DUT Mesures Physiques
Option : Matériaux et contrôles physico-chimiques

RAPPORT DE STAGE

Stage effectué du 3 Mai au 23 Juillet 1999

MONTOYA Michael

Laboratoire de Physique Nucléaire et de Hautes Energies
4, place Jussieu – Tour 33 RdC
75252 Paris Cedex 05

RESUME

J’ai effectué mon stage au sein du service électronique du Laboratoire de Physique Nucléaire et de Hautes Energies de l’université Paris VI-VII.

Le sujet qui m’a été confié se résume comme suit : conception d’un banc de test automatisé. Ce dernier se décompose en deux grandes parties.

Premièrement, un logiciel doit être créé de manière à gérer l’ensemble du banc de test, c’est à dire : la carte à tester, les diverses interfaces de communications et les instruments de mesures.

Deuxièmement, il s’agit de réaliser une carte de multiplexage contrôlée par le programme précédent pour effectuer différentes mesures de manière autonome.

Mon travail a consisté, dans un premier temps, à me documenter sur le logiciel LABVIEW, utilisé pour la réalisation de l’ensemble du logiciel de banc de test. Puis j’ai réalisé différents programmes permettant de contrôler soit une interface GPIB, soit une interface VME. J’ai ensuite inclus ces différents programmes dans une interface utilisateur déjà réalisée en grande partie par Monsieur Jean-François Huppert.

Dans un second temps, j’ai conçu puis réalisé à l’aide de Messieurs Hervé Lebbolo et Alain Vallereau la carte de multiplexage à l’aide du logiciel Cadence sur plate-forme SunOS.

REMERCIEMENTS

J’adresse mes remerciements au Laboratoire de Physique Nucléaire et de Hautes Energies pour m’avoir permis d’effectuer mon stage au sein du service Electronique.

Je remercie plus particulièrement :

Monsieur Hervé Lebbolo, mon maître de stage, qui a fait preuve d’une grande disponibilité à mon égard, pour m’avoir fait confiance dans la réalisation d’une partie du banc de test.

Messieurs Jean-François Huppert, Alain Vallereau et Philippe Bailly, qui grâce à leurs compétences techniques sur les différents outils utilisés, m’ont permis de mener à bien mon stage.

Je remercie également Madame Colette Goffin et Monsieur David Martin.

Enfin, je remercie l’ensemble du personnel du LPNHE avec qui j’ai été amené à travailler, pour avoir fait preuve de disponibilité et d’attention à mon égard tout au long de mon stage.

SOMMAIRE

❹ REALISATION DES ESSAIS ET RESULTATS.

1. Résultats de linéarité en courant.
2. Résultats de linéarité de l’amplitude des impulsions.

❺ CONCLUSION TECHNIQUE

❻ CONCLUSION GENERALE

❼ BIBLIOGRAPHIE

❽ ANNEXES

INTRODUCTION

J'ai été accueilli du 3 Mai au 23 Juillet 1999 au sein du Laboratoire de Physique Nucléaire et de Hautes Energies de l'université Paris VI-VII.

Pendant ce stage, j'ai conçu puis réalisé une partie d'un banc de test visant à établir le bon fonctionnement de cartes électroniques servant à la calibration d'un calorimètre à Argon Liquide.

Tout d'abord, je vais présenter le laboratoire dans lequel j'ai effectué mon stage, ainsi que le projet sur lequel j'ai été amené à travailler.

J'exposerais ensuite les différentes étapes de la conception de la carte de multiplexage, à savoir : le cahier des charges, le choix des composants puis la Conception Assisté par Ordinateur.

Pour terminer, je détaillerais les différentes étapes de la réalisation de l'interface utilisateur du banc de test. Cette étude sera suivi des différents résultats obtenus grâce à ce banc.

PRESENTATION

DU

L.P.N.H.E.

I. Le C.N.R.S. (Centre Nationale de la Recherche Scientifique)

A la suite de leurs aînés, Jean Perrin, fondateur de C.N.R.S. en 1949, prix Nobel de physique en 1926, et Frédéric Joliot-Curie, premier directeur général du C.N.R.S. de l'après-guerre, prix Nobel de chimie en 1935, nombreux sont les chercheurs éminents qui à un moment ou à un autre de leur carrière ont été associés au C.N.R.S.

Couvrant tous les domaines de la science regroupés au sein de départements scientifiques, le C.N.R.S. peut développer de façon privilégiée les collaborations entre spécialistes de disciplines variées. Cette interdisciplinarité est la source de nouveaux champs d'investigation scientifique et permet de répondre aux besoins de la société et de l'industrie.

Le C.N.R.S. est présent dans toutes les disciplines majeures : sciences physiques et mathématiques, physique nucléaire et corpusculaire, sciences pour l'ingénieur, sciences chimiques, sciences de l'Univers, sciences de la vie, sciences de l'homme et de la société.

Le C.N.R.S. regroupe divers départements tels que l'INSU (Institut National des Sciences de l'Univers) pour la science de l'Univers ou bien encore l'IN2P3 (Institut National de Physique Nucléaire et de Physique des Particules) pour la physique nucléaire et corpusculaire.

Créé en 1971, l'IN2P3 est un institut du C.N.R.S. dont la mission est de promouvoir et de fédérer les activités de recherche en physique nucléaire et en physique des particules.

Au cœur du programme scientifique des laboratoires de l'institut, les expériences de recherche fondamentale visent à identifier les constituants fondamentaux de la matière, à étudier leurs interactions, à comprendre les édifices qu'ils forment, c'est à dire les premiers niveaux de structure de la matière.

Pour réaliser ces expériences, les laboratoires de l'IN2P3 développent des accélérateurs et des détecteurs, outils de base dont les performances déterminent les progrès de la discipline.

L'IN2P3 regroupe divers établissements dont le LPNHE.

II. LE LPNHE

1. Généralité

Le laboratoire possède un important programme en physique des particules et en astroparticules. En physique des particules, les équipes utilisent les grandes installations internationales tant en Europe qu'aux Etats Unis, CERN, DESY, SLAC et dernièrement Fermilab. Toutes les expériences auxquelles le LPNHE participe se déroulent au sein de collaborations internationales où d'autres laboratoires sont représentés, tant de l'IN2P3 que du C.E.A.

- Au CERN :

  Physique des collisions électron-positron (DELPHY).
  Recherche des oscillations de neutrinos (NOMAD)
  Physique au LHC (ATLAS)

- Hambourg : DESY

  Structure du proton (Hera H1)

- SLAC

  Violation de symétrie CP (BABAR)

- Fermilab : D0

  Expérience D0

- Site de Thémis : astroparticules

  Proton de hautes énergies dans l'univers (CAT)
  Programme supernovae de mesure des paramètres cosmologiques
  Recherche et développement à l'Observatoire Auger pour l'étude des rayons cosmiques aux énergies extrêmes.

Toutes ces expériences nécessitent, pour leurs bon déroulement, la mise en œuvre de techniques avancées, en mécanique, électronique et informatique.

Dans chacun des thèmes de recherches, le laboratoire est présent par un groupe structuré de chercheurs et d'ITA (ingénieurs, techniciens et administratifs) mettant à profit les compétences des services techniques. Une part très importante de l'activité de recherche nécessite le maintien et le développement de l'outil informatique.

Durant ces quelques dernières années, le nombre d'expériences a augmenté. Toutefois, il faut souligner que certaines arrivent en fin de parcours (BABAR est

maintenant terminée). Un certain redéploiement est souhaitable pour une bonne vitalité scientifique. Par exemple, l'expérience D0, à part ses mérites propres, est un excellent banc d'essai pour les chercheurs d'ATLAS.

Enfin, par son insertion dans l'université, et le nombre important d'enseignants-chercheurs, le laboratoire a un rôle primordial dans la formation des jeunes, tant sur le plan de l'enseignement que celui de l'encadrement de stagiaires et de thèses.

Le LPNHE est présent dans trois formations doctorales : « Champs Particules Matières » (CPM), « Grands Instruments » (GI) et « Modélisation et Instrumentation en Physique » (MIP). Le responsable du DEA MIP est un Professeur du laboratoire, ce qui fait du LPNHE le point d'ancrage de ce DEA.

2. Expérience D0 au Tévatron.

Après la découverte en 1995 du 6ème quark, le « top », par les expériences CDF et D0 au Tévatron près de Chicago, le laboratoire Fermi s'est lancé dans un programme majeur d'amélioration de l'accélérateur et des détecteurs. La luminosité intégrée devrait être multipliée par un facteur supérieur à 20 et l'énergie de chaque faisceau passera à un TeV. Ce programme durera jusqu'en l'an 2000, date à laquelle reprendra la prise de données.

Le détecteur est également en cours d'amélioration avec notamment le remplacement du système de reconstruction des traces par un système plus performant à base de fibres optiques, l'installation d'un champ magnétique solénoïdal de 2 Tesla et l'adjonction d'un détecteur de microvertex au silicium qui permettra d'identifier avec une bonne efficacité les quarks b. L'excellent calorimètre à Argon Liquide reste inchangé sauf pour sa partie électronique.

Au cours de l'été 1997, une équipe du laboratoire, conjointement à d'autres équipes de l'IN2P3 et du DAPNIA, a proposé de participer à cette seconde phase de fonctionnement du Tévatron sur l'expérience D0. Cette participation à une expérience de collisions hadroniques à des énergies de l'ordre du TeV est importante car elle offre aux équipes de physiciens et de techniciens français une transition vers les expériences sur le LHC qui ne commenceront que vers 2005.

L'accord de principe ayant été récemment donné par les conseils scientifiques du laboratoire et de l'IN2P3, le travail s'articulera principalement autour du développement de logiciels (la contribution française sera dédiée en grande partie à l'écriture des logiciels de reconstruction du calorimètre) et de l'analyse physique, qui préparera à celle du LHC et qui pourrait réserver des surprises de première grandeur : découverte de la Supersymétrie ou du boson de Higgs, sans oublier les sujets traditionnels tels que l'étude du quark « top » sur laquelle le groupe souhaite s'impliquer fortement.

Le laboratoire a également pris en charge la mesure précise de la masse du W sur les données déjà prises lors des premières mesures. Cette analyse devrait permettre d'atteindre une précision de 80 MeV, ce qui renforce les contraintes indirectes sur la masse du boson de Higgs. Le laboratoire collabore également avec Orsay, pour l'étude et la réalisation d'un nouveau système de calibration en ligne du calorimètre à Argon

Liquide avec une précision meilleure que 1%. Ce système est rendu nécessaire par l'augmentation de la fréquence de croisements des paquets de particules dans l'accélérateur. C'est sur ce projet de calibration que l'on m'a proposé de participer.

3. Le système de calibration par impulsion

La capacité à calibrer le calorimètre électronique D0 est de première importance pour déterminer avec une précision optimale l'énergie d'une particule, puisqu'une réponse non uniforme dégrade à la fois l'échelle et la résolution d'une mesure.

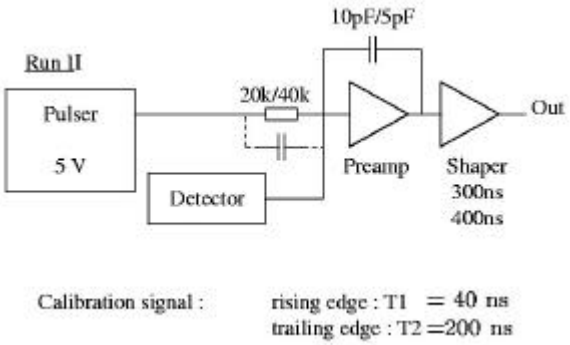
La méthode utilisée pour calibrer le système électronique est basée sur l'envoi d'une quantité précise de charge de valeur connue vers chaque préamplificateur. Le niveau de la charge doit être ajustable de manière à couvrir l'ensemble de la gamme dynamique de l'électronique. La rapidité du signal requiert un control précis de la forme du signal de calibration. La charge délivrée au préamplificateur sert à la fois de signal de calibration mais aussi d'indicateur du bon fonctionnement de chaque canal. Cela permet par exemple de diagnostiquer le mauvais fonctionnement d'un canal ou même son non fonctionnement.

Premier système de calibration :

Douze générateurs d'impulsions servaient à couvrir l'ensemble du calorimètre. Chaque boîte de préamplificateur possède donc son propre système de calibration. Chaque signal de calibration était d'amplitude et de largeur fixe : l'impulsion avait une durée de 400 ns et une amplitude maximale de 135 V dans une charge de 50 Ω avec une fréquence de 1 kHz. Ce signal était de forme rectangulaire avec un temps de monté et de descente de l'ordre de 10 ns. Dans ce premier système de calibration, une seule impulsion permettait de tester simultanément 144 préamplificateurs.

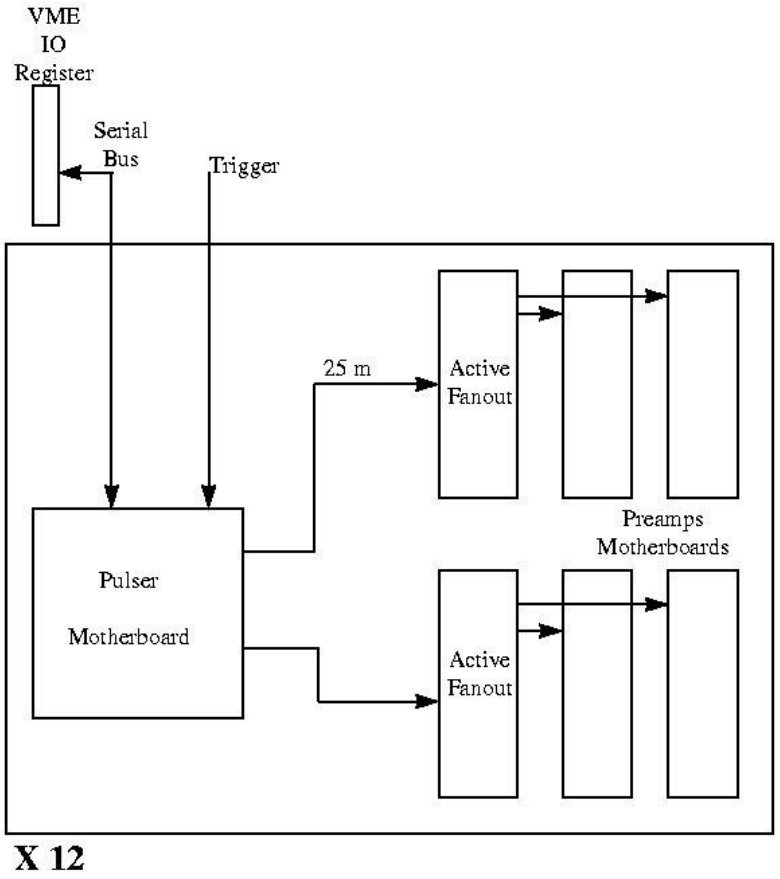
Modification pour améliorer le premier système de calibration.

Il est alors nécessaire de modifier le premier système pour améliorer les résultats. La modification la plus importante est la diminution significative de la valeur des résistances d'injections. Elles sont maintenant localisées sur la carte du préamplificateur de manière à réduire l'influence en tant que capacité parasite de cette dernière, qui a tendance à augmenter avec un signal plus court. Ces résistances permettent alors d'utiliser un signal de calibration basse tension (5V au lieu de 135 V) qui est bien plus facile à générer.



Second système de calibration :

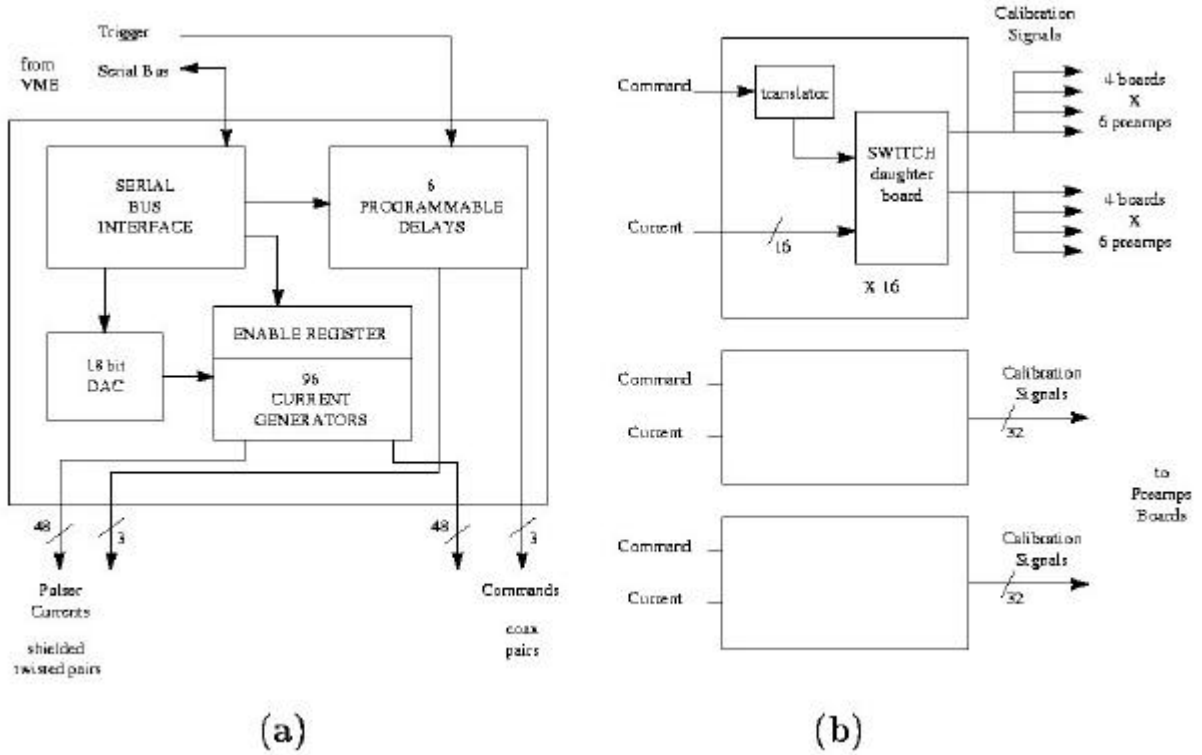
Le système de calibration du calorimètre D0 comprend douze sous-systèmes identiques. Chacun d'eux est composé d'une carte *contrôle mère* (pulser motherboard) qui envoie à la fois des signaux de commandes et des courants continus à deux cartes appelées : *active fanouts*. Ces dernières remplacent les anciens *fanouts* passifs, et sont localisés comme avant dans les boîtes de préamplification. La carte *contrôle mère* est connectée, via un bus série, à la carte VME IO Register, de manière à contrôler l'amplitude et le délais du signal de calibration, ainsi que pour sélectionner le canal sur lequel nous voulons envoyer une impulsion. Voici une vue schématique du nouveau système de calibration :



Les cartes *contrôle mère* comportent donc :

- Une interface par bus série.
- Un registrer de validations des voies.
- Un Convertisseur Numérique-Analogique (CNA) de 18 bits.
- Six délais programmables.
- 96 générateurs de courants pilotés par le CNA.

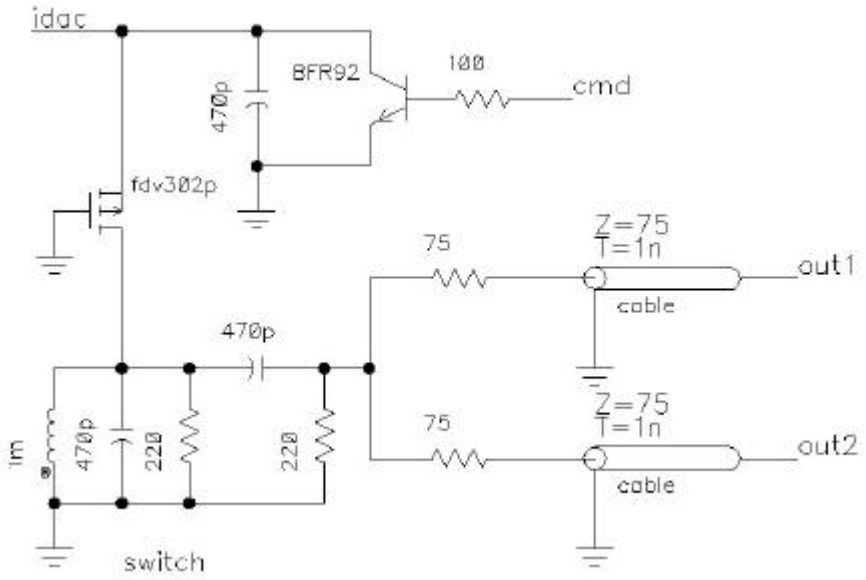
A la réception du signal de déclenchement (trigger), trois commandes de délais sont envoyées vers les deux *active fanouts* correspondants. Les commandes sont en ECL différentiel et sont envoyées à travers des câbles existant de 75 Ω. 48 courants contrôlés sont donc envoyés vers chaque *fanout* grâce à des paires de câbles blindés. Le principal avantage de ce schéma est que seul les courants continus et les signaux de commandes sont envoyés à travers de longs câbles. Les impulsions de calibration sont générées par les cartes *active fanout* juste à côté des préamplificateurs. La conversion de tension en courant est réalisée avec un amplificateur opérationnel à faible décalage, un transistor Darlington et un transistor précis à 0,1 %. Les courants peuvent être éteints grâce au registre de validation des voies.



Vue schématique de la carte *contrôle mère* (a) et la carte *active fanout* (b)

Chaque carte *active fanout* est divisée en trois parties identiques. Chacune reçoit une commande et 16 courants continus. Un tiers de la carte est constitué d'un convertisseur de logique ECL vers une logique TTL et de 16 cartes filles. A la réception du signal de commande, chaque carte fille délivre deux signaux de calibration. Le signal TTL de commande dirige alors le courant continu dans le transistor NPN vers la masse. Ceci bloque alors le PMOS, créant ainsi une variation de courant à l'intérieur de

l'inductance de même amplitude que le courant continu. Cette variation est alors filtrée par deux résistances et deux condensateurs (R1, R2, C1, C2 sur la figure ci dessous). Ceci permet d'avoir un signal ayant un temps de montée très court (60 ns) de forme exponentielle suivie d'un signal de forme exponentielle décroissante, d'une durée d'environ 200 ns. L'inductance a été choisie assez large pour ne pas affecter la forme de l'impulsion. Chaque signal de calibration est envoyé par un câble court (1 ns) vers quatre cartes de préamplification, permettant d'atteindre six préamplificateurs sur chaque carte. Ainsi, chaque impulsion est envoyée simultanément vers 48 préamplificateurs.



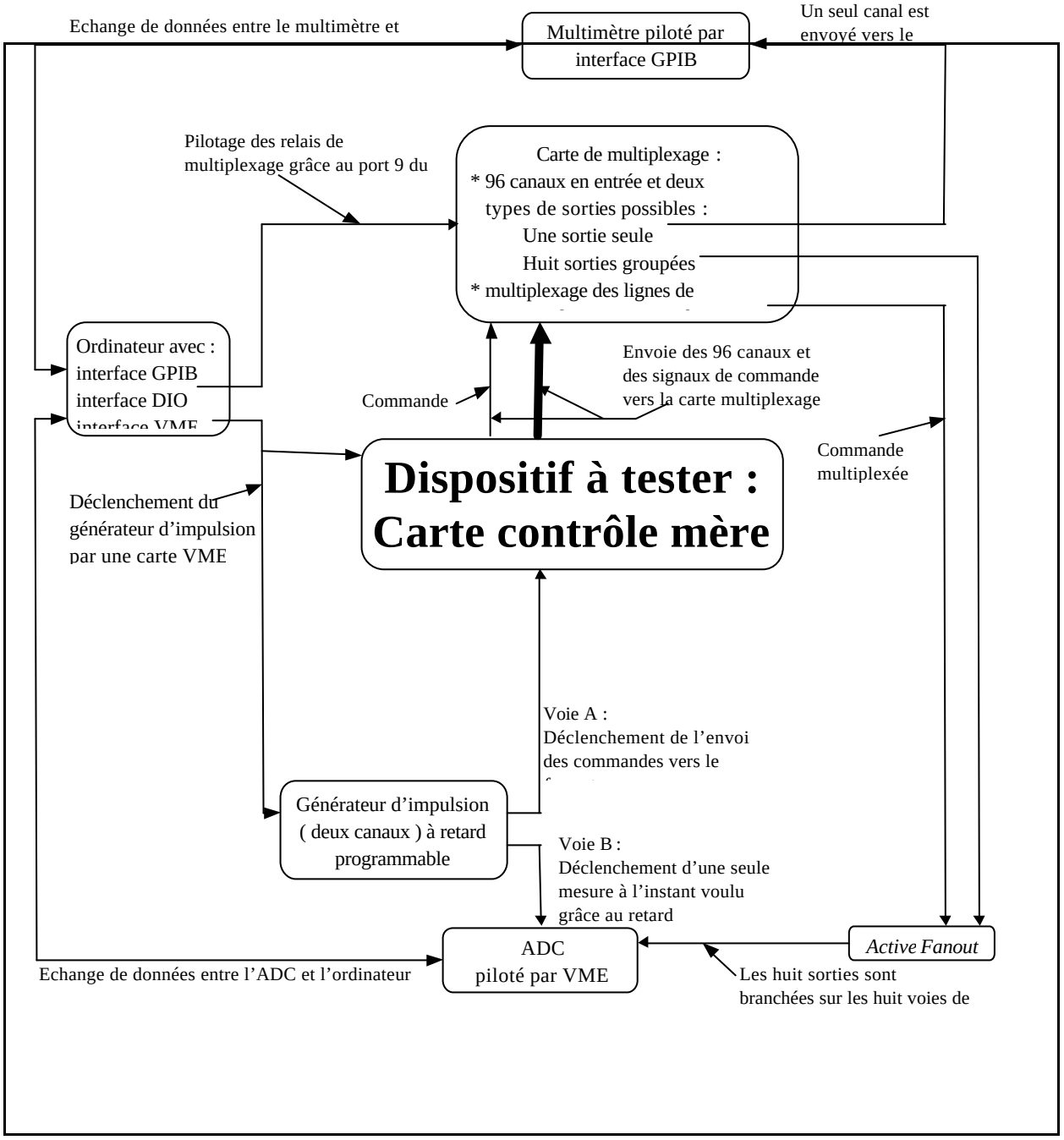
4. But de mon stage.

Le but de mon stage a été de concevoir un banc de test permettant de contrôler le bon fonctionnement de cartes électroniques servant à la calibration d'un calorimètre à Argon liquide.

Il faut donc d'une part tester la linéarité en courant des 96 voies, mais d'autre part, tester la linéarité de l'amplitude des impulsions émises par les cartes *actives fanout*.

Dans un premier temps, nous avons donc cherché à effectuer ces tests séparément, mais pour des raisons pratiques, nous avons ensuite opté pour un banc de test entièrement automatisé. De ce fait, nous avons eu recours à une carte de multiplexage qui permet d'enchaîner l'ensemble des tests.

Voici le schéma fonctionnel du banc de test :



REALISATION

DE LA CARTE DE

MULTIPLEXAGE.

I. But du multiplexage.

La carte *contrôle mère* possède :

- 96 canaux pour les sorties de courant.
- 6 sorties de commandes.

Le banc de test doit contrôler la linéarité de chacun des canaux, les uns après les autres, via un multimètre piloté par GPIB.

La carte *active fanout* possède quant à elle :

- 16 entrées de courant.
- 2 entrées de commandes.

Les courants et les commandes sont envoyés par la carte *contrôle mère*.

Le banc de test doit aussi permettre de mesurer la linéarité de l'amplitude des impulsions émises par cette carte à l'aide d'un ADC (Analog to Digital Converter) de précision contrôlé par VME.

De plus, le banc de test devra servir à contrôler quinze cartes *contrôle mère*.

Ainsi, la réalisation d'un multiplexage s'avère indispensable pour permettre d'effectuer les tests les uns à la suite des autres de manière automatique.

II. Le cahier des charges.

La carte de multiplexage doit permettre d'automatiser le banc de test.
Il a donc fallu prendre en compte différents paramètres avant de déterminer de quelle manière allait être réalisée cette carte.

Le cahier des charges est le suivant :

- La sélection des voies doit s'opérer grâce à l'interface DIO (car son utilisation est relativement simple et pratique).
- Il faut que l'on puisse sélectionner les sorties, soit une par une, soit huit par huit.
- La carte doit aussi multiplexer les signaux de commandes
- Le contrôle de cette carte doit être réalisé à partir de huit bits (correspondant à un port du DIO).

- L'ensemble des connecteurs de la carte doivent être compatibles avec ceux de la carte *contrôle mère*, de la carte *active fanout* mais aussi avec les différents appareils de mesures.

- Le coût de la réalisation ne doit pas être trop excessif.

III. Conception et réalisation de la carte.

1. Les outils de conception.

La conception de la carte a entièrement été réalisée sur un poste de CAO (Conception Assistée par Ordinateur) grâce au logiciel CADENCE Release 97A. Il s'agit d'un logiciel qui s'exécute à partir d'une plate-forme SunOS.

CADENCE apporte tous les outils nécessaires à la conception, au développement et à la réalisation d'une carte électronique. Ce logiciel intervient dans les différentes étapes de la conception électronique. Il permet de réaliser le câblage logique du circuit, de déterminer le boîtier de chaque composant, de simuler un circuit analogique ou numérique, de réaliser le routage du circuit, etc …

C'est donc un outil indispensable pour la conception d'un tel circuit.

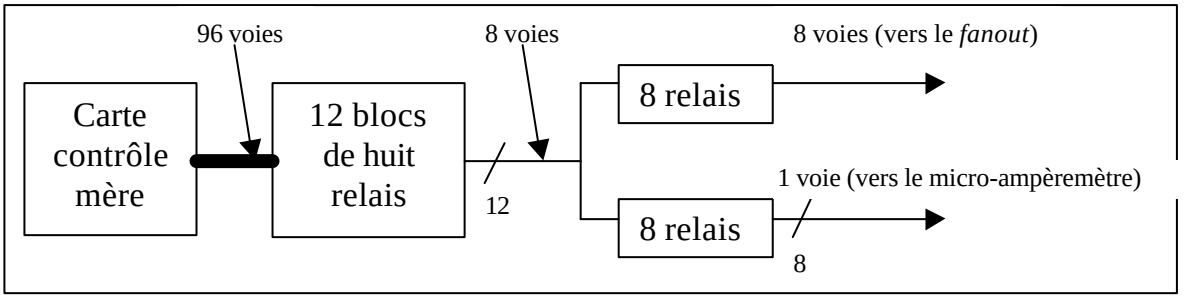
2. Choix des composants et réalisation de la carte.

Multiplexage des 96 canaux :

Le moyen le plus simple pour permettre de «connecter » les voies avec les différents appareils de mesures est d'utiliser des relais. Pour cela, nous utilisons des relais blindés. Leur principe de fonctionnement est simple : chacun possède une bobine introduisant une résistance de 380 Ω et un interrupteur. Si un courant traverse la bobine, alors l'interrupteur se ferme, sinon, il reste ouvert.

La carte doit ainsi pouvoir fournir en sortie soit des groupes de huit voies, soit des voies une par une. On choisi donc dans un premier temps de sélectionner les voies par groupe de huit. Etant donné le nombre total de voies (96), il faut donc douze groupes de huit. Ensuite, il faut avoir la possibilité soit de récupérer en sortie ces huit voies, soit de sélectionner une voie parmi ces huit.

Voici le schéma bloc de la carte de multiplexage :



Il faut utiliser des composants qui permettent de contrôler les douze blocs de huit relais, mais aussi les huit relais qui permettent d'aiguiller une voie vers le micro-ampèremètre.

Pour cela, nous utilisons deux composants différents. Le premier est un démultiplexeur 4 vers 16 (74LS154) qui permet de piloter les douze blocs de huit relais (il nécessite donc 4 bits de contrôle). Le second est un démultiplexeur 3 vers 8 (74LS138) qui permet quant à lui de gérer le bloc de huit relais connectés au micro-ampèremètre (il nécessite 3 bits de contrôle).

Enfin, un dispositif doit être mis en place pour permettre de valider soit le bloc de huit relais connecté au *fanout*, soit le composant 74LS138. On utilise donc un huitième bit qui, suivant son état, valide les huit relais ou le 74LS138.

➢ Mise en place du démultiplxeur 4 vers 16.

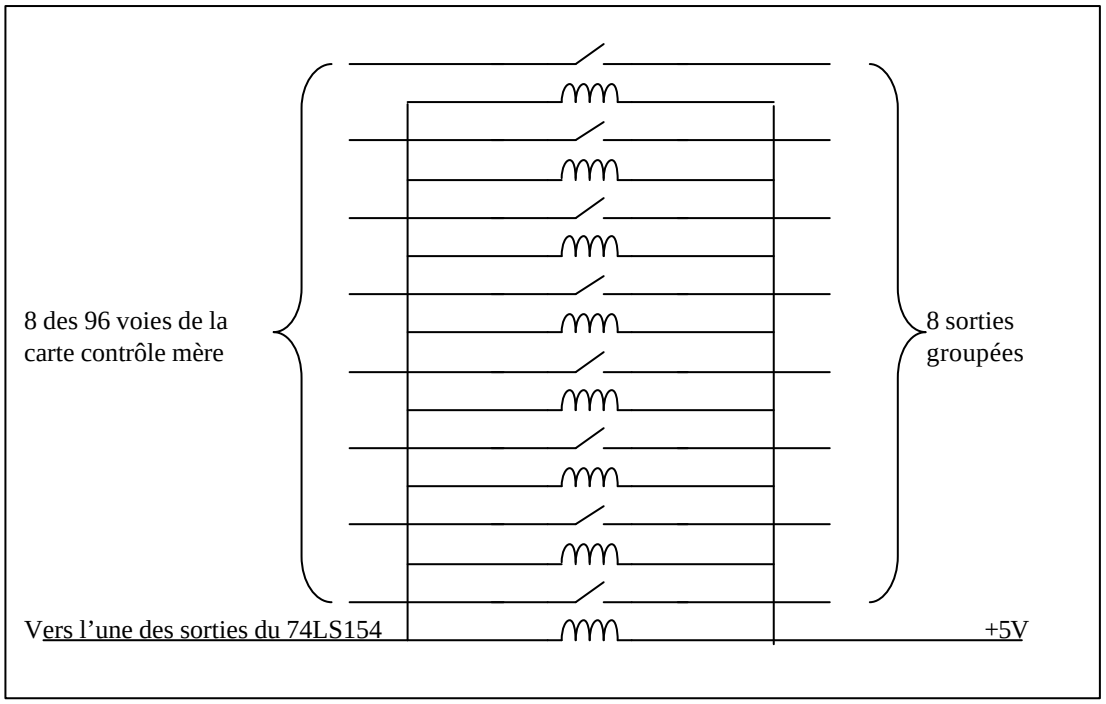
Sa table de vérité est simple :
- Si le composant n'est pas validé : toutes les sorties sont à l'état haut.
- S'il est validé : seule la voie sélectionnée par les 4 entrées est au niveau bas.

On connecte donc les 8 bobines des relais en parallèles. L'une de leurs bornes est reliée au +5V de la carte (VCC) et l'autre est reliée au collecteur d'un transistor. La base de ce dernier est reliée alors à la sortie d'un inverseur via une résistance. L'entrée de l'inverseur quant à elle connectée à l'une des sorties du 74LS154.

Ainsi, lorsque toutes les sorties du 74LS154 sont à l'état haut (+5V), les sorties des inverseurs sont à l'état bas et donc aucun courant n'arrive dans les bases des transistors : il sont donc bloqués et par conséquent, il n'y a aucun courant qui traverse les bobines.

Par contre, si l'une des sorties du 74LS154 est à l'état bas, alors grâce à l'inverseur, il y a un courant dans la base du transistor : il est donc passant. Ainsi, il y a un courant qui traverse les bobines et donc les huit relais se ferment.

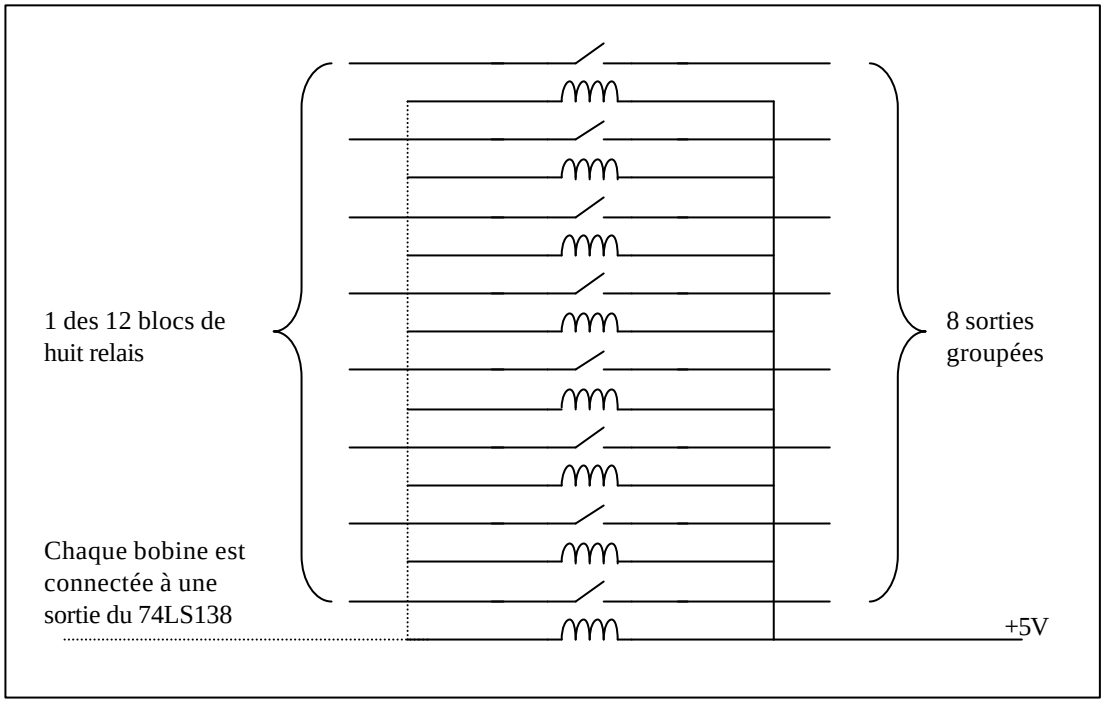
On utilise seulement douze sorties sur les seize disponibles.



➢ Mise en place du démultiplxeur 3 vers 8.

Sa table de vérité est identique à celle du 74LS154, excepté qu'il n'y a que trois entrées pour huit sorties :

Chaque sortie est reliée à l'une des bobines des relais connectés au micro-ampèremètre. L'autre borne des bobines étant reliées au +5V.



Remarque : nous rajoutons une diode de roue libre en parallèle aux bobines de manière à éviter les surtensions dues aux variations de potentiel à leurs bornes.

Multiplexage des signaux de commandes :

Les signaux de commandes sont en ECL différentiel. On les transpose en logique ECL grâce au composant (10H115) puis on réalise un OU câblé. On obtient donc une ligne de commande en logique ECL. Toutefois, à ce stade, nous voulons d'une part, envoyer cette commande dans la carte *active fanout* (donc ECL différentiel) et d'autre part, pouvoir visualiser cette commande sur un oscilloscope (en logique NIM).

La totalité du circuit se trouve dans les feuilles annexes 1 à 5.

Planche 1 (annexe 23) :

Elle est consacrée à la mise en place du composant 74LS154.

On remarque que l'on utilise les bits 3, 4, 5 et 6 du port 9 du DIO pour contrôler le 74LS154. Ce choix vient du fait qu'à ce niveau, les voies sont sélectionnées par groupe de huit. Or nous serons amené par la suite à sélectionner une seule voie parmi ces groupes de huit et nous aurons alors besoins de trois bits supplémentaires. Pour respecter alors la concordance entre la valeur des 7 bits et le numéro du relais choisi, il faut que les trois supplémentaires soit ceux de poids faible.

Application :

Pour sélectionner le relais 37, on envoie alors sur le port 9 du DIO la valeur 100101 : 100 pour valider le bloc de relais n°4 et 101 pour valider le $5^{ème}$ relais de ce bloc de relais.

Planches 2 et 3 (annexe 24 et 25) :

Elles correspondent au câblage des 96 canaux de la carte *contrôle mère* vers les 96 relais correspondants.

Planche 4 (annexe 26) :

Elle représente la partie du circuit qui va permettre de valider soit le bloc de huit relais qui est connecté au *fanout* soit le composant 74LS138. En effet, deux cas peuvent se produire :

Le bit n°7 est à l'état bas : alors le composant 74LS138 est validé. Ainsi, il peut contrôler les huit relais connectés au micro-ampèremètre. En revanche, comme il est à l'état bas, il n'y a donc pas de courant dans la base du transistor Q13 : il est donc bloqué et les huit relais restent alors ouverts.

Le bit n°7 est à l'état haut : c'est l'effet inverse qui se produit. Le composant 74LS138 n'est pas validé. Toutefois, il permet d'introduire un courant dans la base du

transistor Q13 : il est donc passant et les huit relais connectés à la carte *fanout* sont donc fermés.

Planche 5 (annexe 27) :

Cette planche est dédiée au multiplexage des commandes vers la carte *active fanout*.

REALISATION

DU

BANC DE TEST

AVEC

LABVIEW.

I. Qu'est-ce que LABVIEW ?

LABVIEW est un logiciel de développement d'application, comparable à la plupart des systèmes de développement en langage C ou BASIC disponible sur le marché, ou encore à LabWindows de National Instruments. Cependant, LABVIEW se distingue des autres logiciels sur au moins un point important. En effet, la majorité d'entre eux s'articulent autour de langages à base de texte dont la programmation consiste à empiler des lignes de codes, tandis que LABVIEW utilise un langage de programmation graphique, le langage G, pour créer un programme sous forme de diagramme.

Inutile d'être expert en programmation pour pouvoir utiliser LABVIEW. La terminologie, les icônes et les principes inhérents à LABVIEW, tous familiers aux ingénieurs et aux scientifiques, font appel à des symboles graphiques pour décrire les opérations de programmation.

LABVIEW offre des bibliothèques étendues de fonctions et de routines (blocs programmés) capables de répondre à la plupart des besoins en programmation. En ce qui concerne les plates-formes Windows, Macintosh et Sun, LABVIEW comprend également des bibliothèques de fonctions spécifiques à l'acquisition de données et au pilotage d'instruments VXI et GPIB, ou encore d'instruments connectés sur une simple liaison série. Il existe aussi des bibliothèques liées à la présentation, à l'analyse et au stockage des données. LABVIEW intègre une panoplie complète d'outils de développements de programmes conventionnels, de sorte que l'on peut définir des points d'arrêts, animer l'exécution du programme en mettant en évidence le cheminement des données et exécuter pas à pas le programme. Le développement et la mise au point du programme s'en trouvent ainsi facilités.

II. Comment fonctionne LABVIEW :

Un programme LABVIEW est appelé instrument virtuel (VI) tout simplement parce que sa représentation et son fonctionnement ressemblent à ceux des instruments classiques. Néanmoins, les VIs diffèrent en ce sens qu'il tirent leurs fonctionnalités de la programmation informatique. Ils offrent une interface interactive.

- Un VI intègre une interface utilisateur interactive appelée « face avant », d'instruments physiques. La face avant peut contenir des boutons rotatifs, des boutons poussoirs, des graphiques et autres commandes et indicateurs. On saisit les données à l'aide du clavier ou de la souris, puis on visualise les résultats à l'écran.

- Un VI reçoit des instructions de son diagramme, que l'on construit en langage graphique G. Il correspond au code source du VI, et réduit ainsi la programmation à une simple manipulation graphique.

- Le VI présente une structure hiérarchique et modulaire. Nous pouvons l'utiliser comme programme principal ou comme un sous programme à l'intérieur d'un

programme ou de sous programmes. Un VI contenu à l'intérieur d'un autre VI s'appelle un sous VI. Le cadre *icône/connecteur* d'un VI répertorie sous forme graphique les paramètres désirés si bien que d'autres VIs peuvent lui transmettre des données, récupérer des informations, en le considérant comme un sous VI.

Dans notre cas, nous utilisons LABVIEW pour piloter différents éléments et pouvoir ainsi réaliser un banc de test permettant d'effectuer plusieurs types de mesures.

Pour simplifier le banc de test, nous devons réaliser une carte permettant de recueillir les signaux en sortie de la carte *contrôle mère* et pouvoir les envoyer soit vers des appareils de mesures, soit vers la carte *fanout*, laquelle enverra ses signaux directement dans un ADC (Analog to Digital Converter). Chaque donnée sera ensuite récupérée et analysée par l'ordinateur.

III. Réalisation des différentes interfaces.

1. Mesure de courant grâce au multimètre (piloté par GPIB).

➢ Description sommaire du standard d'interface digital IEEE-488.

La communication entre les périphériques aux normes d'interface IEEE-488 (aussi appelée interface GPIB) est réalisée au moyen de messages à travers le système d'interface.

Types de messages :

Le standard d'interface IEEE-488 repose sur un système de bus également appelé IEEE-488 sur lequel transitent des messages dépendant des périphériques (les instructions de programmation, les résultats de mesures, les états machines, les fichiers de données...) et des messages dits d'interface (initialisation du bus, les adressages et les désadressages des périphériques, le positionnement des modes de fonctionnement des périphériques pour une programmation locale ou en télécommande...)

Notions de parleurs, d'écouteurs et de contrôleurs :

Sur le bus IEEE-488 cohabitent trois types de périphériques appelés : parleurs, écouteurs et contrôleurs.

Un parleur envoie des messages de données à un ou plusieurs écouteurs. Le contrôleur gère le flux d'information sur le bus IEEE-488 en envoyant des messages de commandes aux périphériques. Un périphérique peut être écouteur, parleur et/ou contrôleur.

Le rôle d'un contrôleur IEEE-488 peut être comparé à celui d'un centre de communication téléphonique. Le centre (contrôleur) surveille le réseau de communication (bus IEEE-488). Lorsque le centre (contrôleur) note qu'une personne (parleur) veut effectuer un appel (envoyer un message de données), il connecte cette personne (parleur) au receveur (écouteur).

Le contrôleur adresse le parleur et l'écouteur avant que le parleur ne puisse envoyer ses messages de données à l'écouteur. Après que les messages soient transmis, le contrôleur peut désadresser les deux périphériques.

Un contrôleur est nécessaire lorsque le parleur ou les écouteurs doivent changer de rôle durant les échanges. Le contrôleur est généralement représenté par un ordinateur.

Signaux et lignes IEEE-488 :

Le système d'interface se compose de 16 lignes de signaux avec 8 lignes de retour de masse. Les 16 lignes de signaux se décomposent en :

- 8 lignes de données
- 3 lignes d'acquittement
- 5 lignes de gestion d'interface

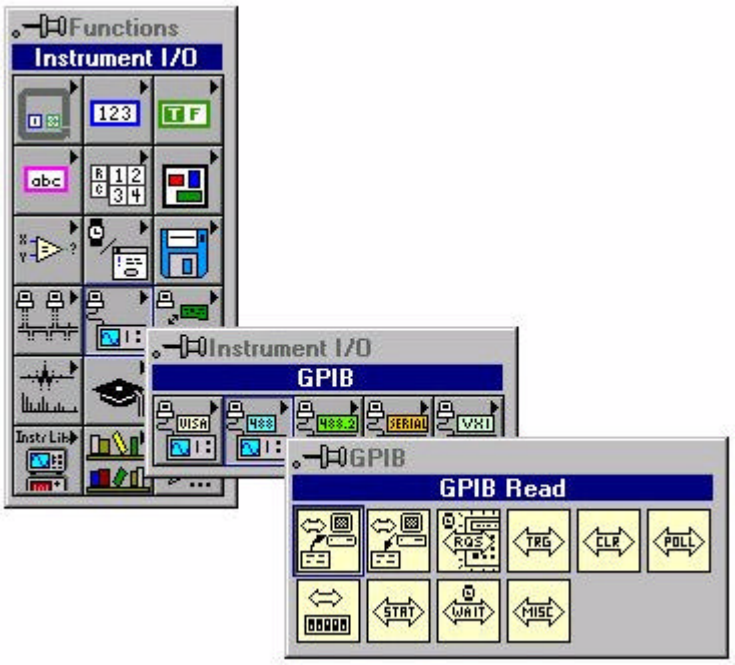
Les lignes de données transportent aussi bien des messages de données que des messages de commandes.

Les trois lignes d'acquittement servent à contrôler le transfert des messages entre les périphériques.

Les lignes de gestion d'interface permettent de déterminer si le message envoyé est une commande ou une donnée, d'initialiser le bus, demander un service au contrôleur, indiquer la fin d'un message.

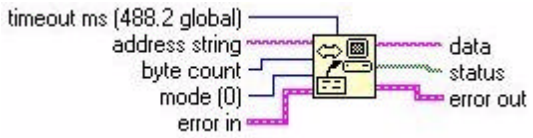
- **LABVIEW et l'interface IEEE-488.**

LABVIEW possède une multitude de VIs permettant de contrôler l'interface IEEE-488.



On remarque ici différents VIs qui permettent de manipuler l'interface IEEE-488 avec aisance. Voici une brève description de quelques VIs utiles :

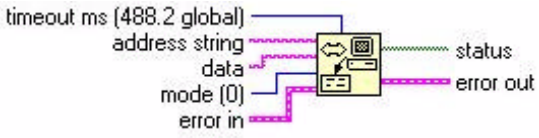
GPIB Read :



Ce VI permet de lire une donnée présente sur l'interface GPIB.

Timeout ms : L'opération de lecture est annulée si elle n'est pas terminée avant le temps indiqué par *timeout ms*.
Address String : C'est une chaîne de caractères qui correspond à l'adresse logique de l'instrument auquel on s'adresse.
Byte count : Spécifie le nombre de bits à lire sur l'interface GPIB.
Mode : Spécifie les conditions selon lesquelles la lecture est considérée comme terminée (par exemple, lors de la rencontre d'un certain caractère).
Error in : C'est une erreur qui serait survenue auparavant. *Error in* permet ainsi de stopper le fonctionnement de ce VI, car l'erreur précédente pourrait provoquer un mauvais fonctionnement.
Data : Il s'agit de la donnée qui est lue.
Status : Il s'agit de l'état de l'instrument auquel on s'adresse.
Error out : S'il y a une erreur lors de la lecture GPIB, on peut grâce à *error out*, reporter cette erreur sur d'autres Vis, permettant ainsi d'éviter un mauvais fonctionnement du programme.

GPIB Write :



Le fonctionnement de GPIB Write est identique à celui de GPIB Read excepté le fait que *data* n'est plus une donnée lue, mais une donnée écrite.

GPIB Clear :

Ce VI permet de remettre à zéro l'interface GPIB.

Il existe aussi d'autres VIs tels que GPIB Initialization, GPIB Misc, GPIB Serial Poll, GPIB Status, GPIB Trigger, GPIB Wait, Wait for GPIB RQS

➢ **Réalisation du VI permettant de contrôler le GPIB.**

Pour cela, j'ai utilisé les trois VIs dont la description est ci-dessus.

La réalisation de l'interface permettant le contrôle de l'acquisition des données en provenance du micro-ampèremètre ne s'est pas faite au hasard. Il s'agit de réaliser un VI qui pourra être utilisé comme sous VI et être appelé par le programme principal. Nous avons donc besoin de pouvoir maîtriser certains paramètres :

- Une initialisation possible à partir du programme principal.
- Pouvoir rentrer l'adresse logique de l'instrument auquel on s'adresse (cela permet d'utiliser ce VI pour plusieurs instruments GPIB).
- Pouvoir lui fournir une commande (ou instruction) à partir du programme appelant.
- Récupérer une donnée en sortie.
- Pouvoir récupérer une éventuelle erreur.

Voici le VI, avec ses connecteurs représentés :

et sa face avant :

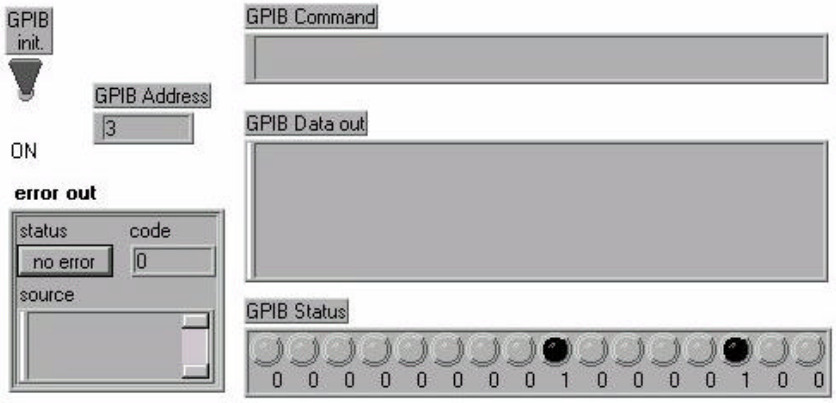
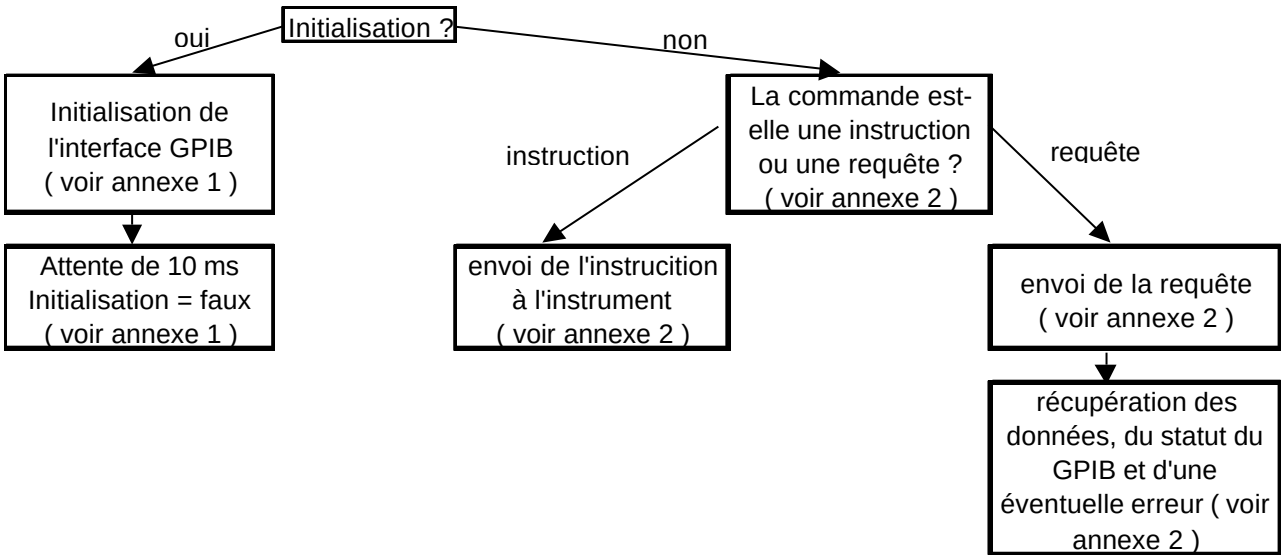


Schéma de fonctionnement de ce programme :



La différence entre la syntaxe d'une instruction et celle d'une requête réside dans la présence d'un point d'interrogation que l'on teste (Voir annexe 2).

➢ Phase de fonctionnement : la commande est-elle une instruction ou une requête ?

Cas d'une requête : la commande contient le caractère « ? ».

Il y a alors deux séquences. La première a pour but d'envoyer la requête (ex : mesure du courant continu). La seconde est destinée à demander le résultat de la mesure (ex : valeur, du courant continu).

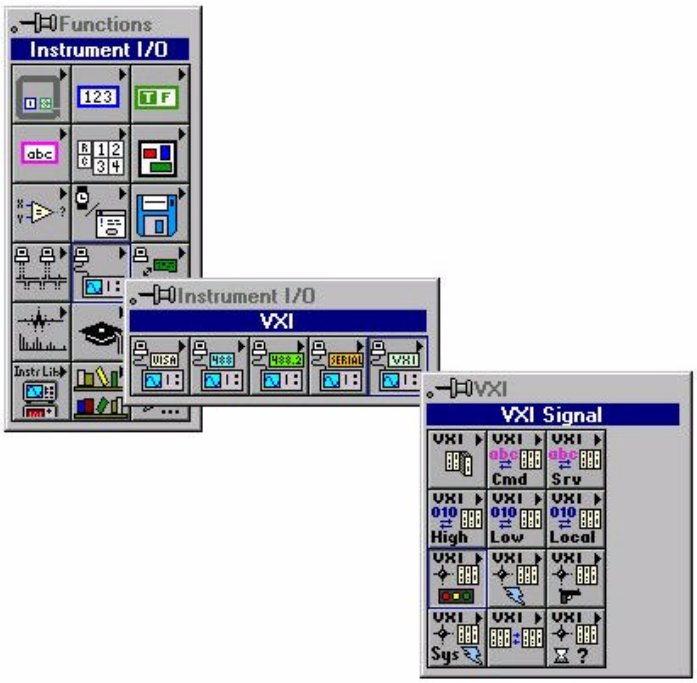
Cas d'une instruction : la commande ne contient pas le caractère « ? ».

Dans ce cas, seule l'instruction (GPIB Command) est envoyée à l'instrument :

2. Mesure de tension grâce à l'ADC (piloté par VME).

➢ LABVIEW et l'interface VME

LABVIEW dispose d'une grande variété de VIs qui permettent d'établir des communications entre l'ordinateur et le bus VME

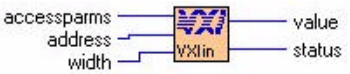


Toutefois, seul est utile dans notre cas la librairie *High Level* VXIbus Access.

Voici cette libraire :

Elle permet d'avoir accès au bus VME grâce à des routines de haut niveau.
On peut donc lire ou écrire sur le bus VME grâce aux VIs suivants :

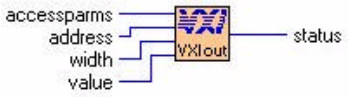
VXI In :



Ce VI permet d'effectuer une lecture du bus VME.
Accessparms : permet de déterminer le mode d'accès vers l'élément. Par exemple, l'accès peut se faire en mode privilégié ou non, que ce soit en transfert de programme, de données ou en transfert par bloc.
Address : représente l'adresse physique de l'élément de bus auquel on s'adresse.
Width : permet d'effectuer des transferts 8, 16 ou 32 bits.
Value : Il s'agit de la valeur lue sur le bus.

Status : Il s'agit de l'état du bus.

VXI Out :

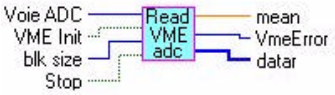


- **Réalisation du VI permettant de contrôler l'ADC du bus VME.**

La réalisation de l'interface permettant le contrôle de l'acquisition des données en provenance de l'ADC ne s'est pas faite au hasard. Il s'agit de réaliser un VI qui pourra être utilisé comme sous VI et être appelé par le programme principal. Nous avons donc besoin de pouvoir maîtriser certains paramètres :

- Une initialisation possible à partir du programme principal.
- Déterminer la voie de l'ADC que l'on veut lire.
- Déterminer la taille des informations à lire.
- Permettre l'effectuer une pause de ce VI lorsqu'il est en cours de fonctionnement, car cela permet de l'utiliser plusieurs fois dans un programme tout en le gardant dans la même configuration.
- Récupérer en sortie la valeur moyenne du signal, ainsi qu'une matrice dont les éléments correspondent aux valeurs de chaque point échantillonné.
- Pouvoir récupérer un éventuel code d'erreur.

Voici le VI et ses connecteurs :



et sa face avant :

datar
0
0
VME Init
Stop
OFF
ON
blk_size
1
size
1
VmeError
0
ReadValue
4
Read One
Read Multiple
Voie ADC
Channel 1
mean
-0.28083
Plot 0
-0.240
-0.245
-0.250
-0.255
-0.260
-0.265
-0.270
-0.275
-0.280
-0.285
-0.290
-0.295
-0.300
0.00000000
0.00000020
0.00000040
0.00000060
0.00000080
0.00000100
0.00000120
0.00000140
0.00000160
0.00000180
0.00000200
0.00000220

Toutefois, ce VI a un fonctionnement un peu spécial. Il ne servira pas à effectuer tout un ensemble de mesures sur l'ADC, mais servira à déclencher la mesure au moment adéquat.

Nous avons choisi cette optique car l'ADC échantillonne à 10 MHz au mieux. Il est alors impossible d'effectuer la mesure d'un signal ayant une durée d'environ 60 ns. Pour cela, nous avons remplacé l'horloge interne de l'ADC par une horloge externe (l'une des deux sorties du générateur d'impulsion).

En effet, l'ADC échantillonne sur un front de monté du signal d'horloge. Ainsi, si l'on envoie une impulsion à la place de l'horloge, l'ADC n'effectuera qu'une seule mesure. L'intérêt est que cette impulsion peut être synchronisé par rapport à l'ensemble du banc de test.

Il existe donc deux façons de programmer l'ensemble.

Dans les deux cas, nous avons recours à un générateur d'impulsion à deux sorties : l'une pilotant la carte *contrôle mère* et l'autre déclenchant une seule mesure sur l'ADC.

➢ Synchronisation de la carte *contrôle mère* et de l'ADC

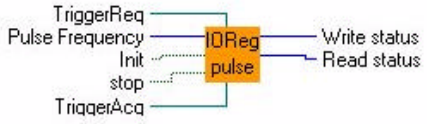
La synchronisation des deux peut être réalisée de deux manières.

La première consiste à utiliser le DIO pour déclencher le générateur d'impulsion. La seconde permet de déclencher ce dernier à partir d'une carte VME. Pour ma part, j'utilise la première méthode. En effet, un VI a déjà été réalisé pour permettre de déclencher le générateur d'impulsion à partir du DIO. J'ai donc légèrement modifié son fonctionnement pour l'adapter aux manipulations désirées.

Ce VI se nomme *IO Reg Pulse* et permet d'envoyer un signal vers le générateur d'impulsion (sur son entrée *External Input*). Il est aussi possible de paramétrer le VI *IO Reg Pulse* de manière à ce qu'il émette un signal périodique.

Il est utilisé à plusieurs niveaux dans le banc de test :

- Il permet de générer des impulsions à partir du DIO ou de l'interface VME, pour permettre l'envoi des données vers la carte *contrôle mère*.
- Il est utilisé pour piloter la carte de multiplexage.
- Il permet aussi de synchroniser les signaux entre la carte *contrôle mère* et l'ADC (toujours à partir du DIO ou VME).



Dans mon cas, seul m'intéresse les trois entrées Pulse Frequency, Init et stop.

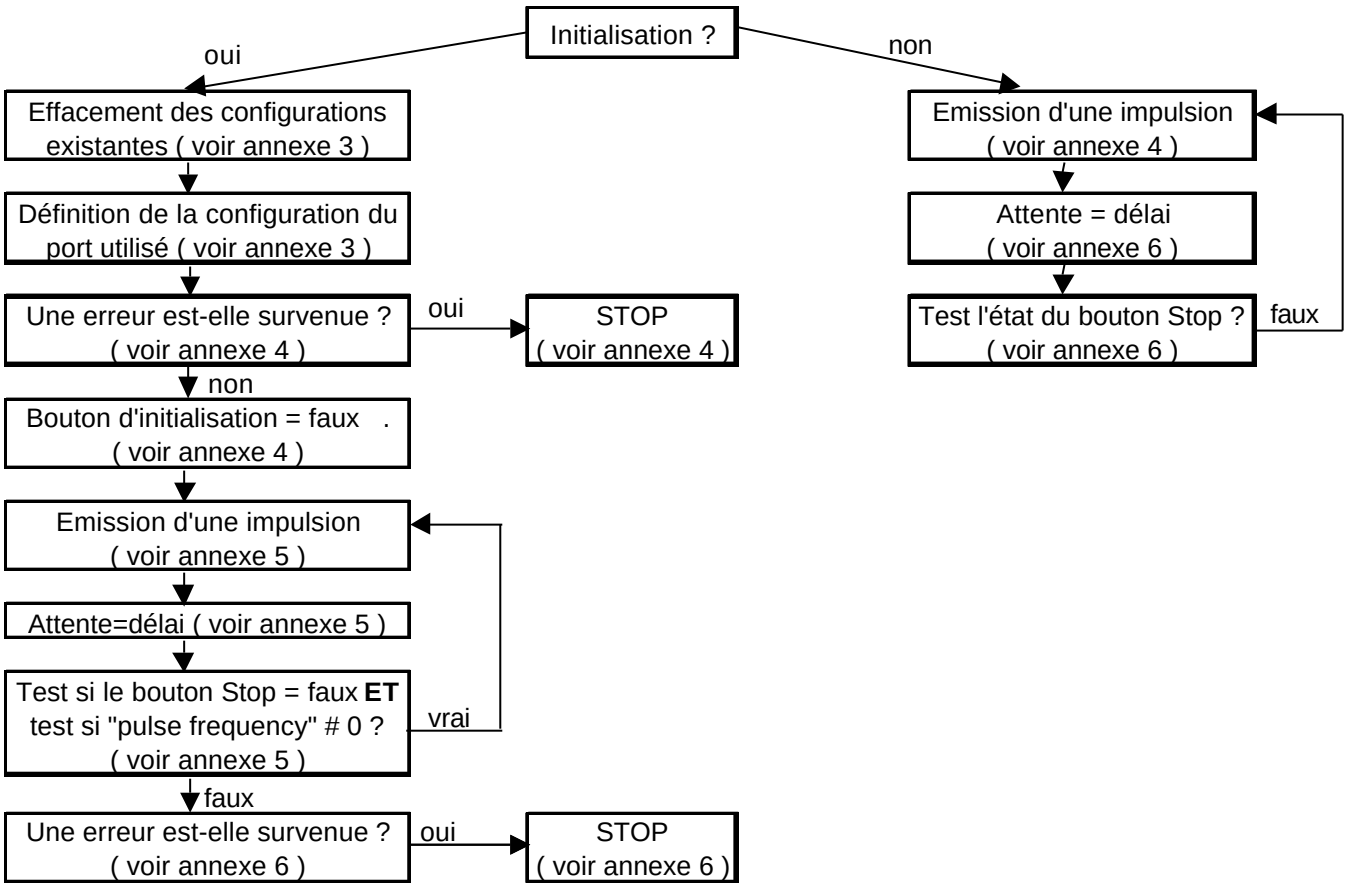
Pulse frequency : il s'agit en fait de la période du signal. Toutefois, si la valeur est nulle, il n'y alors qu'une seule impulsion.

Init : permet d'initialiser le port DIO considéré.

Stop : permet de démarrer l'émission des impulsions ou au contraire de les stopper.

Son principe de fonctionnement est relativement simple : on met à zéro les sorties du port DIO considéré. La séquence suivante, on les met à l'état haut, et la séquence qui suit, on les remet à l'état bas. On effectue ensuite une pause dont la durée permet d'obtenir une fréquence correspondant à la valeur de *pulse frequency* (en ms).

Schéma de fonctionnement du programme.



➢ phase d'initialisation

Séquence n°0 : effacement des configurations existantes (annexe 3).

On efface toutes les précédentes configurations du port DIO considéré.

On utilise pour cela le VI *Port Config*. On lui donne comme arguments : le numéro de la carte considérée (DIO Device), le port considéré de cette carte (DIO port trig) et la taille en bits que l'on veut lui affecter (Port Width ici égal à 0)

Ici, DIO Device est une variable globale définie dans un autre VI. Cette variable dépend directement de l'installation de la carte et reste fixe une fois que cette dernière est installée. Il est nécessaire de préciser que tous les paramètres globaux, adresses de l'ADC, du DIO et d'autres paramètres concernant les instruments utilisés, sont regroupés dans un VI dit VI global.

On place ensuite un VI d'affichage d'erreur qui permet de stopper le programme en cas d'erreur et de l'afficher l'erreur.

Séquence n°1 : définition de la configuration du port utilisé (annexe 3).

On définit alors configuration du port utilisé (port défini par DIO Port Trig), à savoir : un port de huit bits tous sont consacrés à l'écriture (FF signifie que toutes les lignes sont des lignes de sorties). Le VI *port config* renvoie alors une valeur d'identification de ce port (port trig id) qui devra être désormais utilisée pour l'émission de signaux à partir du port du DIO.

Séquence n°2 : une erreur est-elle survenue ?

Cette partie permet de garder uniquement le statut d'une éventuelle erreur.

Si le statut est vrai (une erreur s'est produite), le programme est immédiatement stoppé et affiche le message «UNABLE CONFIGURE DIO PORT FOR TRIGGER (#6) » (annexe 4)

Si le statut est faux (aucune erreur ne s'est produite), le bouton de l'initialisation devient faux et le programme continu. (annexe 4)

Séquence n°3 : émission d'une impulsion, attente = délai puis test si le bouton Stop = faux ET si "pulse frequency" ≠ 0 ? (annexe 5)

Cette séquence est destinée à émettre l'impulsion sur le port DIO choisi.

De plus, il est possible de réaliser un masque (cacher certains bits).

L'impulsion est décomposée en quatre parties : état bas, état haut, état bas puis attente.

On remarque que l'on utilise bien l'identificateur du port (port trig id) pour s'adresser au DIO.

Séquence n°4 : une erreur est-elle survenue ?

Cette séquence a pour but de stopper le programme si une erreur est survenue (annexe 6).

Dans le cas faux, il ne se passe rien.

- Phase de fonctionnement : Emission d'une impulsion, attente = délai puis test l'état du bouton Stop (annexe 6).

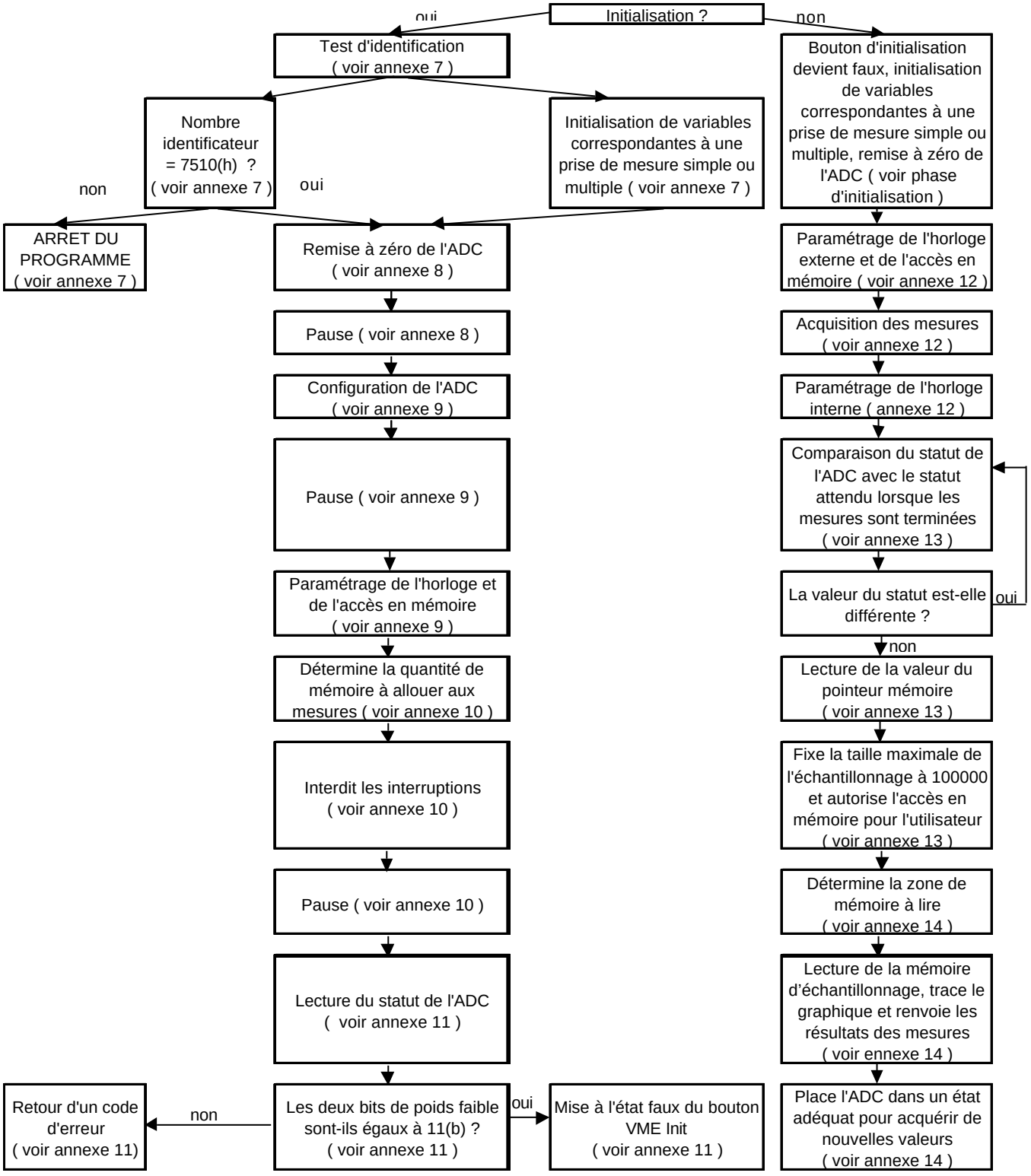
Ici, seule la séquence n°3 de l'émission de l'impulsion diffère par rapport à l'initialisation.

En effet, il faut alors tenir compte d'un éventuel délai que l'on place en fin de séquence.

Ainsi, on boucle sur les séquences d'émission de l'impulsion tant que le bouton stop n'est pas actionné.

- réalisation de l'interface permettant la lecture de l'ADC :

Le schéma de fonctionnement du programme est le suivant :



Ce programme est divisé en deux parties : la phase d'initialisation de l'ADC et le mode de fonctionnement normal.

➢ phase d'initialisation.

Cette phase d'initialisation se décompose en douze étapes. Ces dernières permettent entre autre de vérifier la présence de l'ADC, de le placer dans un état connu, de paramétrer les différents registres en fonction du type de mesures effectuées.

Etape n°0 : test d'identification (annexe 7).

Cette étape permet de lire une valeur sensée être présente dans l'ADC à l'adresse de base 16 bits + 16_h

Etape n°1 : nombre identificateur = 7510(h) ? et initialisation de variables correspondantes à une prise de mesure simple ou multiple (annexe 7).

D'une part, la valeur d'identification est alors comparée à la valeur 7510_h. Pour que le programme continu, il faut donc réunir deux conditions : cette valeur doit être égale à 7510_h et il ne doit pas y avoir eu d'erreur sur le bus VME, sinon, le programme est stoppé (annexe 7).

D'autre part, on initialise deux variables avec des valeurs correspondantes à l'état attendu de la carte en fin de mesure. Deux cas sont possibles : un seul échantillonnage ou un ensemble d'échantillonnages.

Etape n°2 : remise à zéro de l'ADC (annexe 8).

Etape n°3 : pause (annexe 8).

L'étape 3 est une pause pour permettre à l'écriture du registre de l'ADC de s'effectuer sans problème.

Etape n°4 : configuration de l'ADC (annexe 9).

Cette séquence permet de réaliser plusieurs opérations :

La remise à zéro du pointeur de mémoire d'échantillonnage.

L'arrêt de l'échantillonnage lorsque la mémoire d'échantillonnage est remplie.

Le pointeur de mémoire n'est pas automatiquement remis à zéro lorsque l'échantillonnage est effectué.

Etape n°5 : pause (annexe 9).

De même que l'étape 3, cette séquence permet d'attendre que l'écriture vers l'ADC s'effectue.

Etape n°6 : paramétrage de l'horloge et de l'accès en mémoire (annexe 9).

Il s'agit du paramétrage de l'horloge : on sélectionne ici l'horloge interne sans aucune division de cette dernière. En effet, une horloge est nécessaire au bon fonctionnement de l'ADC pour lui permettre d'écrire dans ses propres registres.

Etape n°7 et 8 : détermine la quantité de mémoire à allouer aux mesures (annexe 10).

Ces deux étapes permettent de préciser à l'ADC la quantité de mémoire à allouer aux mesures qu'il doit effectuer.

Etape n°9 : interdit les interruptions (annexe 10).

Cette séquence permet d'empêcher les interruptions éventuelles.

Etape n°10 : pause (annexe 10).

L'étape 10 est une pause pour permettre à l'écriture du registre de l'ADC de s'effectuer sans problème.

Etape n°11 : lecture du statut de l'ADC (annexe 11).

Cette étape est destinée à lire la valeur du registre de statut de l'ADC.
Ainsi, on peut obtenir différents renseignements quant à la configuration de l'ADC. Le résultat est ensuite stocké dans une variable nommée *ReadValue*.

Etape n°12 : les deux bits de poids faible sont-ils égaux à 11(b) (annexe 11) ?

Si la valeur du statut correspond au statut attendu en fonction de la configuration qu'on lui a paramétré précédemment, alors on place le bouton VME Init à l'état faux. (annexe 11)
Sinon, On retourne une valeur d'erreur. (annexe 11)

➢ phase de fonctionnement.

Dans cette phase de fonctionnement, on retrouve plusieurs éléments de la phase d'initialisation. J'expliquerais donc la trame du programme en insistant sur les parties propres à l'acquisition.

Le programme commence par mettre en position fausse le bouton d'initialisation. Ensuite, on reprend les étapes n°1, 4 et 6 de la phase d'initialisation, à l'exception du fait que l'on sélectionne l'horloge externe pour n'effectuer qu'une seule et unique mesure.

Séquence n°2 : paramétrage de l'horloge et de l'accès en mémoire (annexe 12).

On sélectionne l'horloge externe et on autorise l'ADC à écrire dans sa mémoire d'échantillonnage. Cela signifie que seul l'ADC peut avoir accès à cette mémoire et non plus l'utilisateur (via le bus VME).

Séquence n°6 : acquisition des mesures (annexe 12).

Puisque l'on veut synchroniser la mesure, il est utile de déclencher le générateur d'impulsion, de manière à pouvoir ensuite déclencher une mesure sur l'ADC. Pour cela, nous utilisons le VI *IO reg pulse* décrit auparavant en plaçant la fréquence à 0 (une seule impulsion)

La mesure est alors déclenchée lorsque le bouton « stop » est à l'état vrai (le cas faux ne contient aucune opération). De plus, le VI *IO reg pulse* est inclus dans une boucle recommençant tant que le bouton « stop » est à l'état faux.

Autrement dit, le programme attend que l'utilisateur place le bouton «stop » à l'état vrai pour effectuer une seule et unique mesure, sinon, il attend que cette opération soit réalisée.

Séquence n°8 : paramétrage de l'horloge interne (annexe 12).

On sélectionne ensuite l'horloge interne de manière à pouvoir effectuer les différentes opérations nécessaires à l'ADC.

Séquence n°9 : comparaison du statut de l'ADC avec le statut attendu lorsque les mesures sont terminées (annexe 13).

Si la valeur lue est différente de celle attendue, on recommence la lecture jusqu'à ce que la valeur soit satisfaisante. Cette comparaison permet d'attendre jusqu'à ce que l'ADC est effectué toutes les conversions nécessaires.

Séquence n°10 : lecture de la valeur du pointeur mémoire (annexe 13).

Cette lecture permet de déterminer le nombre de mesures à lire. La lecture s'effectue en deux parties : la première consiste à lire les quatre bits de poids fort, la seconde permet de lire les seize bits de poids faible. On place ensuite cette valeur dans une variable nommée *Size*.

Séquence n°12 : fixe la taille maximale de l'échantillonnage à 100000 et autorise l'accès en mémoire pour l'utilisateur (annexe 13).

Séquence n°13 : détermine la zone de mémoire à lire (annexe 14).

On détermine alors la zone de mémoire à lire en fonction de la demande de l'utilisateur, c'est à dire selon la voie de l'ADC qu'il a choisi. En effet, chaque échantillonnage est enregistré à une place différente en mémoire selon la voie de ce dernier.

Séquence n°14 : lecture de la mémoire d'échantillonnage, trace le graphique et renvoie les résultats des mesures (annexe 14).

Séquence n°15 : place l'ADC dans un état adéquat pour acquérir de nouvelles valeurs (annexe 14).

3. Contrôle de la carte de multiplexage.

Le but de ce VI est de permettre d'automatiser le banc de test. Il doit donc permettre de mesurer l'ensemble des 96 canaux grâce à la carte de multiplexage.

Ainsi, la conception de ce VI est étroitement liée à la façon dont a été réalisée la carte de multiplexage.

Cette dernière utilise le port 9 du DIO. De plus, le câblage de la carte a été réalisé de telle sorte que chaque relais possède un numéro correspondant au canal auquel il est relié. Par exemple, si l'on envoie la valeur 12 vers le port 9 du DIO, la carte fermera alors le relais numéro 12.

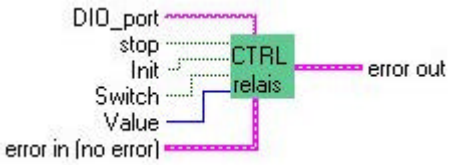
Ainsi, sa programmation devient simple. En effet, pour passer d'une voie à la suivante, il suffit d'incrémenter une variable et de l'envoyer vers le DIO.

En ce qui concerne l'envoi des canaux huit par huit vers l'ADC, il suffit simplement de décaler de 3 bits vers la gauche la valeur à envoyer (cela revient à multiplier le nombre binaire par 8).

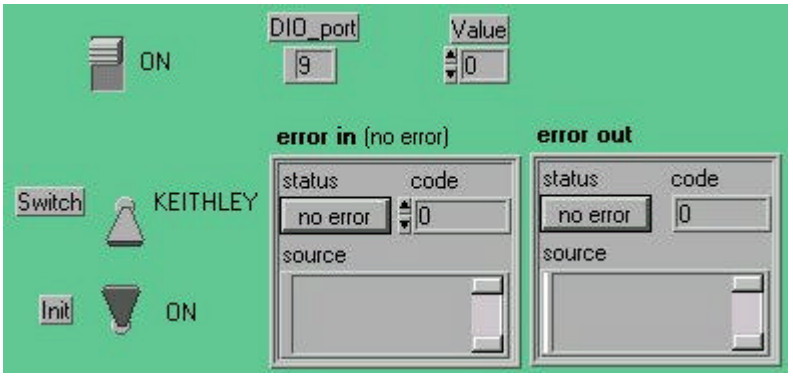
Il s'agit alors de réaliser un VI qui pourra être appelé par le programme principal. Nous avons donc besoin de pouvoir maîtriser certains paramètres :

- Préciser le numéro du port DIO considéré.
- Une initialisation possible à partir du programme principal.
- Déterminer si la mesure s'effectue par le micro-ampèremètre ou par l'ADC.
- Permettre d'effectuer une pause de ce VI lorsqu'il est en cours de fonctionnement, cela permet de l'utiliser plusieurs fois dans un programme tout en le gardant dans la même configuration.
- Pouvoir envoyer le numéro du canal à tester.
- Pouvoir récupérer un éventuel code d'erreur.
- Empêcher la mise en route du programme si une erreur s'est produite auparavant.

Voici donc le VI *CtrlDIO* et ses connecteurs :



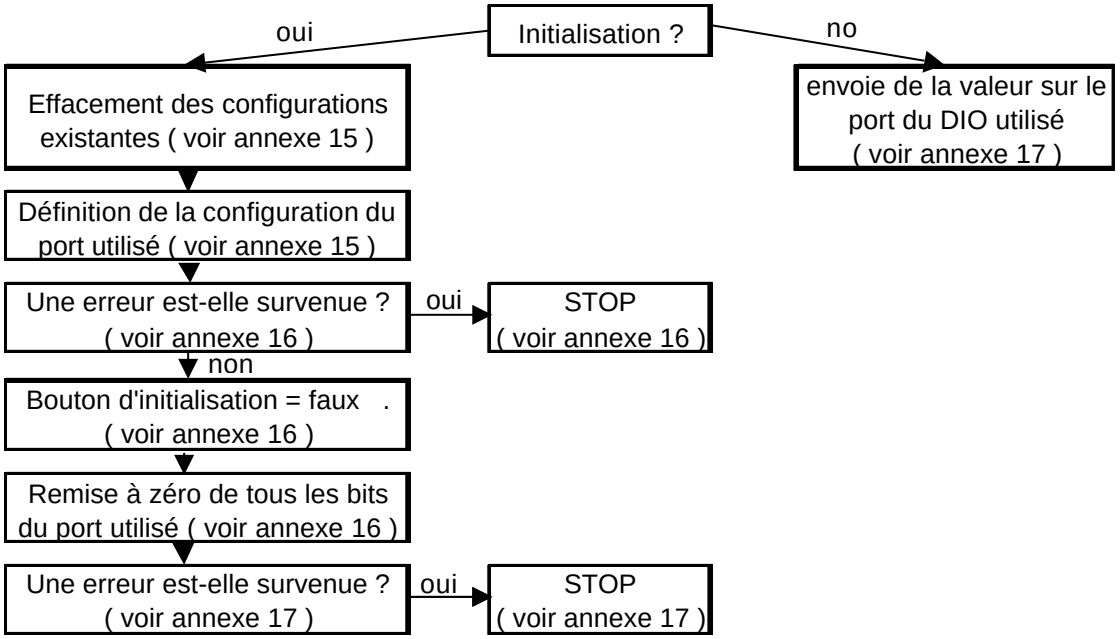
Ainsi que sa face avant :



Le bouton *switch* permet de choisir si l'on envoie les signaux vers le micro-ampèremètre (KEITHLEY) ou bien vers l'ADC.

Comme les programmes présentés auparavant, il comporte deux modes de fonctionnement : l'initialisation et l'envoi des données.

L'organigramme du programme est le suivant :



➢ Phase d'initialisation :

Elle comporte en tout 4 séquences :

Séquence n°0 : effacement des configurations existantes (annexe 15).

Cette séquence permet d'effacer toutes les configurations existantes sur le port DIO considéré par le numéro DIO_port.

Séquence n°1 : définition de la configuration du port utilisé (annexe 15).

Cette phase a pour but de construire une configuration sur le port considéré du DIO. C'est à dire ici : le port numéro *DIO_port* sera un port de 8 bits, tous consacrés à l'écriture (grâce à la constante FF).

La configuration renvoie alors une valeur d'identification de ce port (port trig id) qui devra être désormais utilisée pour l'émission de signaux à partir de ce port.

Séquence n°2 : une erreur est-elle survenue (annexe 16) ?

Cette séquence est un peu particulière car elle permet, si une erreur est survenue, d'afficher un message d'erreur sur l'écran comportant le message :

« UNABLE CONFIGURE DIO PORT FOR TRIGGER (#9) ».

Si aucune erreur n'apparue, alors, le bouton d'initialisation passe à l'état faux (annexe 16).

Séquence n°3 : remise à zéro de tous les bits du port utilisé (annexe 16).

Ici, on remet les huit bits du port DIO utilisé à zéro.

Séquence n°4 : une erreur est-elle survenue (annexe 17) ?

Cette dernière séquence d'initialisation permet de stopper le programme si une erreur est survenue lors de la remise à zéro du port.

➢ Phase d'envoi des données : envoie de la valeur sur le port du DIO utilisé.

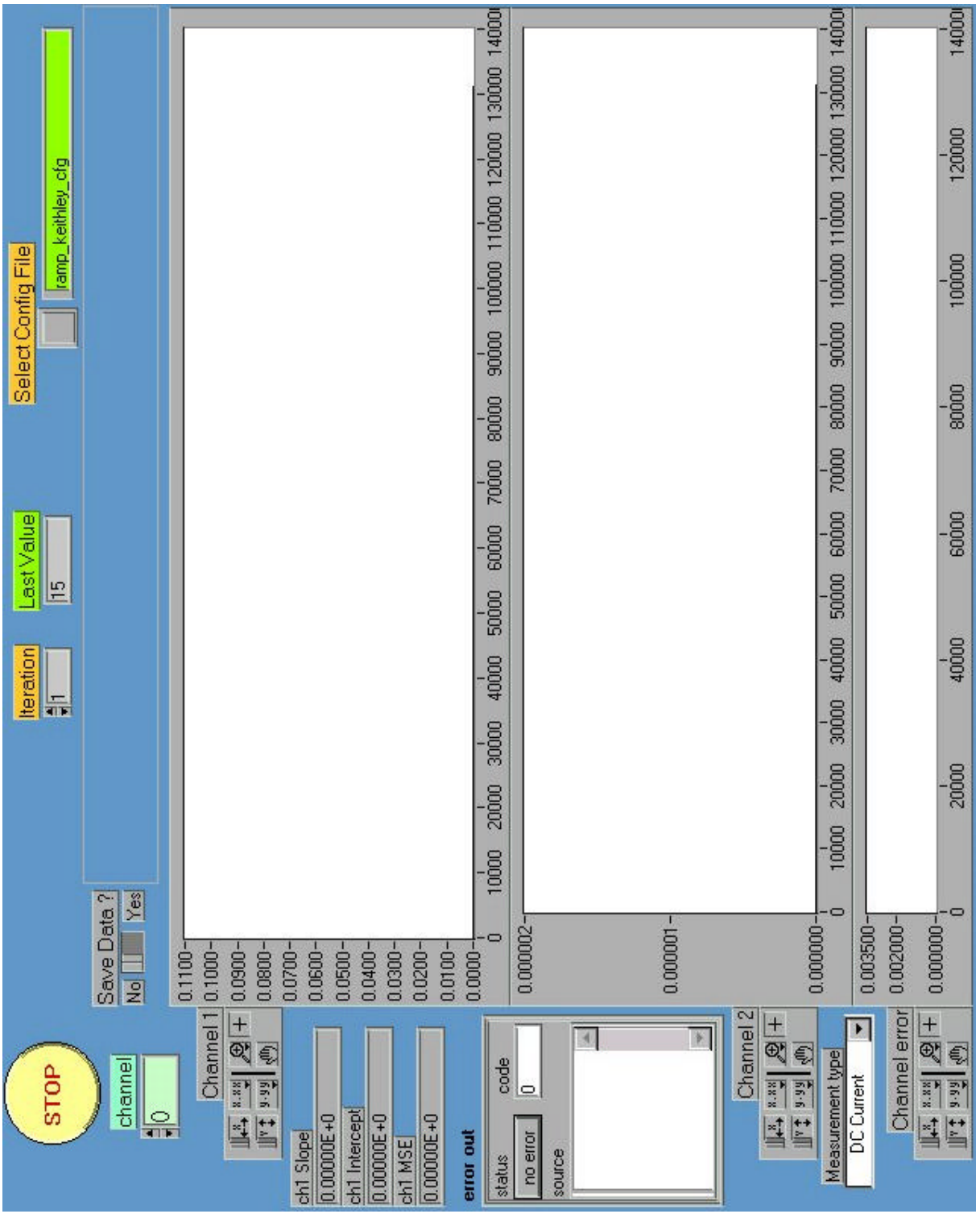
Cette phase ne comporte qu'une seule séquence qui permet d'envoyer une valeur vers le port du DIO utilisé. Cette dernière permet notamment de gérer indépendamment le bit de poids fort pour contrôler l'envoi des données vers le bon instrument de mesure (avec le décalage de bits nécessaire).

IV. Réalisation de l'interface du banc de test

Il s'agit de l'interface utilisateur qui va permettre d'effectuer l'ensemble des tests.

1. Interface pour la linéarité en courant.

Cette interface est la suivante :

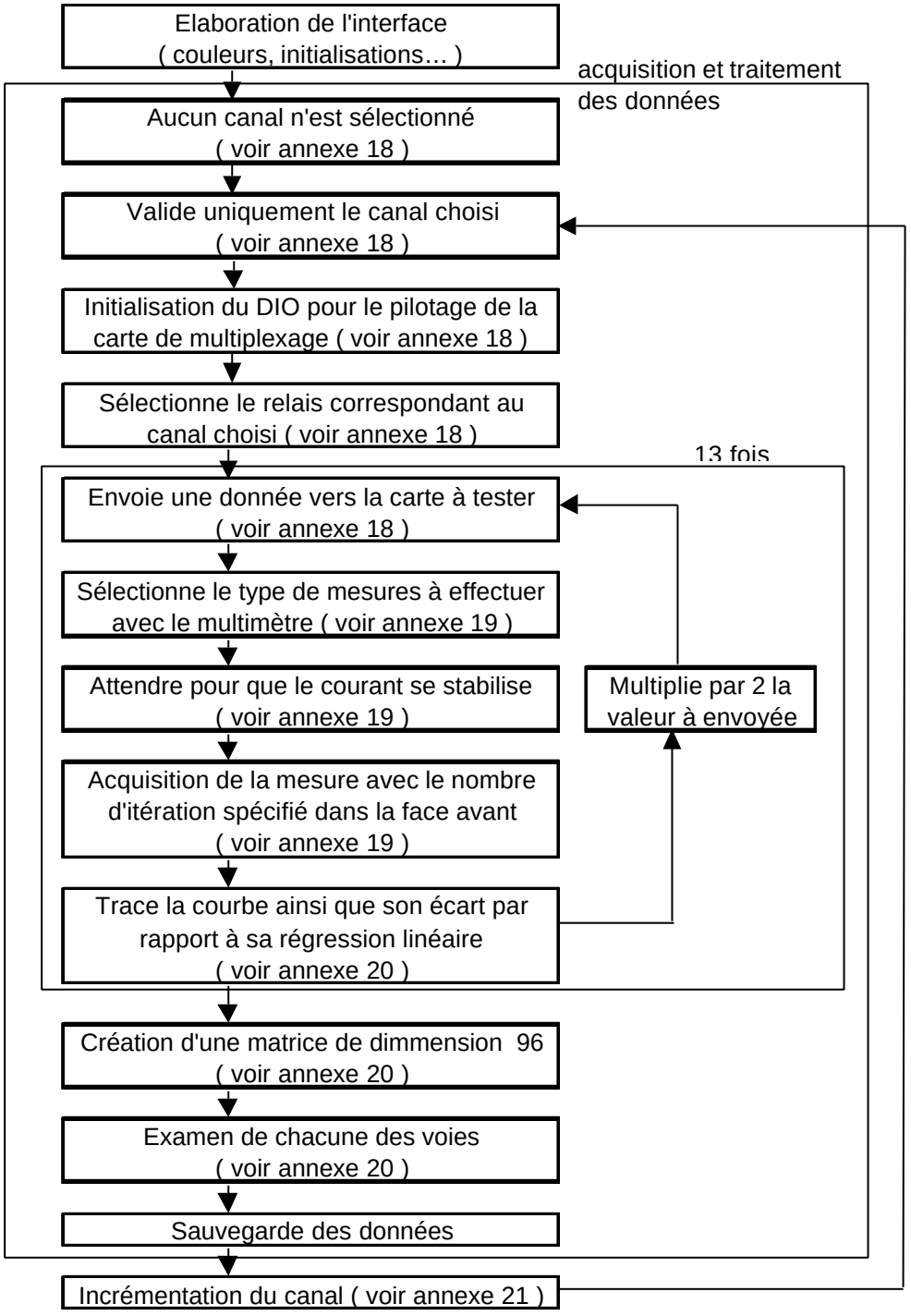


Description de cette interface (de gauche à droite et de haut en bas) :
Bouton stop : permet de lancer les mesures.
Itération : il s'agit du nombre de mesures effectuées pour chaque valeur envoyée vers la carte.
Last Value : indique à l'utilisateur la dernière valeur envoyée vers la carte.
Select config file : permet de sélectionner un fichier de configuration pour la carte, à savoir les voies que l'on déclenche, les retards qu'on leur impose…
Chanel : permet de visualiser le canal que l'on est en train de tester.
Save Data : cette option est dédiée à la sauvegarde des données, c'est à dire les points des courbes, les pentes…
Le cadran « channel 1 » : il s'agit de la courbe de réponse à la rampe. En abscisse, nous voyons les valeurs envoyées vers la carte, et en ordonnée, la valeur du courant émis par cette dernière.
CH1 Slope : représente la pente de la régression linéaire effectuée sur la courbe de réponse à la rampe.
CH1 Intercept : il s'agit de l'ordonnée à l'origine.
CH1 MSE (Mean Square Root) : correspond à l'écart type de la courbe par rapport à sa régression linéaire.
Error out : permet de visualiser une éventuelle erreur survenue lors de l'exécution du programme.
Le cadran « channel 2 » : permet de visualiser l'écart (en ampère) entre les résultats des mesures et la régression linéaire.
Measurement type : Il s'agit du type de mesure que l'on veut effectuer avec le multimètre. Par défaut, ce dernier sera considéré comme un micro-ampèremètre.
Le cadran « channel error » : permet de visualiser la valeur du courant en sortie des autres voies lorsqu'elles sont sensées être désactivées. Cela permet de vérifier s'il y a des interférences entre les voies.

Ce programme a été en grande partie réalisé par Jean-François Huppert. Toutefois, j'ai été amené à effectuer différents changements dans son mode de fonctionnement, notamment pour les mesures. En effet, ce programme avait été réalisé pour effectuer des mesures avec d'autres appareils. J'ai donc modifié une partie du programme pour permettre d'utiliser les nouveaux instruments.

Pour les parties que je n'est pas réalisées, j'expliquerais le principe de fonctionnement du programme. En revanche, je détaillerais les modifications apportées.

Le diagramme du programme est le suivant :



La première partie du programme consiste à élaborer l'interface et à charger les différentes configurations nécessaires au bon fonctionnement du programme. Ensuite viennent les phases de tests qui permettent de déceler la présence d'une éventuelle erreur. Puis vient la partie des mesures.

Cette dernière partie se décompose en huit séquences qui permettent de réaliser l'ensemble du test de linéarité en courant. Ces séquences sont inclues dans une boucle ayant pour conditions de sortie soit l'action du bouton stop, soit un numéro de canal supérieur à 95 (le premier canal ayant pour numéro : 0).

Séquence 0 : aucun canal n'est validé (annexe 18).

Il s'agit ici de l'étape qui permet d'« éteindre » l'ensemble des canaux. On pourra donc par la suite les allumer les uns après les autres. Le VI « 96 CH setEn » a été réalisé et modifié par Jean-François Huppert pour pouvoir l'utiliser de cette façon.

Séquence 1 : valide uniquement le canal choisi (annexe 18).

Séquence 2 : initialisation du DIO pour le pilotage de la carte de multiplexage (annexe 18).

Séquence 3 : sélectionne le relais correspondant au canal choisi (annexe 18).

Séquence 4 :

Cette séquence se décompose en trois parties.

➢ La première ci-dessus permet d'effectuer différentes opérations :

- Envoyer une donnée vers la carte à tester (c'est le VI « 96 CH setdac » qui permet l'envoi de la donnée) (annexe 18).
- Sélectionner le type de mesures à effectuer avec le multimètre (annexe 19).
- Attendre que le courant se soit stabilisé pour ensuite réaliser les mesures (annexe 19).
- Prendre les mesures avec le nombre d'itérations spécifié dans la face avant (annexe 19).

➢ La seconde partie permet de tracer la courbe de réponse à la rampe ainsi que son écart par rapport à sa régression linéaire (annexe 20).

➢ La troisième partie consiste à sauvegarder les différentes données sous forme de fichiers.

Cette troisième partie a été réalisée par Jean-François Huppert.

Séquence 5 : création d'une matrice de dimension 96 (annexe 20).

Séquence 6 : examen de chacune des voies (annexe 20).

Cette séquence est dédiée à l'examen de chacune des voies excepté celle en cours de test. Les cas qui précèdent celui ci-dessous ne sont autres que la fermeture

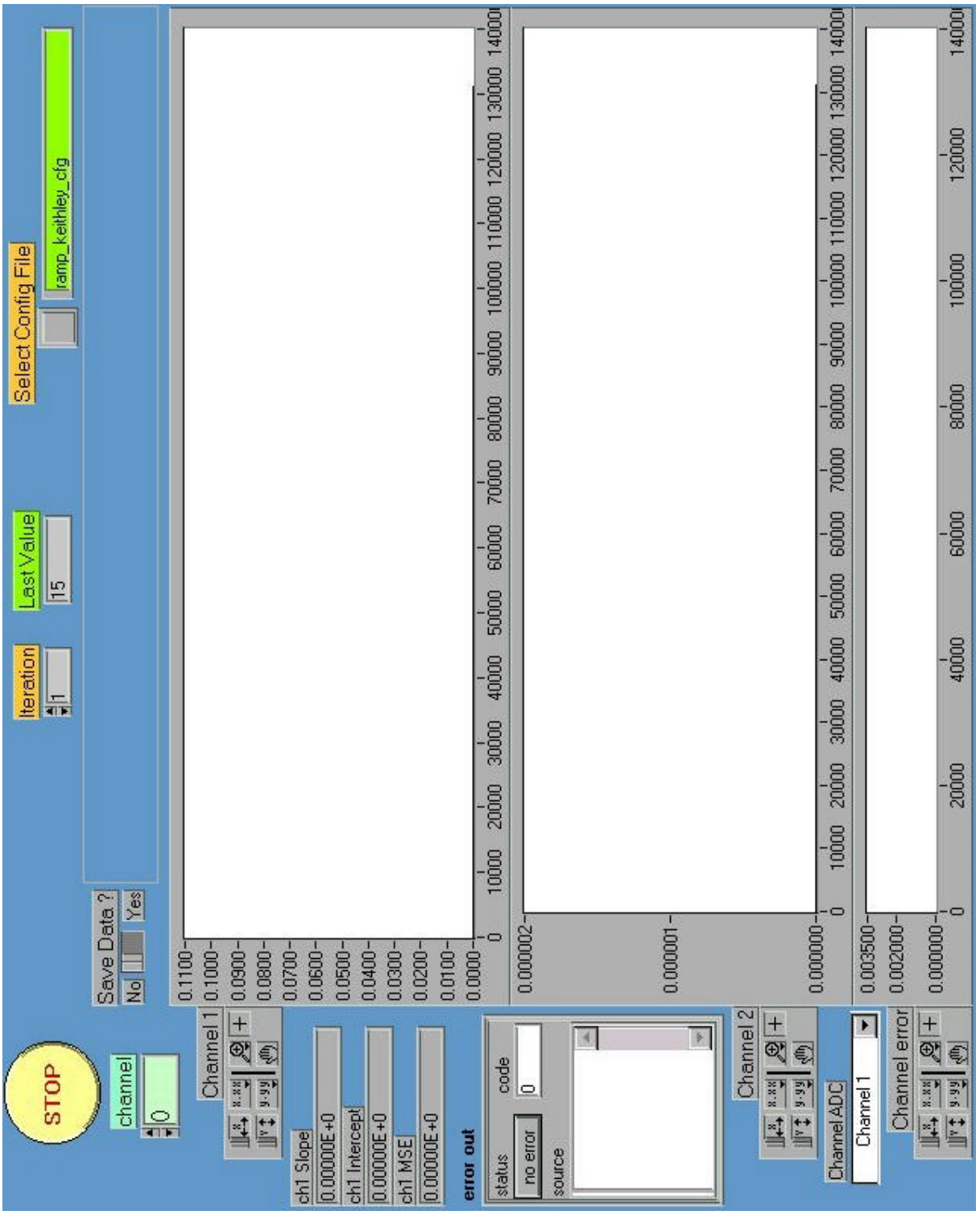
successive des différents relais, une phase d'attente, puis le mécanisme qui permet de recueillir la valeur du courant. Le programme passe donc en revue chacune des voies et mesure ensuite le courant émis. Le résultat est ensuite envoyé dans le cadran Chanel error.

Séquence 7 : incrémentation du numéro du canal jusqu'à 95 (annexe 21).

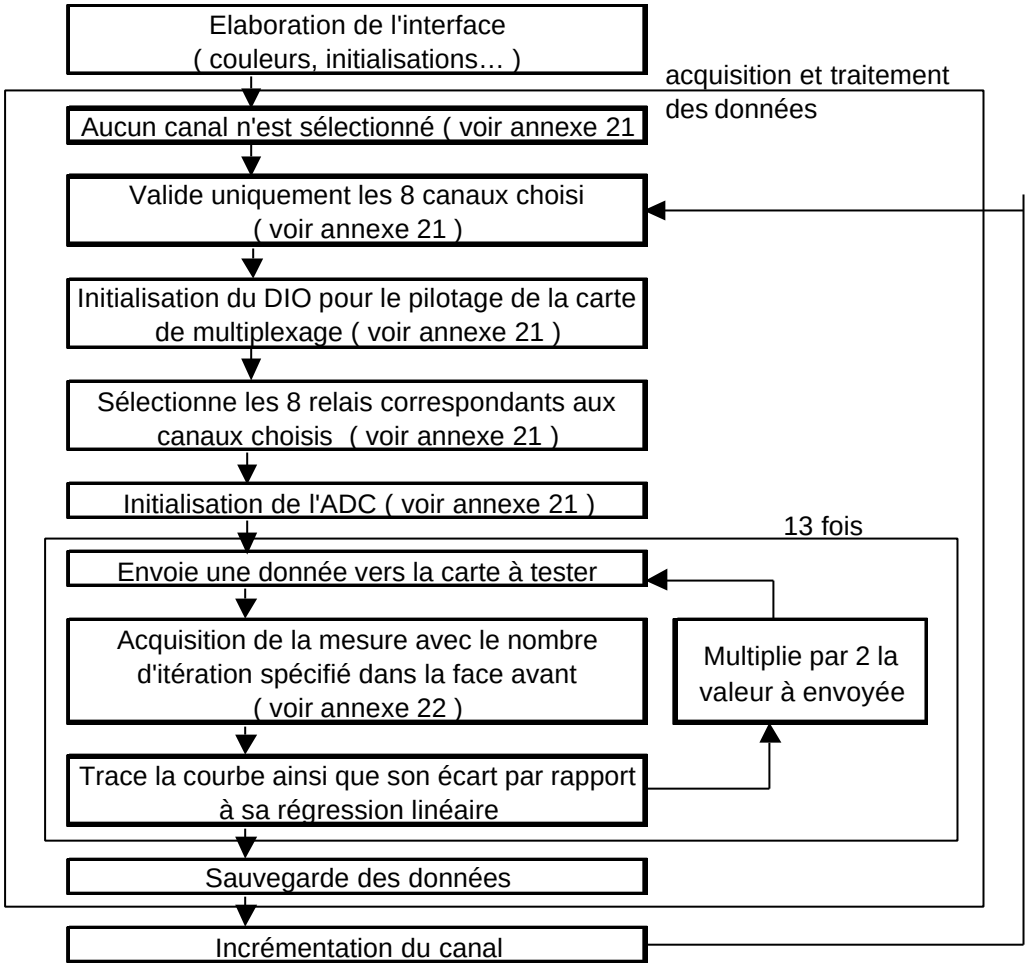
2. Mesure de l'amplitude des impulsions en sortie du *fanout*.

Cette mesure est réalisée grâce à la carte ADC pilotée par VME.

Cette interface ayant pour but de mesurer la linéarité de l'amplitude des impulsions, il est donc normal de reprendre dans sa grande majorité le VI concernant la mesure de linéarité en courant. Le seul changement vient du fait que la mesure s'effectue sur l'ADC et non sur le micro-ampèremètre. On utilisera donc le VI permettant le contrôle de l'ADC du bus VME (Read-VME-ADC), présenté auparavant.



Le diagramme de ce programme est sensiblement identique au précédent :



Pour ce test, seuls les cas expliqués auparavant sont modifiés. Il n'y a donc plus que six séquences et non huit. En effet, le test permettant de déceler s'il n'y a pas d'interférences n'est plus utile.

Voici donc l'explication des différents changements opérés.

Séquence 0 et 1 : sélection des voies huit par huit (annexe 21).

Ici, les changements viennent du fait qu'il est maintenant nécessaire de sélectionner les voies non plus une par une mais huit par huit. Il s'agit ici de la constante booléenne (en haut à droite) qui permet d'opérer le changement dans la sélection des voies.

Séquence 2 et 3 : sélection des relais huit par huit (annexe 21).

De même ici, la seule différence vient de la sélection des relais huit par huit, grâce à la constante booléenne la plus basse.

Séquence 4 : initialisation de l'ADC (annexe 21).

Séquence 5 : acquisition des mesures (annexe 22).

Ici, seule la partie effectuant la mesure a été changée. En effet, on a maintenant recours au VI Read VME ADC. Ce dernier effectue ici une seule mesure à chaque fois, mais recommence cette même mesure le nombre de fois indiqué par « Itération ».

Le reste de l'interface est identique à celle permettant de mesurer la linéarité en courant, excepté le fait que l'on ne teste plus les voies inactives.

REALISATION

DES ESSAIS

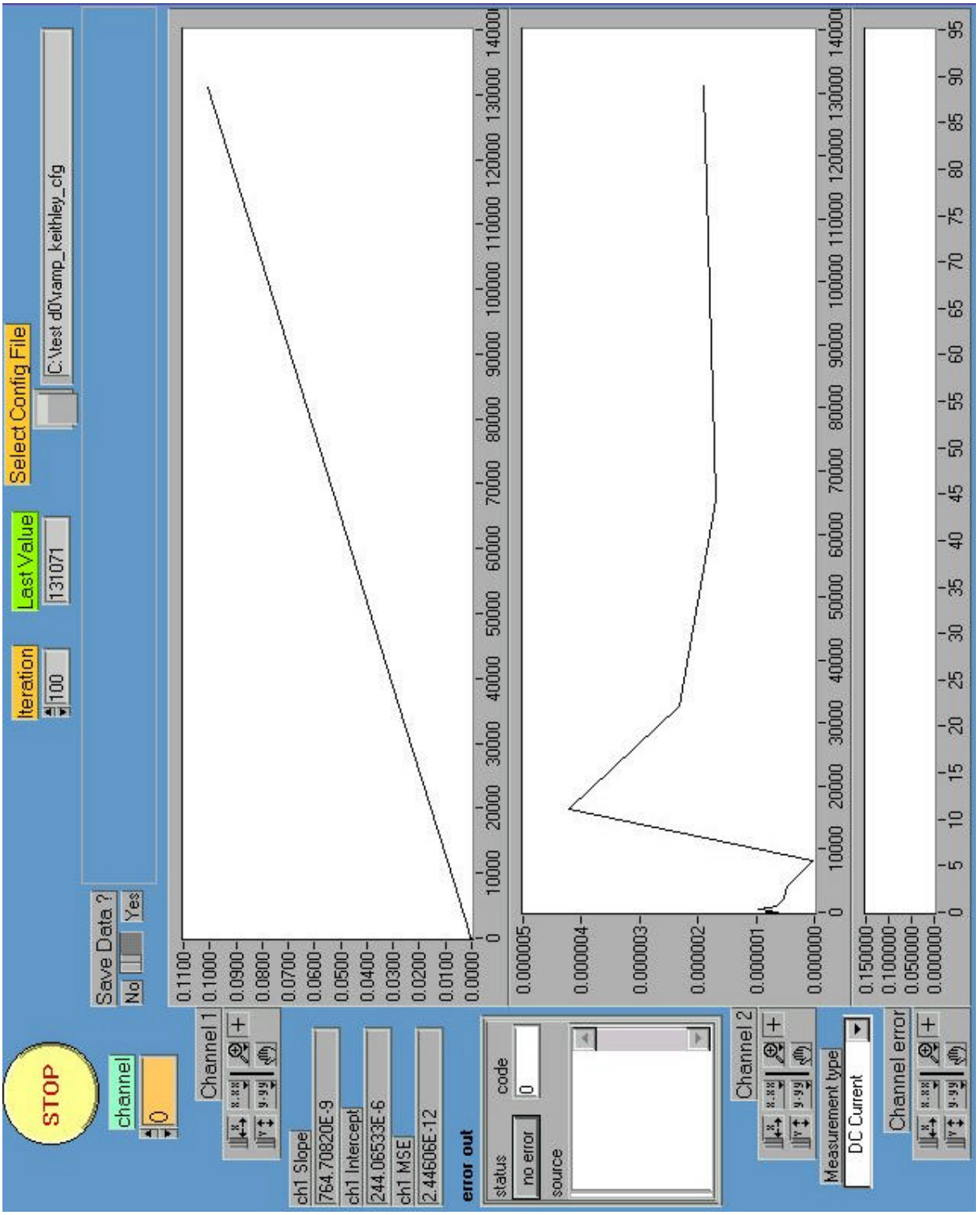
ET

RESULTATS

Le prototype que nous avons à notre disposition pour vérifier le bon fonctionnement du banc de test ne comporte que seize voies.

D'autre part, la carte de multiplexage est en cours de fabrication et donc, il nous est impossible de réaliser l'examen de chacune des voies non validées.

Pour l'instant, seul le test de linéarité en courant est disponible.



Le résultat précédent montre une linéarité très précise pour la voie 0. En effet, l'écart type est de $2{,}5.10^{-12}$ A.

Résultats des autres voies :

voie	Pente (A / unité de DAC)	Ordonnée à l'origine (A)	Ecart type
1	7,6472816E-07	2,3535101E-04	4,1687800E-12
2	7,6486115E-07	2,3672064E-04	3,2203700E-12
3	7,6465697E-07	2,3656484E-04	3,6117100E-12
4	7,6483023E-07	2,3866320E-04	1,3880000E-12
5	7,6468458E-07	2,3789694E-04	3,5829156E-13
6	7,6483412E-07	2,3871217E-04	3,7239100E-12
7	7,6463374E-07	7,6463374E-07	3,4602870E-13
8	7,6481205E-07	2,4125983E-04	2,2629400E-12
9	7,6473514E-07	2,3364280E-04	2,5240900E-12
10	7,6485686E-07	2,3970963E-04	1,9154800E-12
11	7,6468472E-07	2,3661468E-04	3,3243600E-12
12	7,6493166E-07	2,4022797E-04	1,2405200E-12
13	7,6455109E-07	2,3320046E-04	3,1159617E-13
14	7,6484312E-07	2,3699083E-04	4,7868100E-12
15	7,6472493E-07	2,3918315E-04	3,0824200E-12

Chaque courbe de linéarité est réalisée avec cent itérations par mesures.

Pour ces 16 premières voies, on peut connaître les renseignements suivants :

Pente moyenne $\approx 7{,}65.10^{-7}$ ($\sigma = 1{,}04702.10^{-10}$)

Ordonnée à l'origine moyenne $\approx 2{,}22.10^{-4}$ ($\sigma = 6{,}12.10^{-5}$)

Ecart type moyen $\approx 2{,}42.10^{-12}$ ($\sigma = 1{,}45.10^{-12}$)

CONCLUSION TECHNIQUE

Ce stage a été une première expérience professionnelle très enrichissante sur tous les plans : aussi d'un point de vue de l'approfondissement de mes connaissances en électronique et informatique que du point de vue relationnel.

J'ai donc eu le loisir d'apprendre un nouveau langage de programmation, le langage G, à travers LABVIEW.

Une fois le banc de test achevé, le personnel du service électronique pourra effectuer l'ensemble des test pour vérifier le bon fonctionnement des cartes *contrôle-mère* ainsi que des cartes *active fanouts*.

Cependant, des modifications pourront être apportées.

Des tests pourront être ajoutés au programme existant ce qui rendra plus complète l'étude de la carte.

Enfin, le logiciel de banc de test ainsi que la carte de multiplexage vont être installés sur le site de Fermilab aux Etats-Unis de manière à pourvoir renouveler des test sur les cartes déjà en place.

CONCLUSION GENERALE

Au terme de ce stage, j'ai eu la satisfaction d'avoir réalisé une partie d'un banc de test. Mais plus que cette satisfaction, j'ai eu le plaisir de travailler dans un domaine qui me passionne.

En effet, ce stage m'a permis non seulement d'approfondir mes connaissances en électronique et informatique mais aussi d'acquérir une expérience extrêmement valorisante d'un point de vue personnel.

Dans la mesure où il reflète parfaitement le domaine dans lequel j'aimerais poursuivre mes études, j'estime être heureux d'avoir pu effectuer ce stage entouré de personnes compétentes qui ont su me guider dans mes démarches tout en me laissant une certaine autonomie.

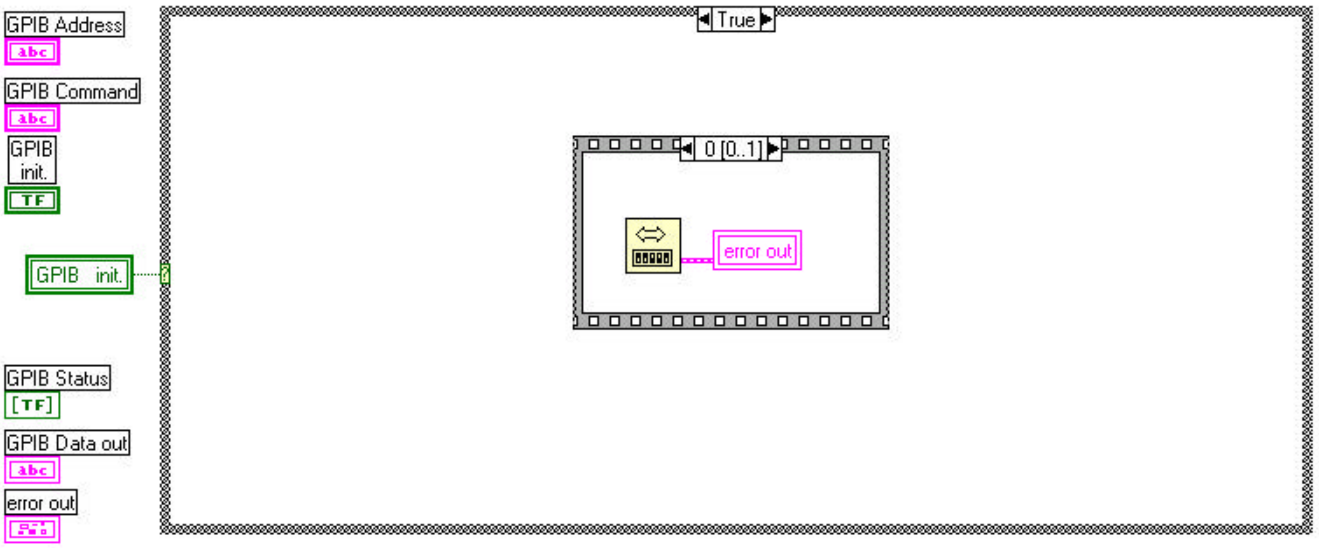
ANNEXES

annexesRéalisation du VI permettant de contrôler le GPIB.

Le diagramme correspondant se décompose en deux parties : l'initialisation ou le fonctionnement normal.

- Phase d'initialisation.

Séquence n°0 : initialisation de l'interface GPIB.

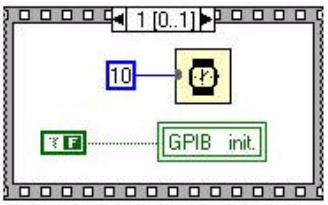


On remarque, alignés en colonne à gauche, les terminaux présents sur la face avant : on les place comme ceci car ces derniers ne sont pas duplicables. En effet, si l'on duplique un terminal, le *control* associé présent sur la face avant est lui aussi dupliqué. Pour remédier à cela, on utilise des variables locales. Ces dernières sont des liens vers leur terminal respectif. La seule différence est que l'on peut les copier indéfiniment. Une variable locale peut être soit accessible en lecture, soit en écriture.

Ici, nous pouvons observer la présence d'une variable locale correspondant au terminal *error out* accessible en écriture. Ainsi, il nous est possible de faire à nouveau appel à cette variable pour d'autres utilités : par exemple, la réinjecter dans un autre VI en tant que *error in*. Elle sera alors accessible en lecture.

Cette séquence permet d'initialiser l'ensemble de l'interface GPIB et de renvoyer une éventuelle erreur.

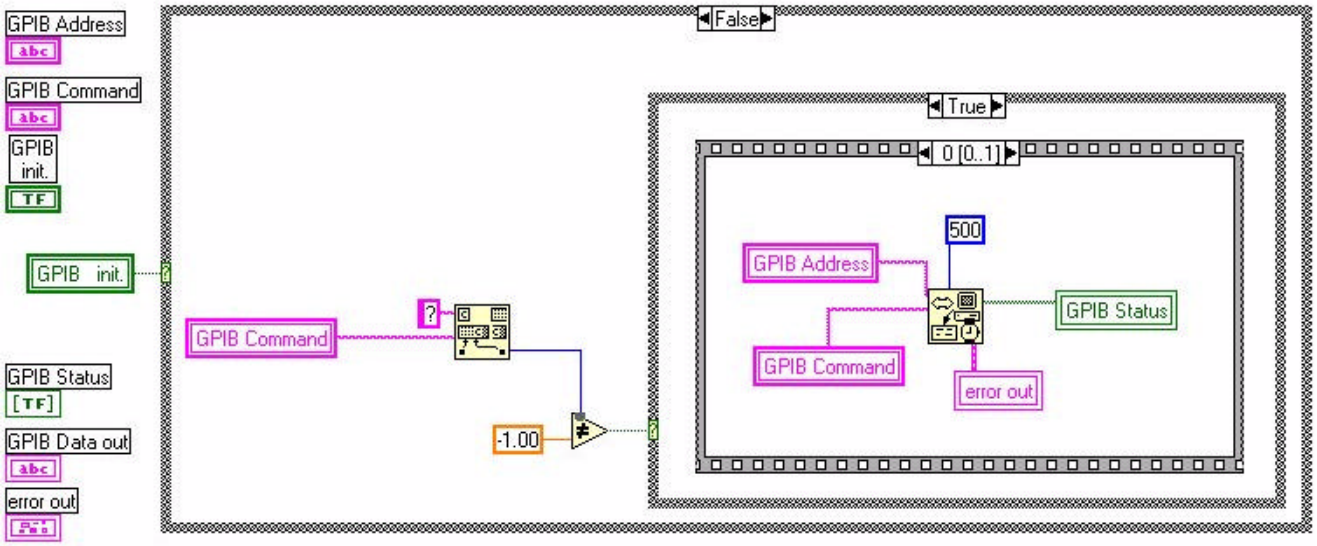
Séquence n°1 : attente de 10 ms et Initialisation = faux



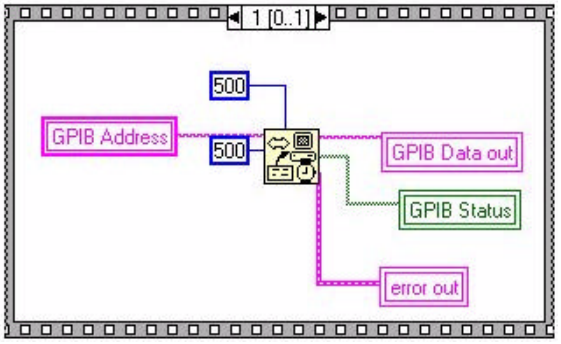
➢ Phase de fonctionnement : la commande est-elle une instruction ou une requête ?

Cas d'une requête : la commande contient le caractère « ? ».

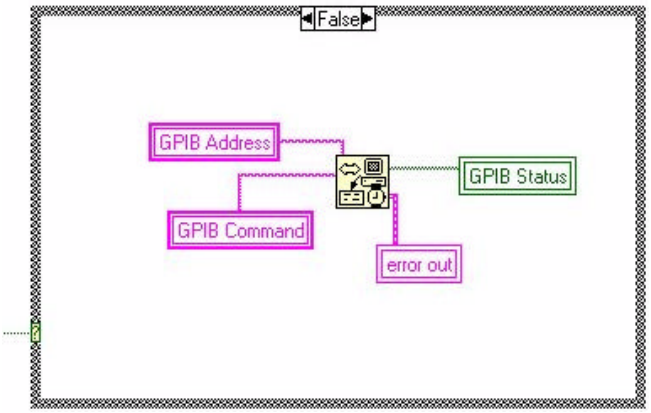
Séquence n°0 : envoi de la requête.



Séquence n°1 : récupération des données.



Cas d'une instruction : la commande ne contient pas le caractère « ? ».

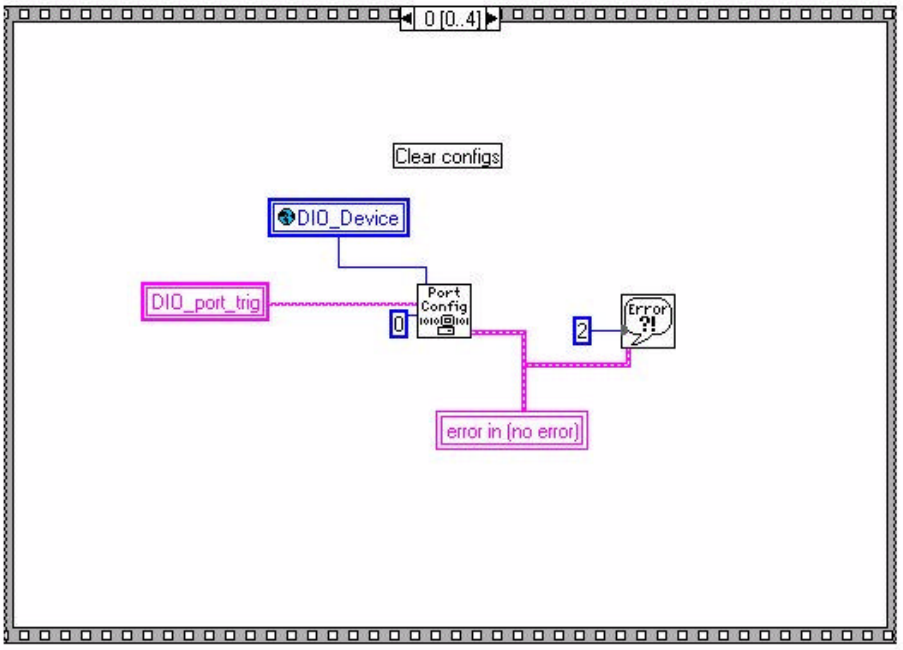


Réalisation du VI permettant de contrôler l'ADC du bus VME.

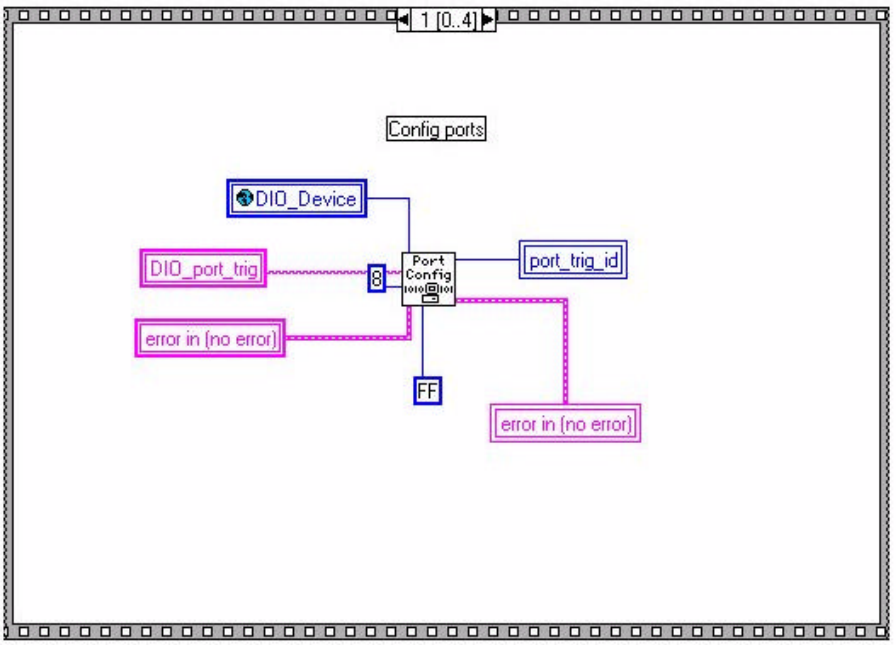
➢ Synchronisation de la carte *contrôle mère* et de l'ADC

➢ phase d'initialisation

Séquence n°0 : effacement des configurations existantes.

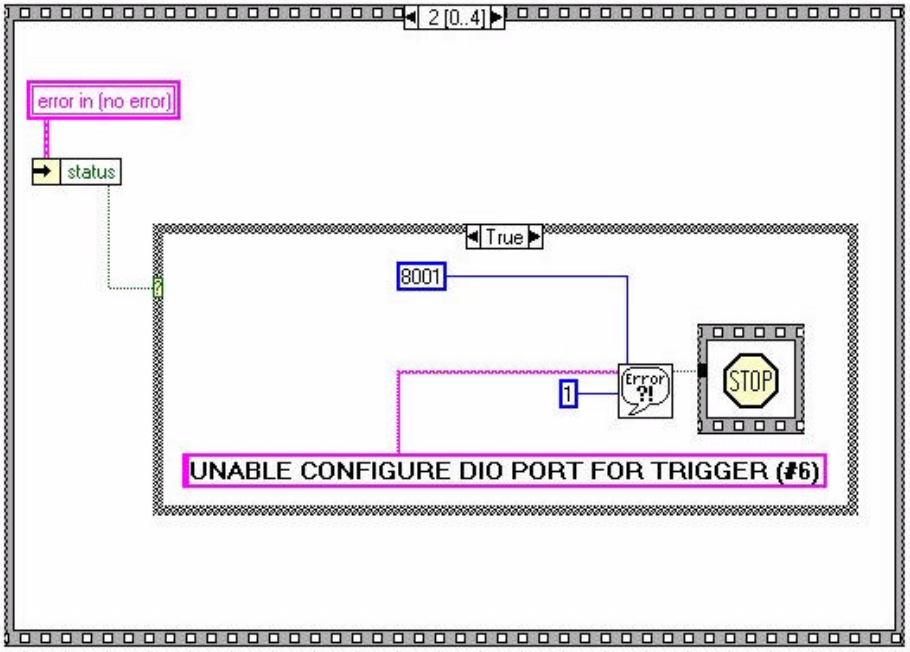


Séquence n°1 : définition de la configuration du port utilisé.

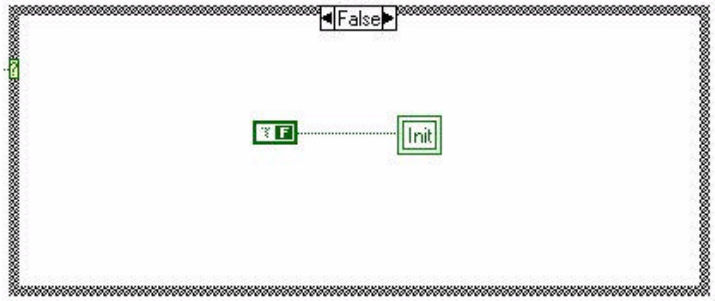


Séquence n°2 : une erreur est-elle survenue ?

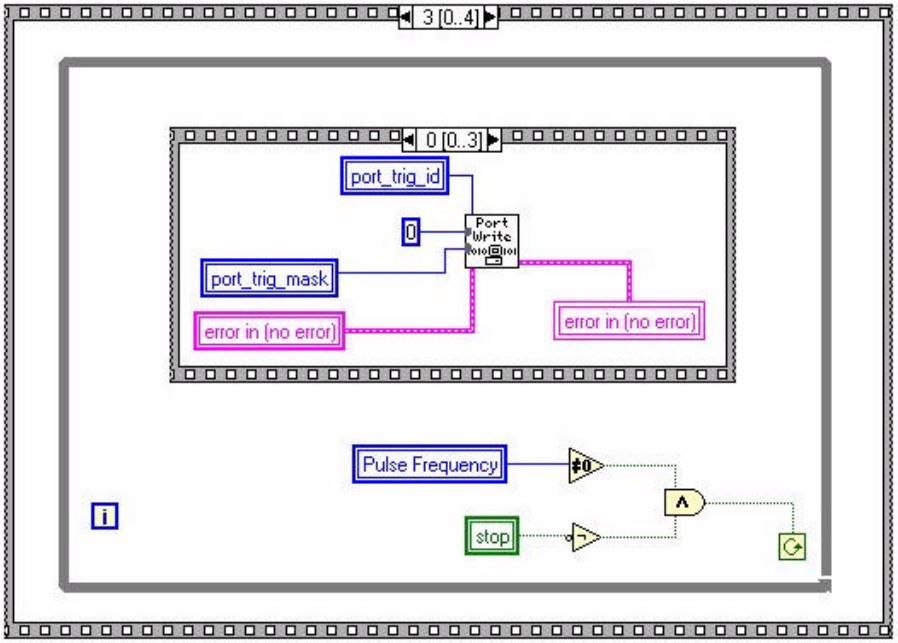
Si une erreur est survenue :



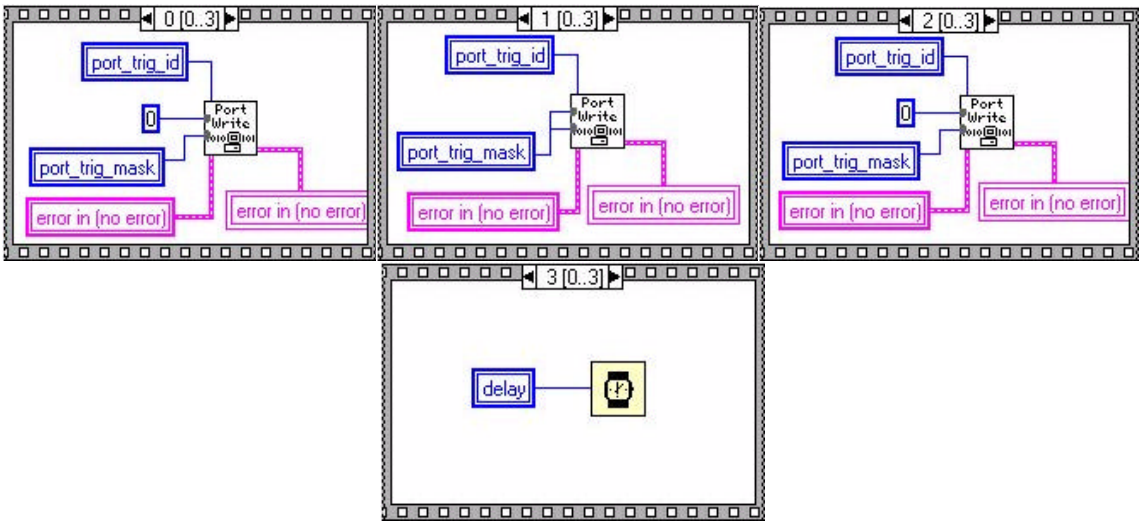
Si aucune erreur n'est survenue :



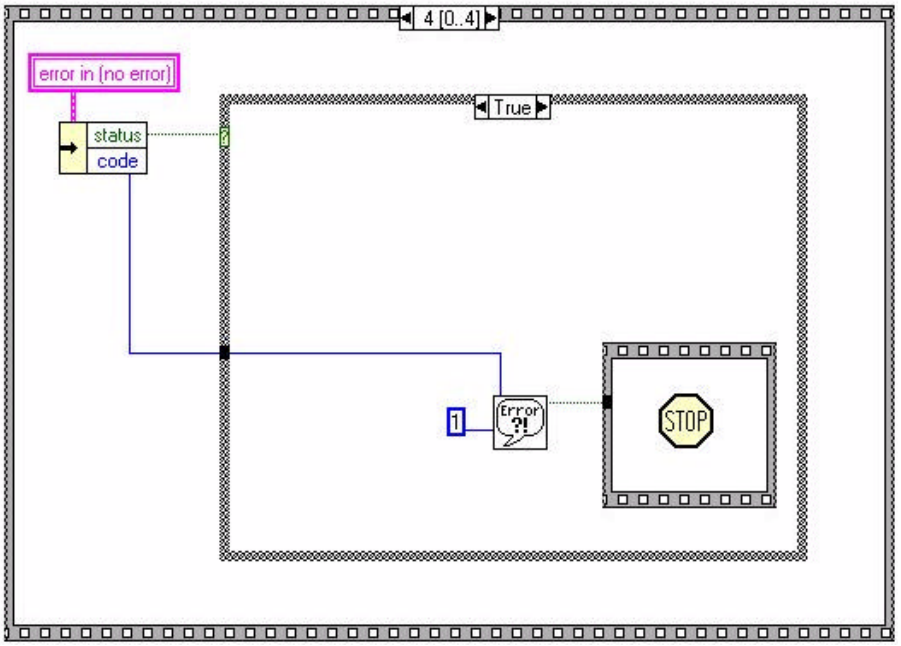
Séquence n°3 : émission d'une impulsion, attente = délai puis test si le bouton Stop = faux ET si "pulse frequency" ≠ 0 ? (annexe 5)



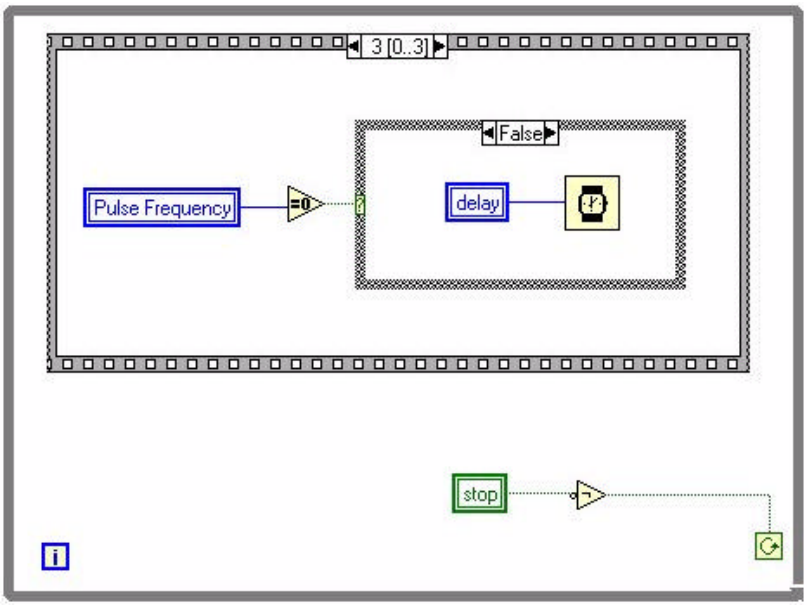
Séquence pour l'émission de l'impulsion :

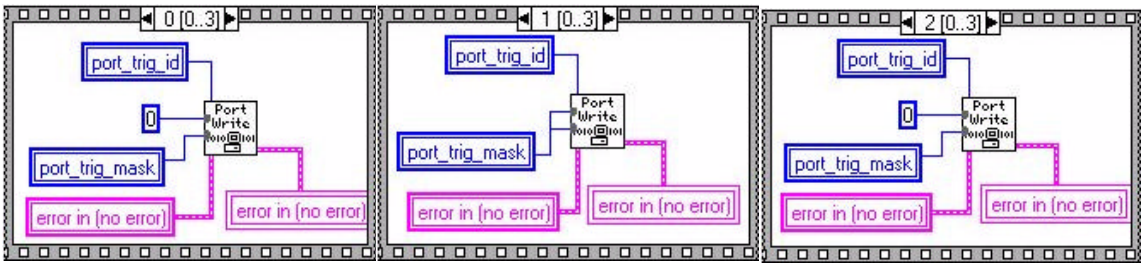


Séquence n°4 : une erreur est-elle survenue ?



- Phase de fonctionnement : émission d'une impulsion, attente = délai puis test l'état du bouton Stop (annexe 6).

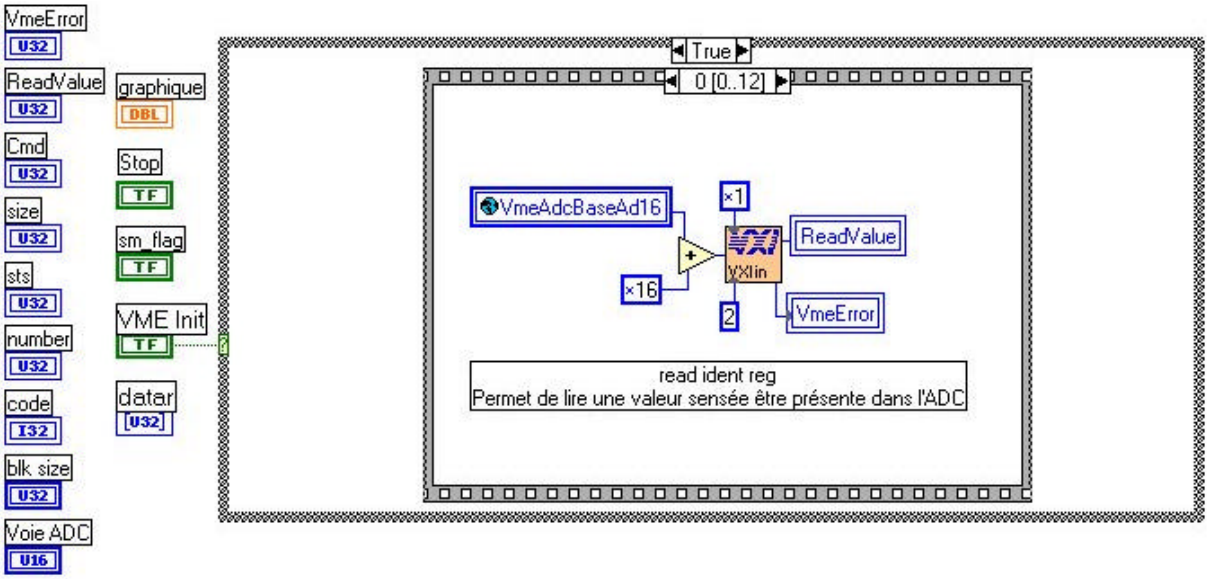




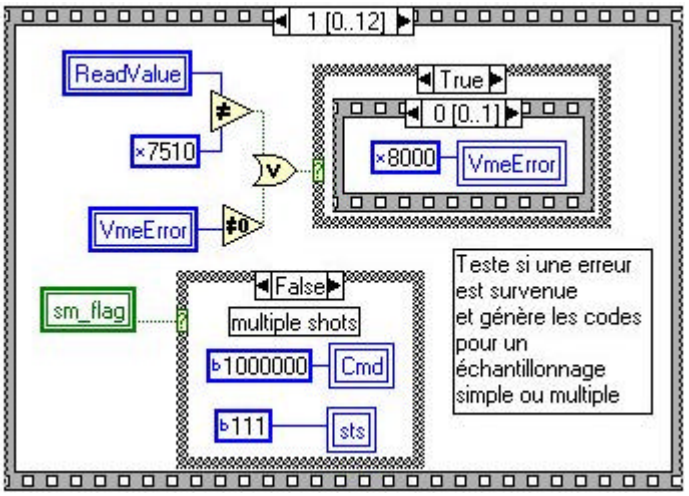
➢ réalisation de l'interface permettant la lecture de l'ADC :

➢ phase d'initialisation.

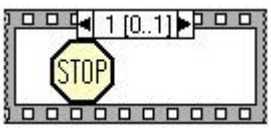
Etape n°0 : test d'identification.



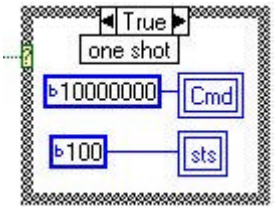
Etape n°1 : nombre identificateur = 7510(h) ? et initialisation de variables correspondantes à une prise de mesure simple ou multiple.



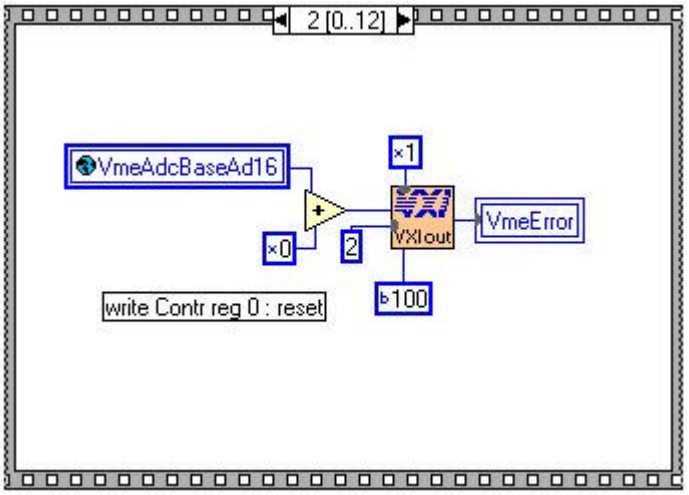
Cas où la valeur lue est différente de la valeur d'identification de la carte (7510_h.) ou s'il y a une erreur sur le bus VME :



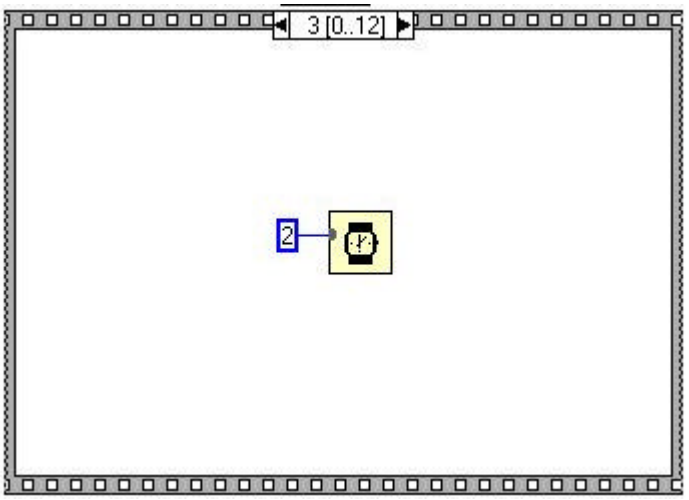
Si aucune erreur n'est survenue : initialisation de variables correspondantes à une prise de mesure simple ou multiple:



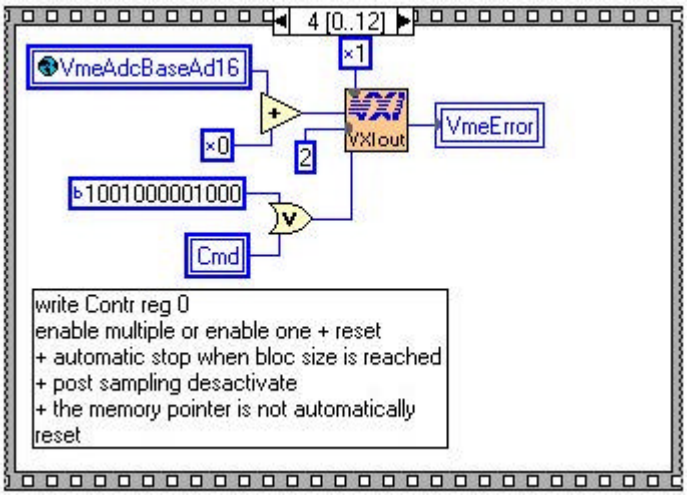
Etape n°2 : remise à zéro de l'ADC.



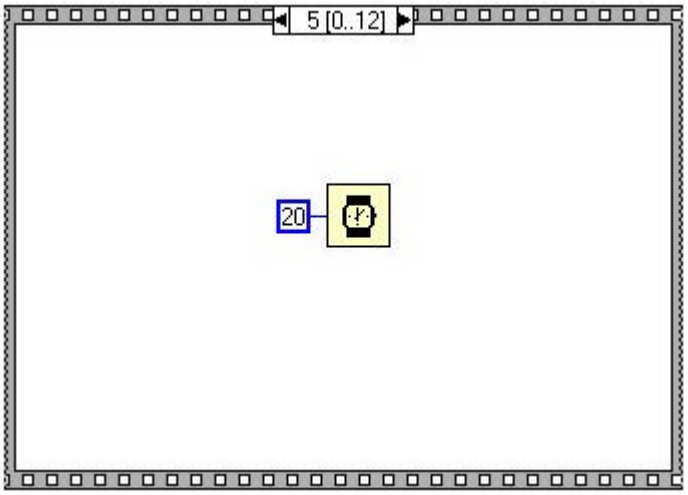
Etape n°3 : pause.



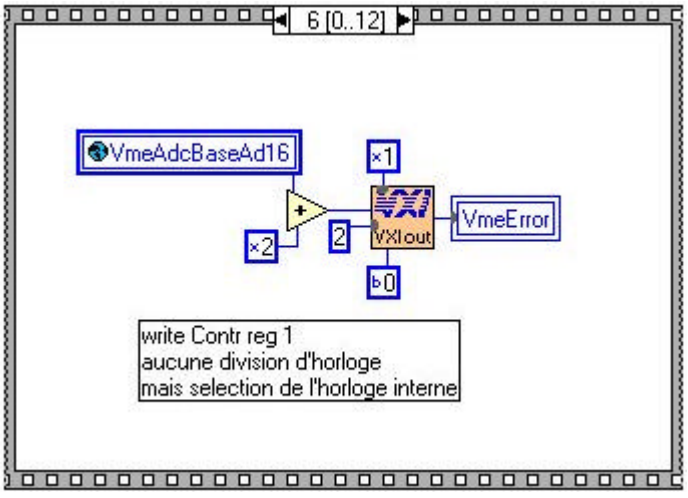
Etape n°4 : configuration de l'ADC.



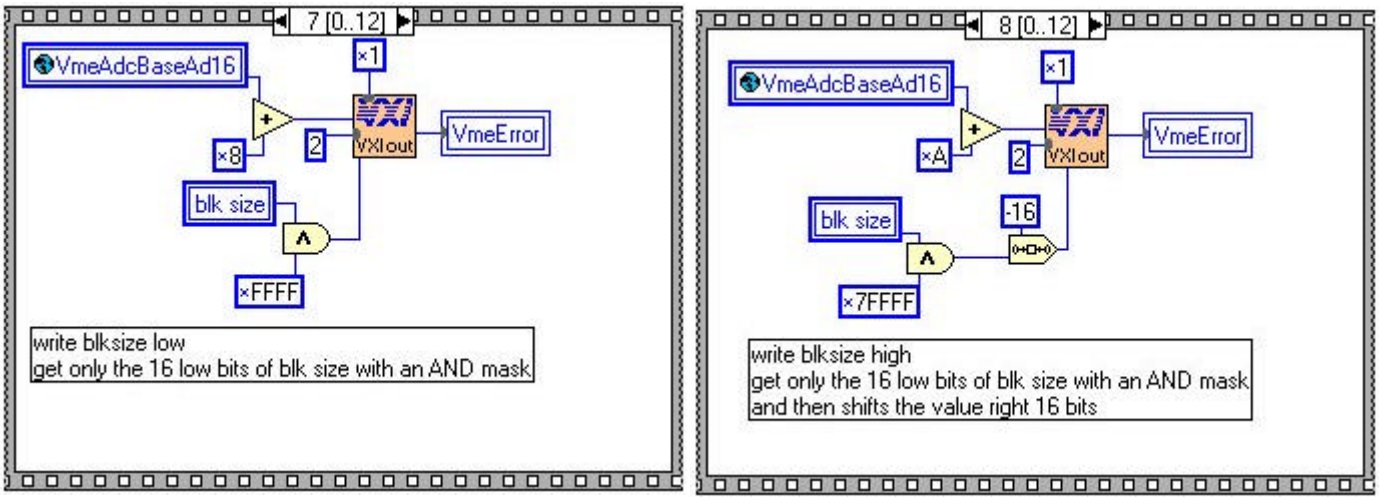
Etape n°5 : pause.



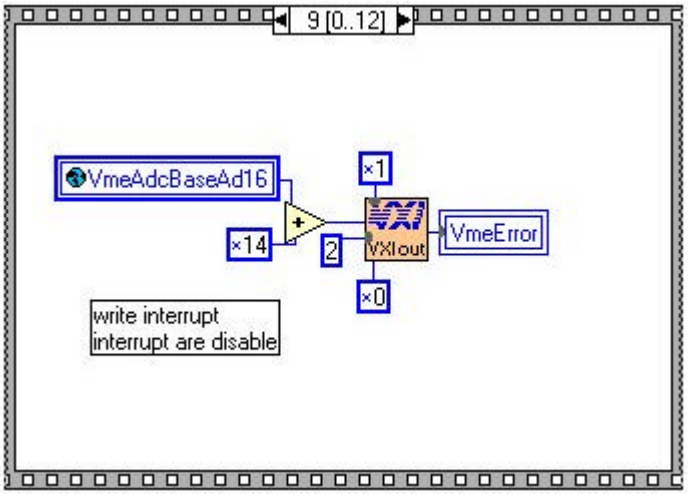
Etape n°6 : paramétrage de l'horloge et de l'accès en mémoire.



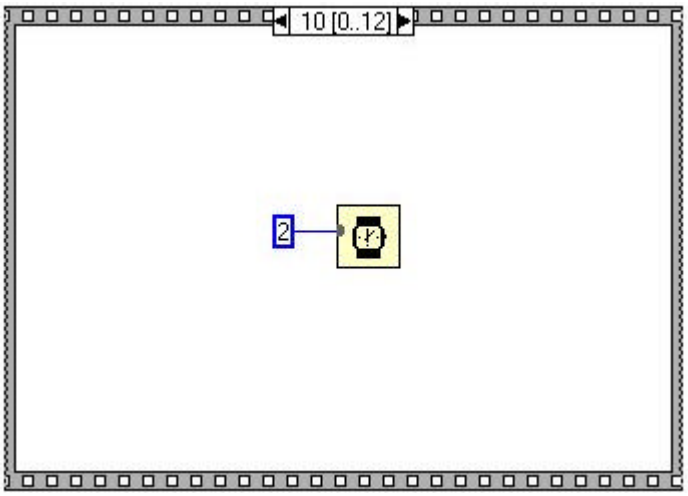
Etape n°7 et 8 : détermine la quantité de mémoire à allouer aux mesures.



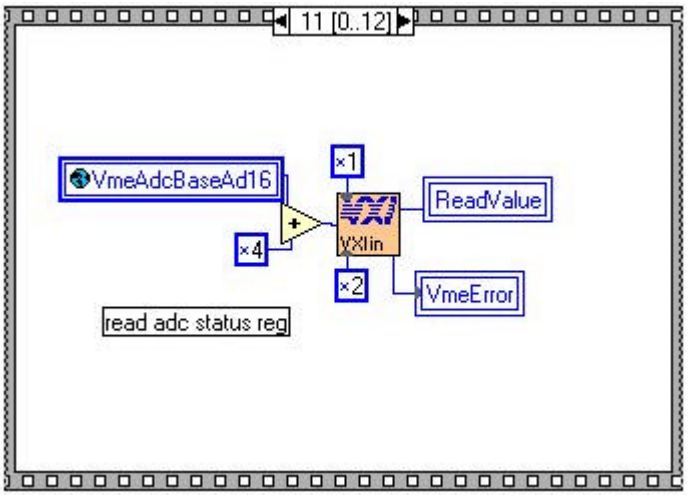
Etape n°9 : interdit les interruptions.



Etape n°10 : pause.

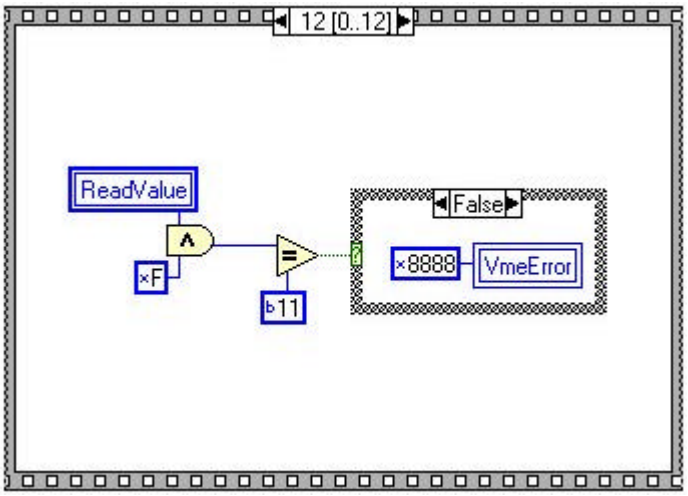


Etape n°11 : lecture du statut de l'ADC.

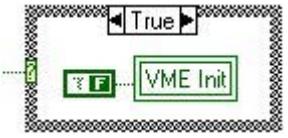


Etape n°12 : les deux bits de poids faible sont-ils égaux à 11(b) ?

Si la valeur du statut correspond au statut attendu en fonction de la configuration qu'on lui a paramétré précédemment :

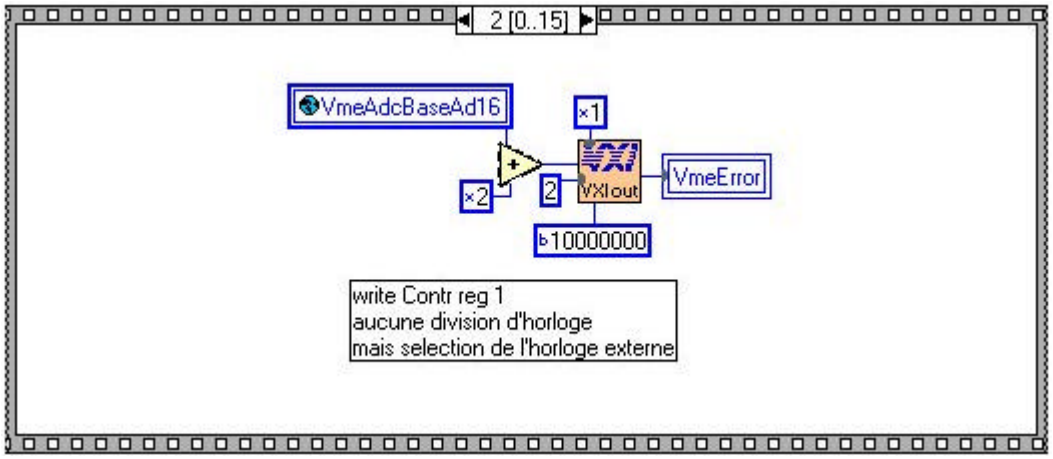


Sinon :

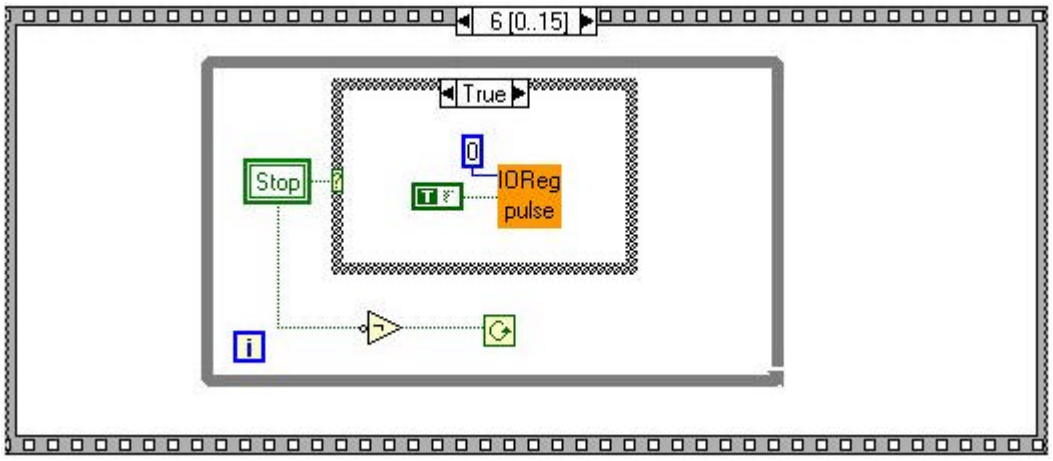


➢ phase de fonctionnement.

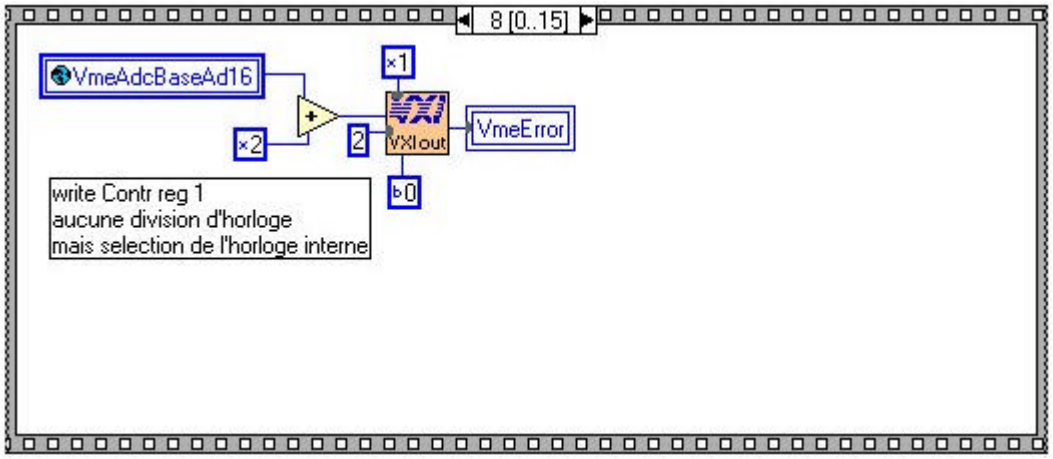
Séquence n°2 : paramétrage de l'horloge et de l'accès en mémoire



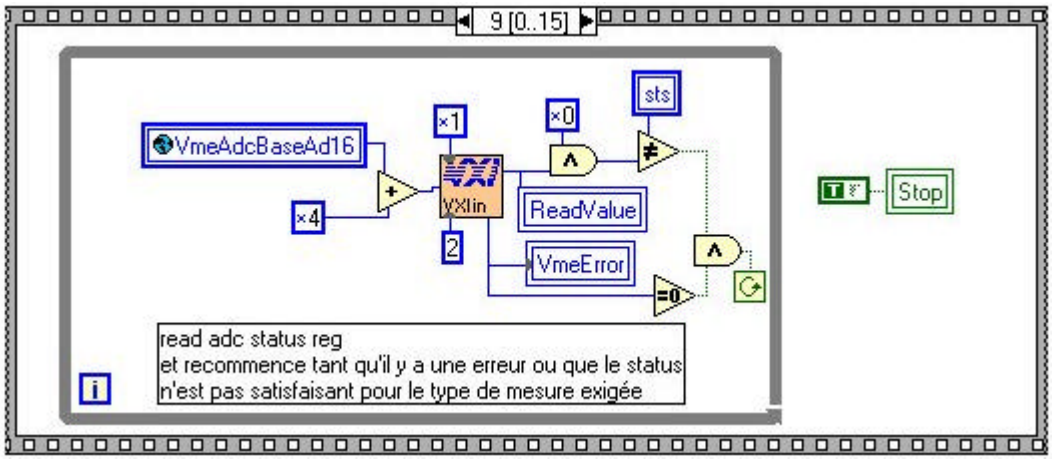
Séquence n°6 : acquisition des mesures.



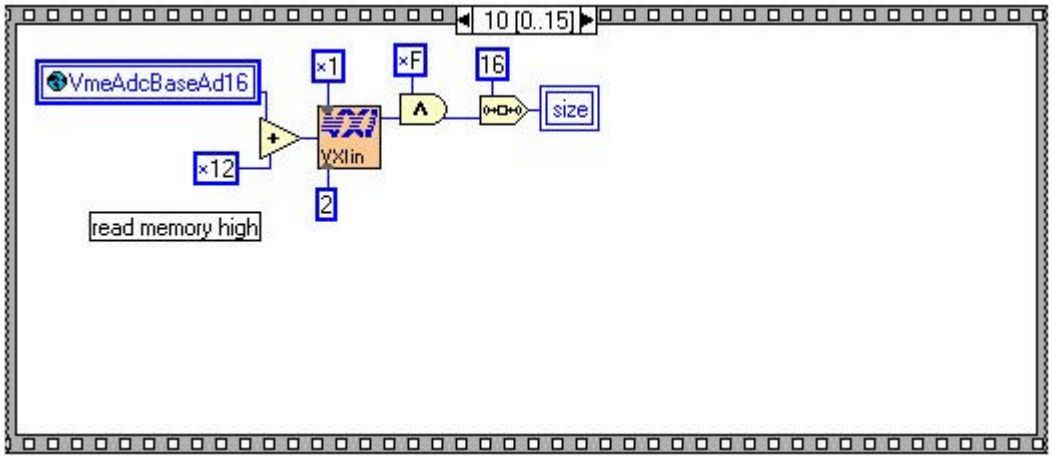
Séquence n°8 : paramétrage de l'horloge interne.



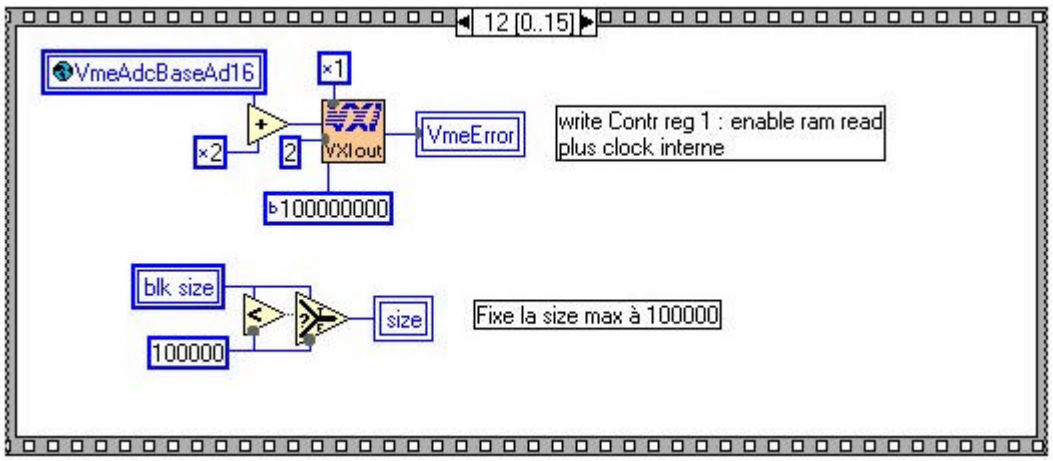
Séquence n°9 : comparaison du statut de l'ADC avec le statut attendu lorsque les mesures sont terminées.



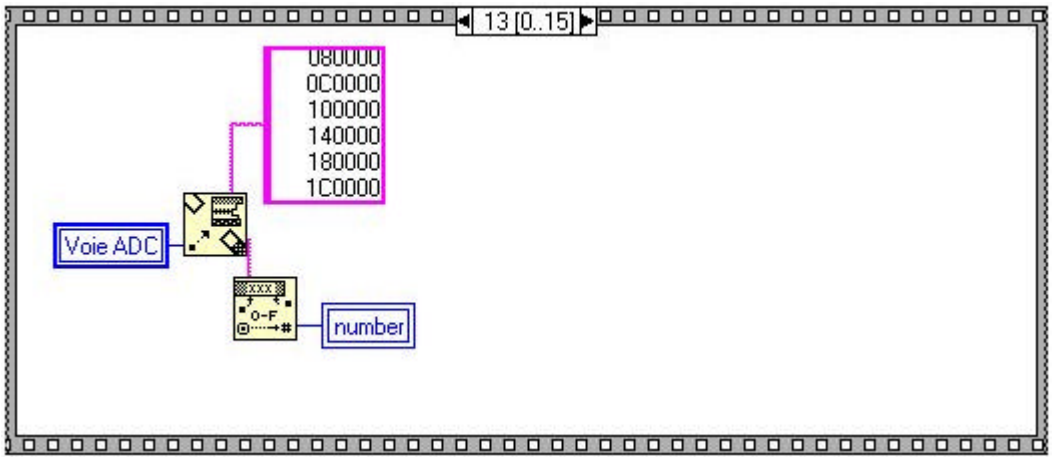
Séquence n°10 : lecture de la valeur du pointeur mémoire.



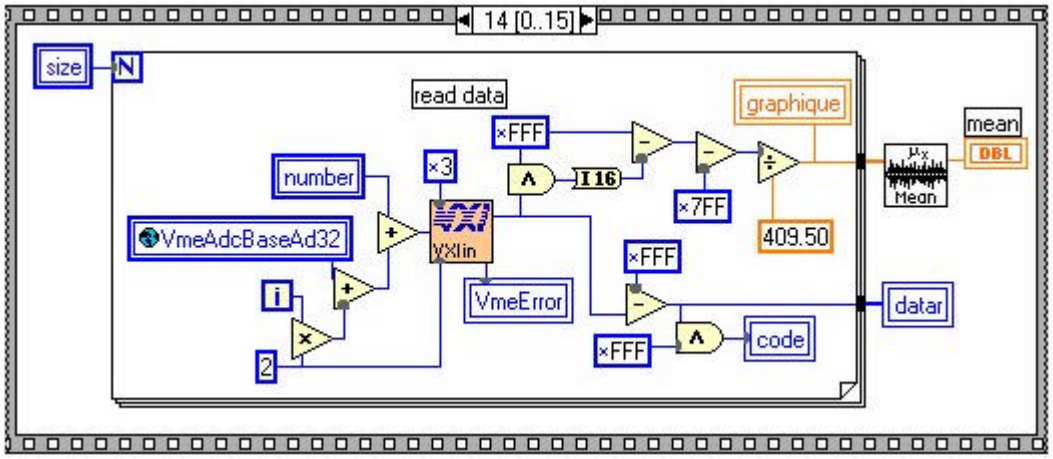
Séquence n°12 : fixe la taille maximale de l'échantillonnage à 100000 et autorise l'accès en mémoire pour l'utilisateur.



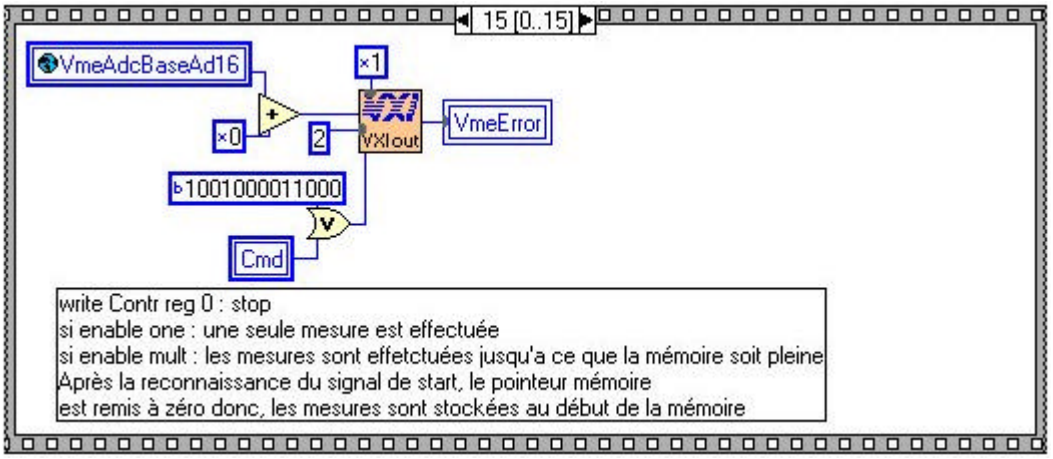
Séquence n°13 : détermine la zone de mémoire à lire.



Séquence n°14 : lecture de la mémoire d'échantillonnage, trace le graphique et renvoie les résultats des mesures.



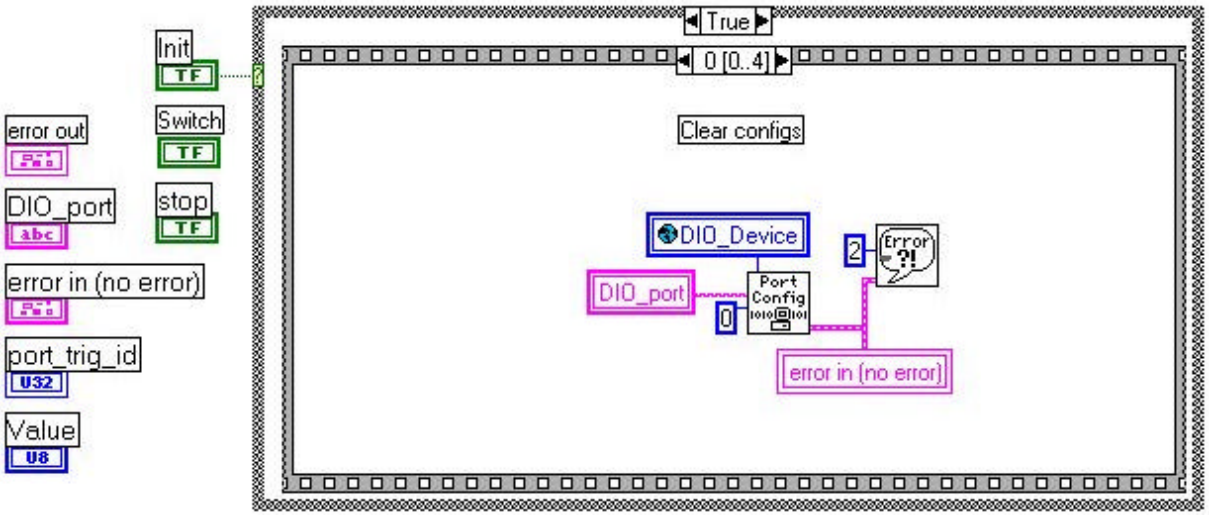
Séquence n°15 : place l'ADC dans un état adéquat pour acquérir de nouvelles valeurs.



Réalisation d'un VI pilotant la carte de multiplexage.

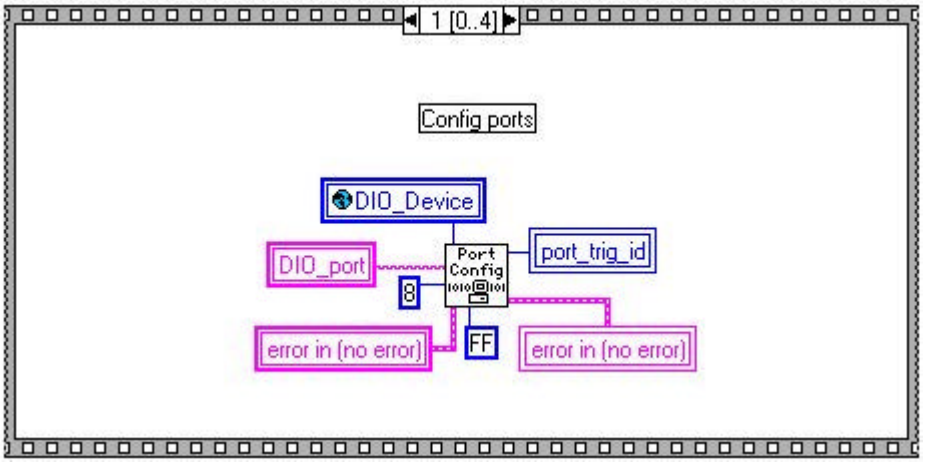
➢ Phase d'initialisation :

Séquence n°0 : effacement des configurations existantes.



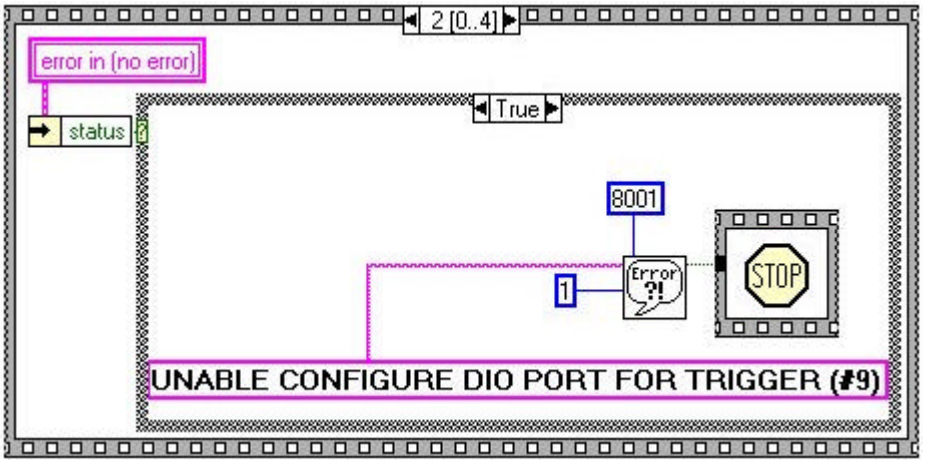
Cette séquence permet d'effacer toutes les configurations existantes sur le port DIO considéré par le numéro DIO_port.

Séquence n°1 : définition de la configuration du port utilisé.

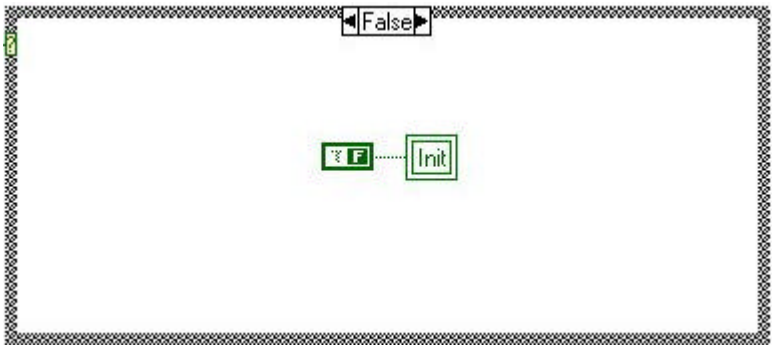


Séquence n°2 : une erreur est-elle survenue ?

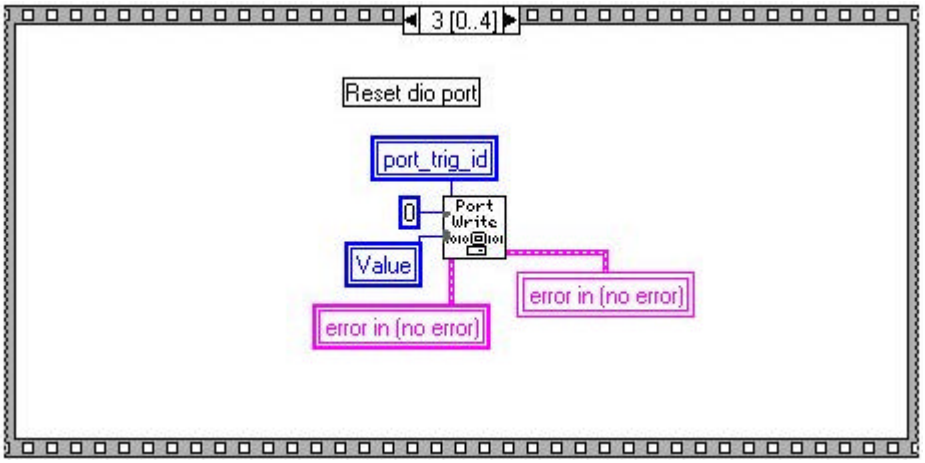
Si une erreur s'est produite :



Si aucune erreur n'est apparue :

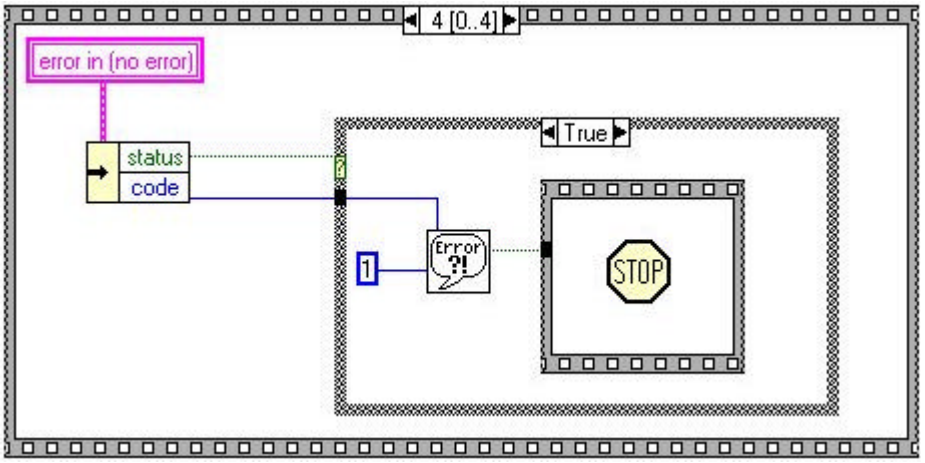


Séquence n°3 : remise à zéro de tous les bits du port utilisé.

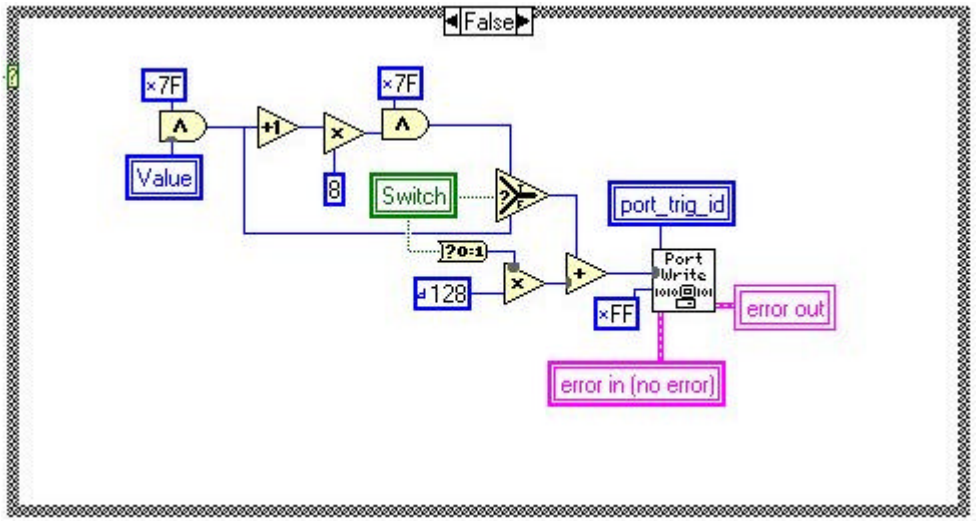


Séquence n°4 : une erreur est-elle survenue ?

Si une erreur s'est produite :



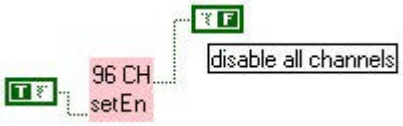
➢ Phase d'envoi des données : envoie de la valeur sur le port du DIO utilisé.



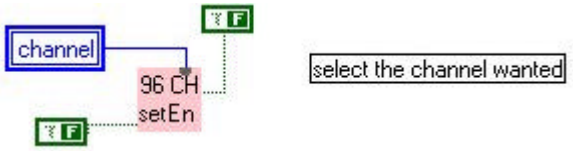
Réalisation de l'interface du banc de test

1. Interface pour la linéarité en courant.

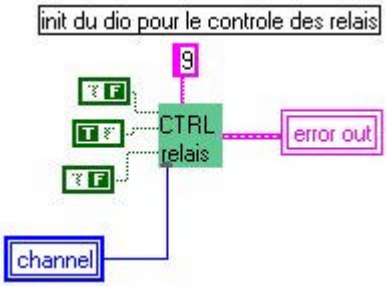
Séquence 0 : aucun canal n'est validé.



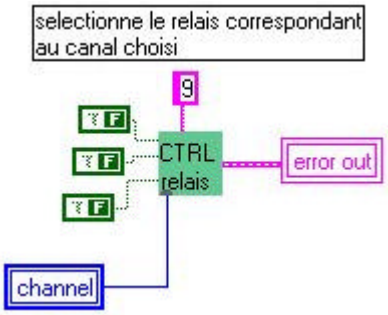
Séquence 1 : valide uniquement le canal choisi.



Séquence 2 : initialisation du DIO pour le pilotage de la carte de multiplexage.

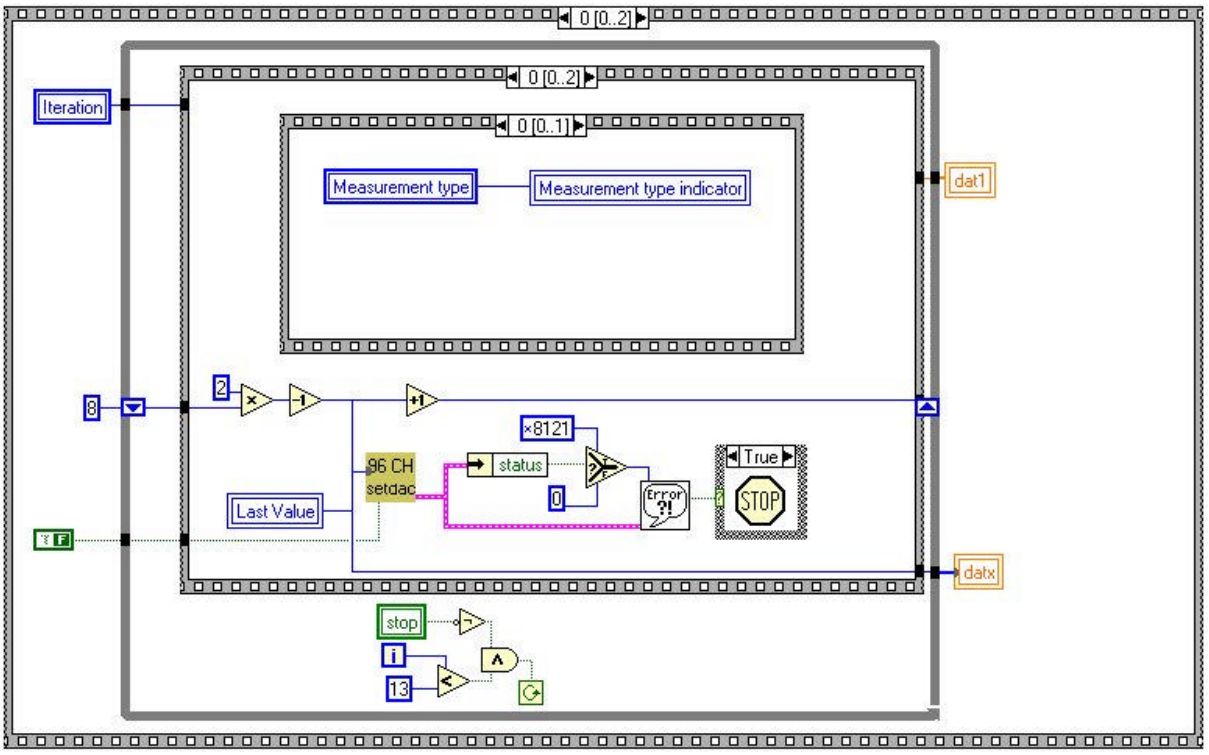


Séquence 3 : sélectionne le relais correspondant au canal choisi.

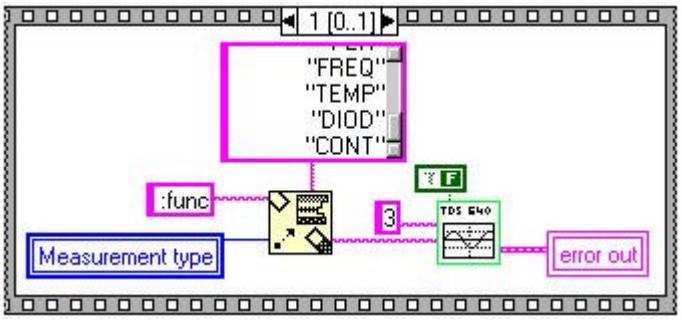


Séquence 4 :

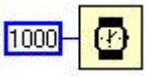
Envoie une donnée vers la carte à tester (c'est le VI « 96 CH setdac » qui permet l'envoi de la donnée).



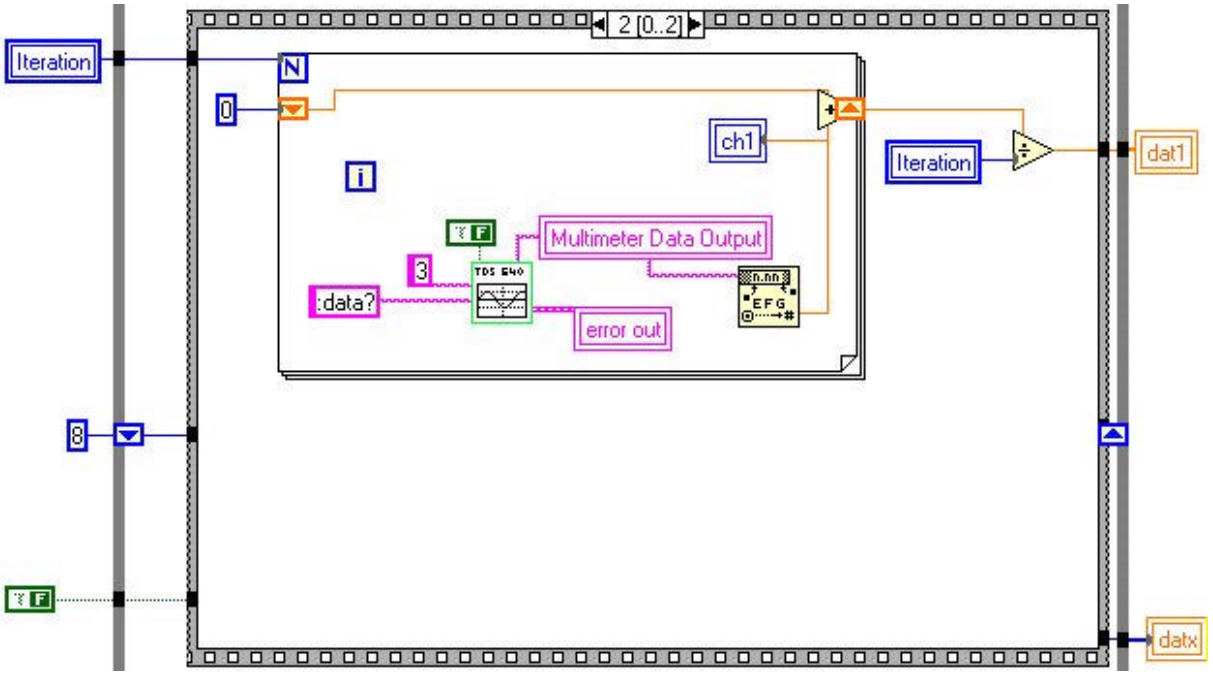
Sélectionne le type de mesure à effectuer avec le multimètre.



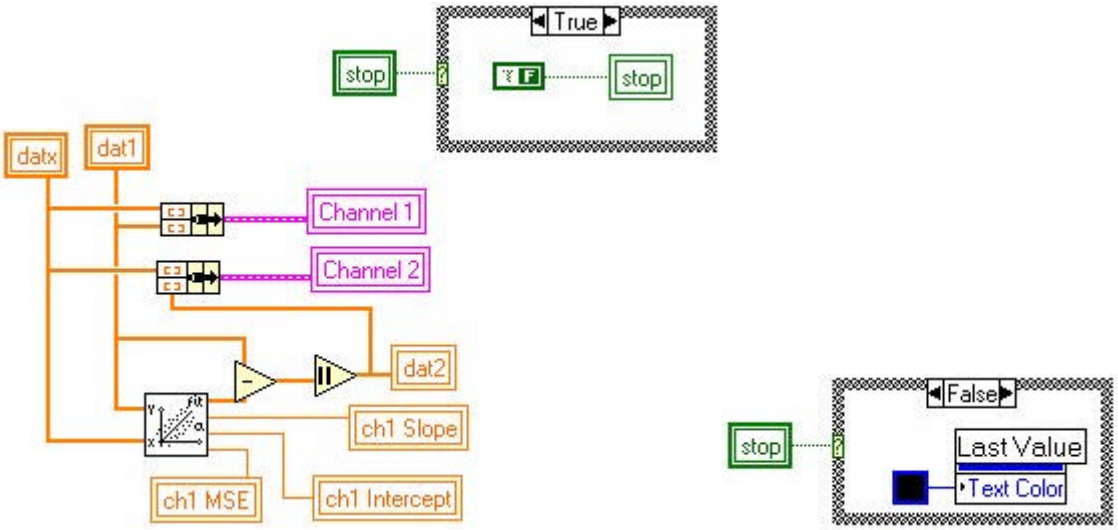
Attendre que le courant se soit stabilisé pour en suite réaliser les mesures.



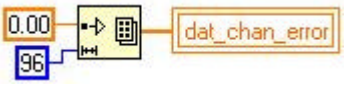
Prendre les mesures avec le nombre d'itérations spécifié dans la face avant.



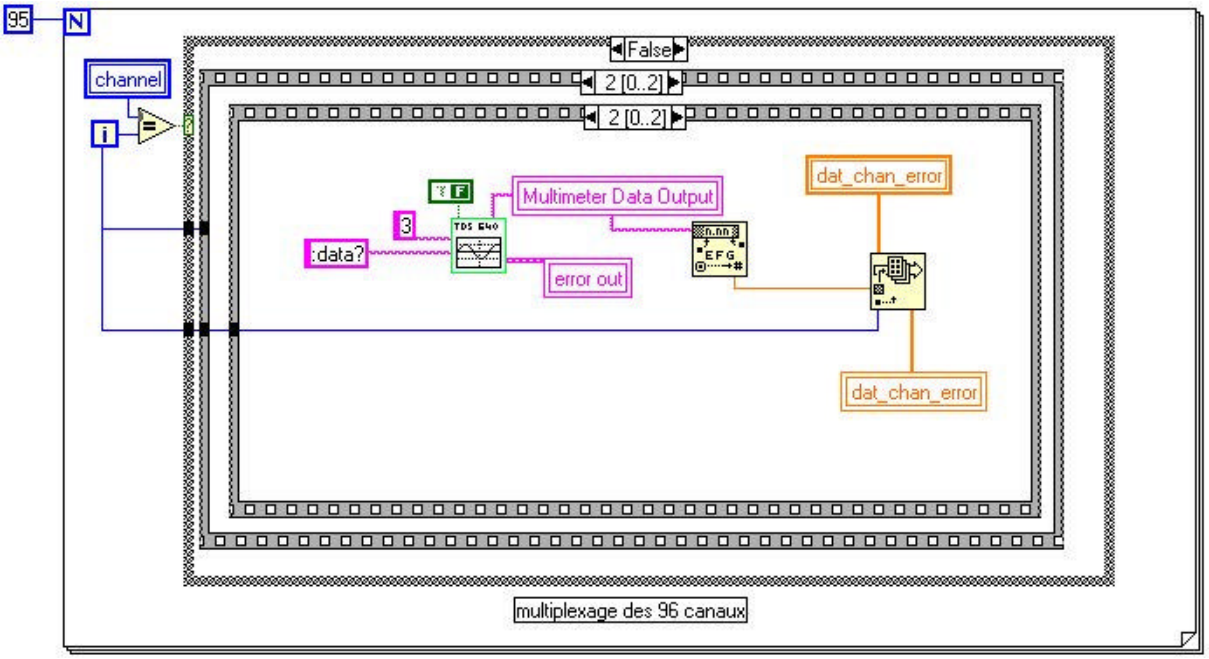
Trace la courbe de réponse à la rampe ainsi que de son écart par rapport à sa régression linéaire.



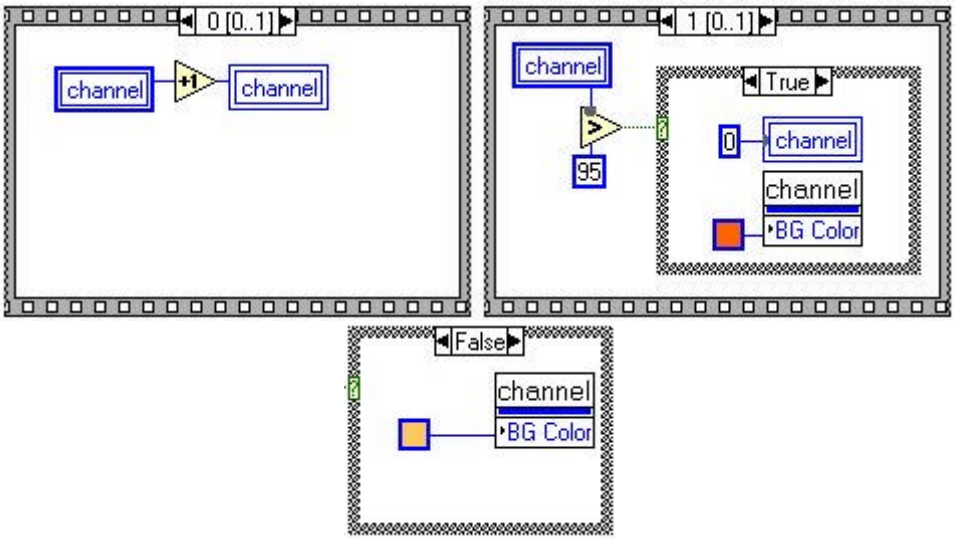
Séquence 5: création d'une matrice de dimension 96.



Séquence 6 : examen de chacune des voies.

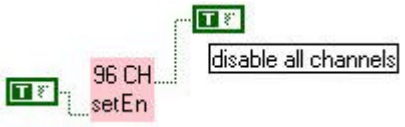


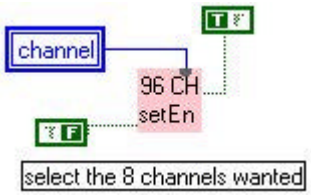
Séquence 7 : incrémentation du numéro du canal jusqu'à 95.



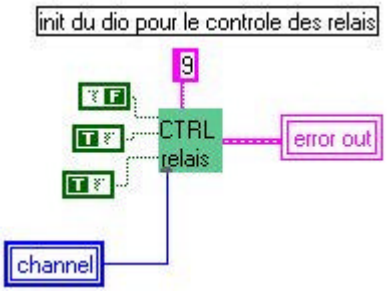
2. Mesure de l'amplitude des impulsions en sortie du *fan out*.

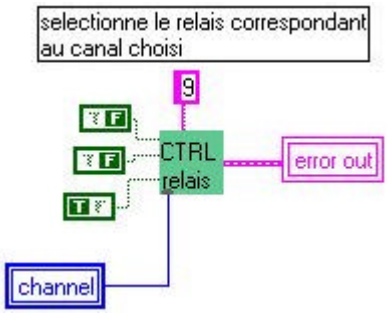
Séquence 0 et 1 : sélection des voies huit par huit.





Séquence 2 et 3 : sélection des relais huit par huit.





Séquence 4 : initialisation de l'ADC.

Séquence 5 : acquisition des mesures.