcb-online/ecs/ccpc/default.htm>*
- [CERN] *<http://public.web.cern.ch/public/>*
- [DIM] GASPAR Clara  
*<http://dim.web.cern.ch/clara/fw/FwDim.html>*
- [ELMB] COOK James  
*<http://elmb.web.cern.ch/ELMB/ELMBhome.html>*
- [fwCCPC] KOESTNER Stefan, Fernandes Ricardo  
*[http://lhcb-online.web.cern.ch/lhcb-online/ecs/PVSS\\_CCPC/default.html](http://lhcb-online.web.cern.ch/lhcb-online/ecs/PVSS_CCPC/default.html)*
- [JCOP] JCOP group  
*<http://itcobe.web.cern.ch/itcobe/Services/Pvss/>*
- [LHCb] Tatsuya Nakada Violation of Particle Anti-particle Symmetry, CERN Summer Student Lectures 6, 9, 10 and 11 August 2004
- [PVSS] PVSS service  
*<http://itcobe.web.cern.ch/itcobe/Services/Pvss/>*
- [TELL1] HAEFELI Guido, BAY Aurelio, LEGGER Federica, LOCATELLI Lorent, CHRISTIANSEN Jorgen, WIEDNER Dirk  
*[http://lphe.epfl.ch/%7Eghaefeli/specification\\_and\\_documents/TELL1.pdf](http://lphe.epfl.ch/%7Eghaefeli/specification_and_documents/TELL1.pdf)*
- [TFC] JACOBSSON Richard, Beat JOST  
*[http://lhcb-online.web.cern.ch/lhcb-online/TFC/documents/LEB2001\\_paper.pdf](http://lhcb-online.web.cern.ch/lhcb-online/TFC/documents/LEB2001_paper.pdf)*
- [ODIN] JACOBSSON Richard  
*[http://lhcb-online.web.cern.ch/lhcb-online/TFC/documents/TNS-56-2003\\_R1\\_full.pdf](http://lhcb-online.web.cern.ch/lhcb-online/TFC/documents/TNS-56-2003_R1_full.pdf)*



# ***Annexes***

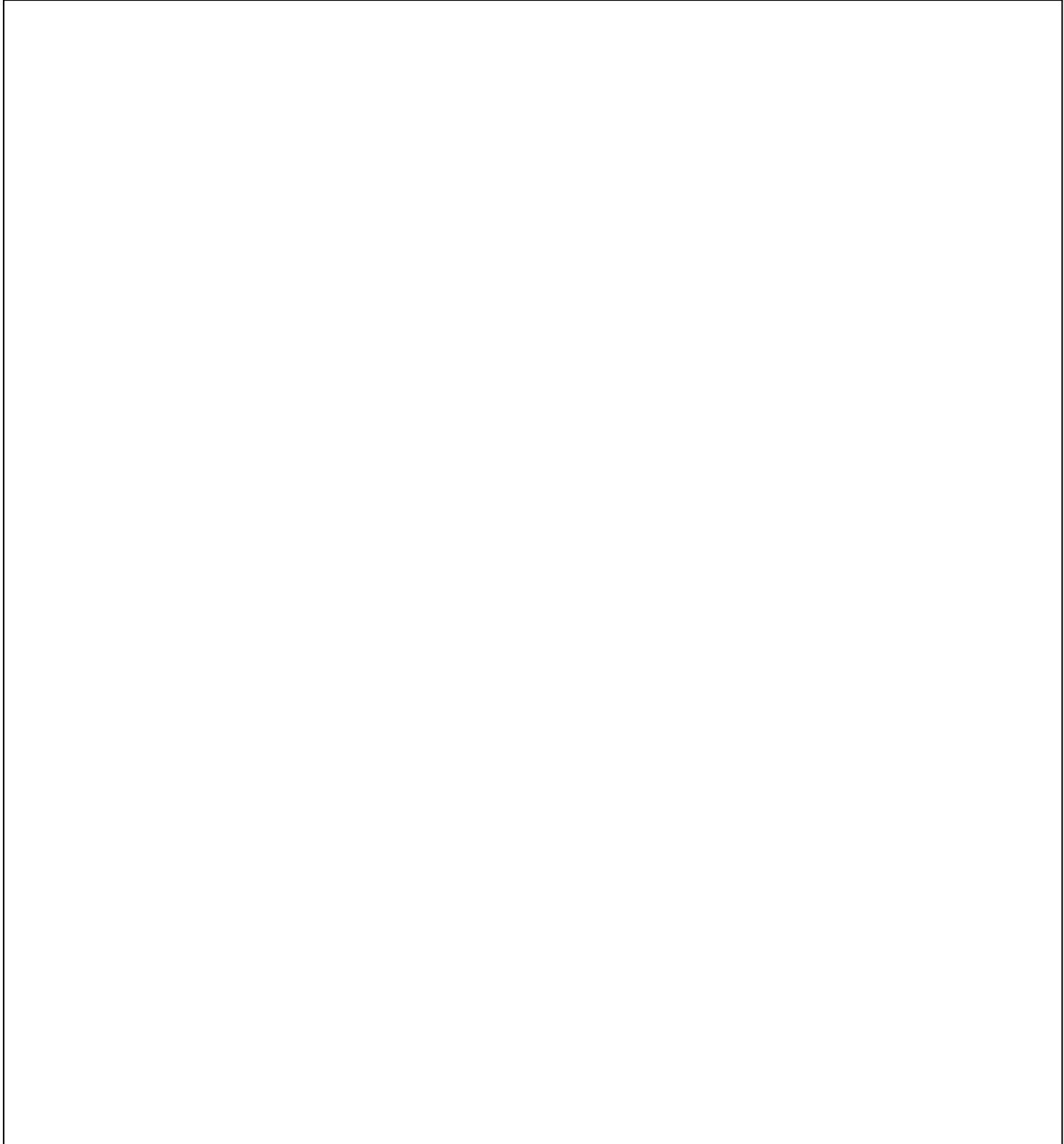
# Annexe A

## A.1 Boucle while du serveur de calibration

*Figure A. 30: Code de la boucle while du serveur de calibration*

## A.2 Description du composant ELMB

L'horloge interne du ELMB fonctionne avec une fréquence de 4 MHz et le processeur embarqué sur ce composant a l'architecture d'un RISC. Voici la fiche technique des caractéristiques principales du ELMB :



*Figure A. 31: Fiche de description du ELMB*

# Annexe B

## B.1 Détails de l'architecture du système TFC

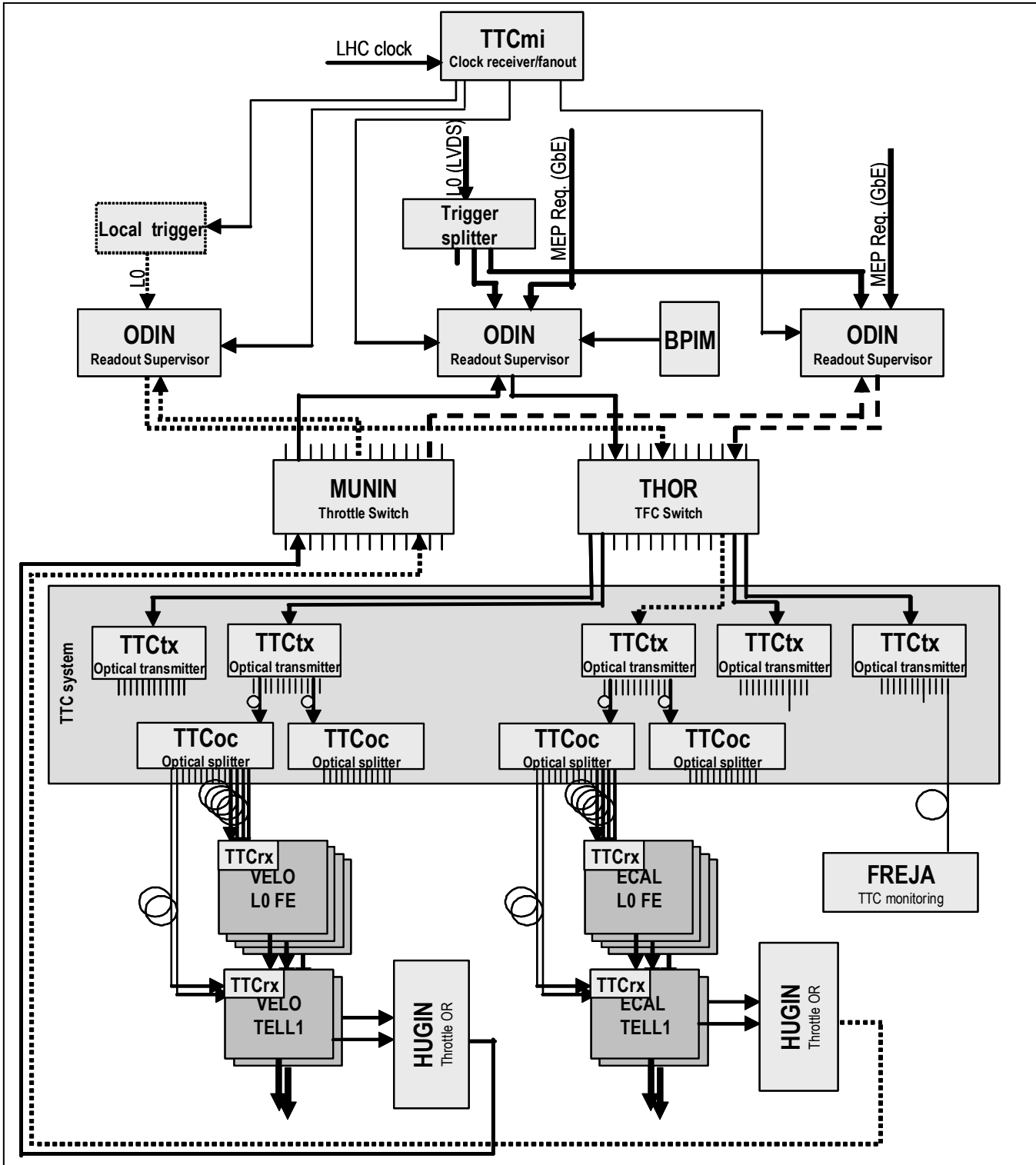
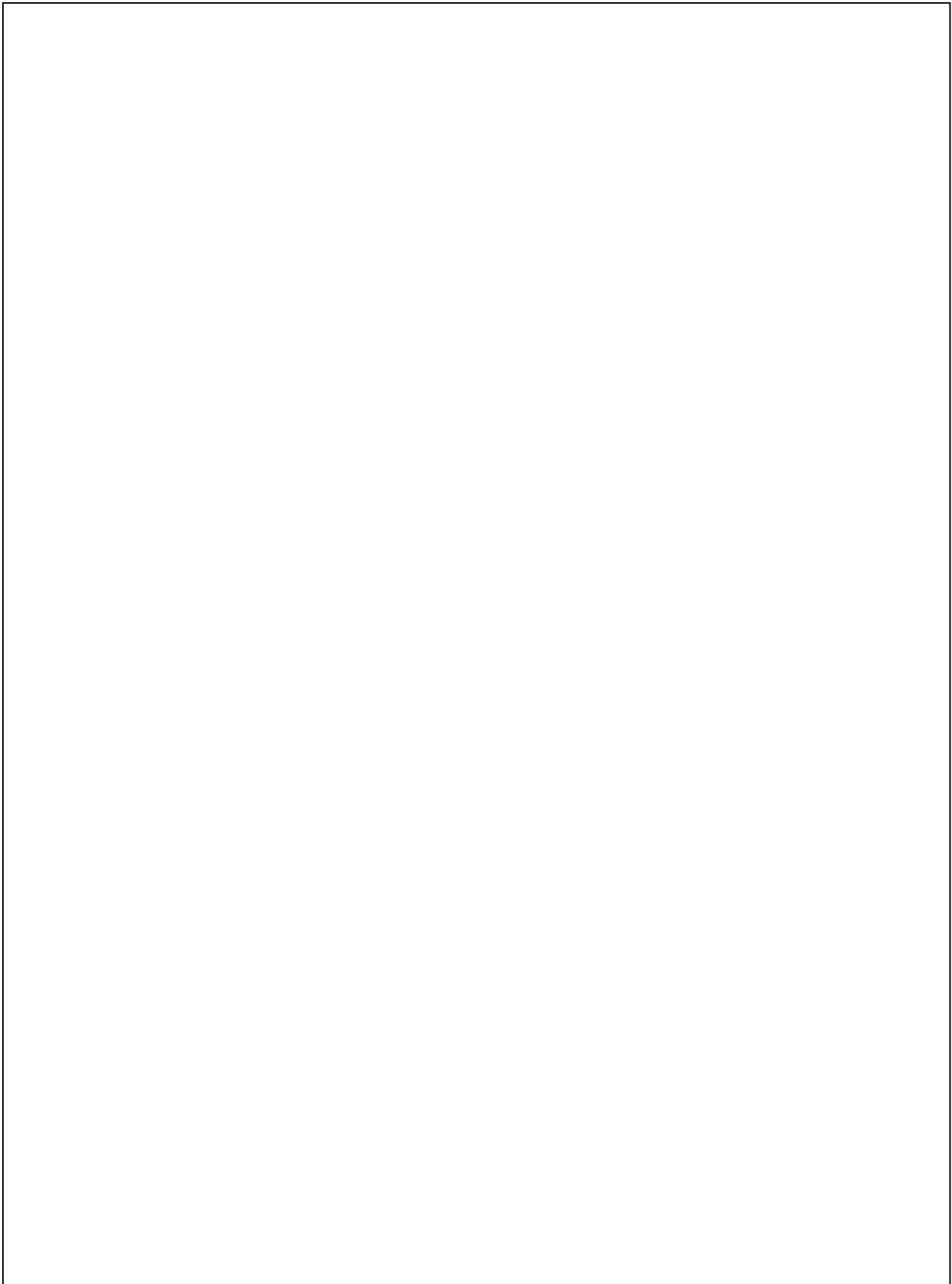


Figure B. 32: Architecture du système TFC

## B.2 Prototypes des fonctions de la librairie tell1.ctl



*Figure B. 33: Prototypes des fonctions de la librairie tell1.ctl*

## B.3 Principales fonctions pour la communication avec HUGIN

```

/* ***** */
/* Startup functions: */
/* ===== */
/* void HUGIN_GetHwInfo(string hugin_name) */
/* void HUGIN_SystemReset(string hugin_name) */
/* void HUGIN_SoftReset(string hugin_name) */
/* void HUGIN_ResetAllCounters(string hugin_name) */
/* void HUGIN_UpdateAllCounters(string hugin_name) */
/* void HUGIN_SubscribeAllCounters(string hugin_name) */
/* void HUGIN_UpdateAll(string hugin_name) */
/* */
/* Run control functions: */
/* ===== */
/* void HUGIN_AckError(string hugin_name) */
/* void HUGIN_RunStart(string hugin_name) */
/* void HUGIN_RunEnd(string hugin_name) */
/* */
/* Configure functions: */
/* ===== */
/* void HUGIN_EnableInputsAll(string hugin_name, int enable, int level=0) */
/* void HUGIN_EnableInput(string hugin_name, int input, bool enable, int level=0) */
/* void HUGIN_EnableOutputsAll(string hugin_name, int enable, int level=0) */
/* void HUGIN_EnableOutput(string hugin_name, int output, bool enable, int level=0) */
/* */
/* "level = 0/1" corresponds to the two separate throttle lines that the throttle
/* network provides. Currently only level 0 is used.
/* */
/* Status functions
/* =====
/* int HUGIN_GetThrottleInput(string hugin_name, int input, int level=0)
/* int HUGIN_GetThrottleOutput(string hugin_name, int input, int level=0)
/* */
/* =====
/* Put the name of the HUGIN you are using */
const string hugin_name = "HUGIN/HuginV1_02";

/* TRUE = use recipe cache, FALSE = use central ConfigurationDB */
const bool recipeCache = TRUE;

const string hierarchyType = "HARDWARE";

```

Figure B. 34: Prototypes des fonctions de communication avec HUGIN



## B.4 Adresses utiles pour la procédure de test des cartes

### B.4.1 Accès au registre d'horloge sur le bus I2C

Device	ADDRESS
<b>Description:</b> Bus used for several devices.	
TELL1 board ID prom	0x50
TTCrx base address	0x58
FEM Beetle base address	0x70

### B.4.2 Registre throttle pour communiquer avec HUGIN

0x000018

Bit	Name	Description	Type	Default
<b>Register Description:</b> This is L0 board throttle control & status register. It will process 4 PPx_L0_throttle signals coming from 4 PP-FPGA to generate a final L0_throttle signal.				0x07000009
31	SL_THRO	Throttle from MEP buffer	R	\
30	PP3_THRO	Throttle from cluster_derandomizer	R	
29	PP2_THRO	Throttle from cluster_derandomizer	R	
28	PP1_THRO	Throttle from cluster_derandomizer	R	\
27	PP0_THRO	Throttle from cluster_derandomizer	R	\
26-8	MEP_USE_THR	Threshold for MEP buffer	R/W	0x70000
7-4	RESERVE0			
3	THRO_CNT_EN	1 = enable the throttle counter	R/W	1
2	THRO_CNT_CLR	1 = Clear (see register below)	R/W	0
1	THRO_SIMU_GEN	To fake a throttle (works also with en=0)	R/W	0
0	THRO_EN	1 = Enable the throttle source	R/W	1
R = Read Only;      W = Write;      R/W = Read/Write;      N = Not exist;				

### B.4.3 Registre utilisé pour vérifier les quatre ports Ethernet des TELL1

0x102000 – 0x10207F

word	31	24	23	16	15	8	7	0
0	DA[47:16]							
1	DA[15:0]				SA[47:32]			
2	SA[31:0]							
3	TYPE				VVERSION/IHL		Type of Service	
4	TOTAL_LEN[15:0]				IDENTIFICATION[15:0]			
5	FLAG[2:0]	FRAGMENTOFFSET[12:0]			TTL		PROTOCOL	
6	HEADER_CHECKSUM[15:0]				IP-SA[31:16]			
7	IP-SA[15:0]				IP-DA[31:16]			
8	IP		DA[15:0]			Reserved		
9- 31	Reserved							