# Réalisation d'un banc de deux machines asynchrones linéaires en opposition avec mise en œuvre du contrôle commande spécifique

Rolf Pérot

# MEMOIRE

Présenté en vue d'obtenir le

## DIPLOME D'INGENIEUR C.N.A.M

## en Electrotechnique

par Rolf PEROT

# Réalisation d'un banc de deux Machines asynchrones linéaires en opposition avec mise en œuvre du contrôle commande spécifique.

**Présenté le : 05/07/2013**

Jury :

| | |
|---|---|
| Mr Clément RAMIARINJAONA | Professeur au CNAM Président |
| Mr Sami HLIOUI | Maitre de conférences au CNAM de Paris |
| Mr Laurent Prévond | Maitre de conférences au CNAM de Paris |
| Mr. Hervé VALAT | Directeur de la société SERVAT |

**Résumé**

Ce mémoire se porte sur l'étude, la réalisation et la mise en œuvre d'un banc d'essai avec sa commande spécifique. Ce banc comprend 2 machines asynchrones linéaires innovantes montées en opposition. L'objectif principal de ce banc est de valider le principe de la génération d'électricité par une machine asynchrone linéaire, objet d'un important projet de micro cogénérateur Stirling dans lequel s'intègre ce mémoire. Les machines asynchrones ont été dimensionnées pour répondre au cahier des charges de ce micro cogénérateur.

Le travail a consisté tout d'abord à définir le banc d'essai qui a été ensuite réalisé par la société SERVAT. J'ai ensuite mis en place l'ensemble des éléments du banc permettant son fonctionnement. Un important travail a consisté dans la mise en place du contrôle commande. Le matériel électronique de puissance, de la société BR Automation, nous a permis de piloter le moteur et le générateur. J'ai développé une interface homme machine qui permet de simplifier le pilotage. La mise en place du programme a constitué, à elle seule la partie principale de ce mémoire, car la familiarisation avec la logique de programmation et du logiciel est laborieuse.

Cette interface nous permet de définir la course et la fréquence mécanique de battement. Le logiciel permet aussi de faire des acquisitions des grandeurs électriques (courants, puissance, etc…) temporelles triphasées et dans le domaine de Park. Nous avons constaté que les commandes n'étaient pas optimales et induisaient d'importantes pertes. La variation des paramètres nécessaires au contrôle vectoriel a permis de diminuer les pertes. Le fonctionnement en mode moteur et générateur n'est toujours pas optimal, de ce fait la régénération n'a pas pu être faite.

**Mots-clefs :** Machine linéaire asynchrone, Commande Vectorielle, API, axe maitre/esclave, sustentation magnétique

**Summary**

This engineer thesis is regarding the study, the achievement, and the implementation of a platform with it's specific control system. The platform consists of two linear inductions machines connected together. The main goal is to validate the possibility of supplying electricity with an linear induction machine, which is part of an important micro cogeneration project including a Stirling motor. The machines were configured to comply with the design set.

The work had consisted first to establish the platform, then after being developed by SERVAT, I did put into place all the elements. An important and major work has taken place concerning the handling and the programming of the automaton. The electronic devices from the BR Automation company have allowed the monitoring of the motor and the generator. Thus I did make a browsing interface which simplifies the use of it. To put into place the programs has taken the most important part of this engineer thesis because of its difficulty.

The browsing interface has allowed us to define the length and the mechanical frequency of the moving shaft. The soft also enable us to acquire different parameters such as the current and the power (whether in two or in the three phase domain).

We've seen that the control system weren't optimal and power losses were then important. The variation of some parameters had some influence on the results. The control of the motor and the generator being not completely achieved, the regeneration of electrical power can't be done yet.

**Keywords :** Linear induction machine, Vector Control, PLC, slave/master axis, magnetic levitation

Sommaire

# I. Introduction générale

Le projet de micro cogénérateur (projet CETI ANR du programme blanc) consiste à mettre en œuvre un moteur Stirling constitué d'un échangeur à plaques pour le réchauffement d'un circuit d'eau et d'une génératrice asynchrone en bout d'arbre. Le mémoire se porte sur la partie électrique du projet, à savoir la génératrice asynchrone.

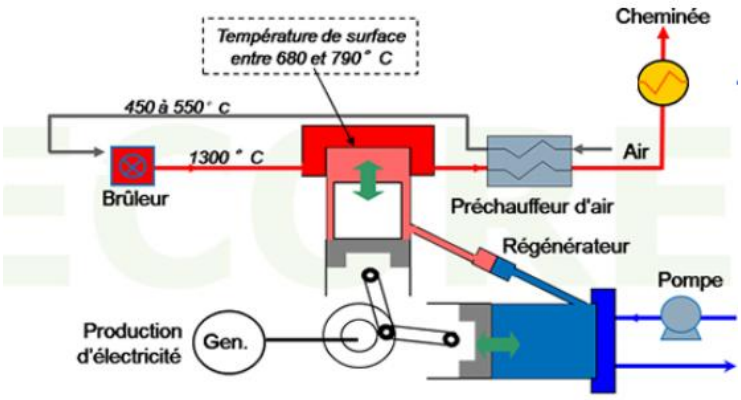



**Figure I-1**

Schéma de principe du moteur Stirling

**Figure I-2**

Photo de la maquette Stirling en cours de construction au laboratoire FEMTO de Belfort.

Un financement ANR a été obtenu pour financer le matériel et les personnes travaillant sur le projet. Le laboratoire SATIE est le porteur de ce projet et les partenaires sont le laboratoire FEMTO et l'industriel GDF SUEZ.

En outre, les parties thermique et électrique ont été traité par :

- M. Rachid Hani qui a fait l'objet d'un mémoire d'ingénieur sur la partie électrique
- M. Pierre François qui a fait l'objet d'une thèse sur la partie électrique
- Mme Garcia Isabelle qui a fait l'objet d'une thèse sur la partie électrique, automatique et mécanique.

Ce mémoire d'ingénieur porte uniquement sur la partie électrique de la cogénération.

# II. Remerciements

Ce mémoire a été réalisé au sein du Laboratoire d'Électrotechnique au Cnam de Paris en collaboration avec la société SERVAT (Spécialisée dans la fabrication de moteurs électriques spéciaux) ainsi que la société BR automation (Spécialisé en automatique).

Mes remerciements vont dans en premier lieu à Laurent PREVOND, Maître de conférences à la chaire d'électrotechnique du CNAM de Paris pour sa confiance en me proposant ce sujet.

Je tiens à remercier Messieurs Sami HLIOUI maitre de conférences au CNAM qui est mon tuteur enseignant dans mon cycle d'ingénieur au CNAM, ainsi que Frédéric SEGRETAIN que j'ai sollicité à plusieurs reprises sur des questions techniques.

Je remercie également Messieurs Thibault et Hervé VALAT, pour m'avoir accueilli au sein de leur entreprise afin de me familiariser avec le logiciel Solidworks et de pouvoir suivre les étapes de fabrication de la machine asynchrone linéaire.

Je suis reconnaissant envers mon épouse Svetlana qui grâce à son travail pénible a pu subvenir au besoin financier du foyer.

Enfin, je remercie vivement mon précédant employeur la société AXONE, à savoir Mme Véronique FERRAGLIO et Mr Sylvain VIRATEL qui sans leur dévouement, ce mémoire aurait sans doute était abrogé.

# III. Abbreviations/Acronyms

ANR : Agence National de la Recherche

API : Programmable Programmable Industriel

AS: Automation Studio

GRAFCET : Graphe Fonctionnel de Commande Etape Transition

MAS: Machine Asynchrone

MASL : Machine Asynchrone Linéaire

MCCL : Machine à Courant Continue Linéaire

d: distance de la course du mover

Lm : Inductance magnétisante

Mover : partie mobile de la machine

n= nombre de phase

p= nombre de paire de pôle

PLC : Programmable Logic Controller

Rr : Résistance rotorique

Lr : Inductance rotorique

τ : pas polaire

# IV. Historique et usage des moteurs linéaires

Le moteur linéaire, comme son nom l'indique, permet d'avoir un mouvement de type rectiligne tout en s'affranchissant d'une quelconque transformation mécanique d'un mouvement rotatif en un mouvement linéaire (par le biais de vis sans fin,ou bielle manivelle, …).

Le concept est aussi ancien que celui du moteur rotatif. Il dérive directement des expériences d'Arago et de Faraday et des dispositifs à champ glissant de Tesla et de Ferraris. L'idée est cependant réapparue au début des années 1900 ou deux américains, Robert Goddard et Emile Bachelet ont proposé l'idée d'utiliser la force magnétique pour le transport à haute vitesse, mais la technologie de la commande avait ses limites…

D'autre recherche ont depuis eu lieu notamment par Hermann Semper en 1933 ou encore par Eric Laithwaite qui en 1940 mis au point un modèle de train linéaire synchrone fiable. (Fig. IV-1) **[A]**

Figure IV-1

Depuis des projets d'ampleur ont vu le jour, et pas seulement sur le transport.

En effet la volonté d'optimiser les processus, demande parallèlement une exigence accrue en termes de rapidité et de précision sur l'ensemble du procédé de fabrication. Les commandes actuelles de moteur permettent de faire varier avec grande précision la pente d'accélération ou de décélération ce qui donne aux moteurs linéaires une application importante tel que le transport ou dans le guidage linéaire.

## A. Exemple d'application :

Premier exemple: Le train à sustentation magnétique

Figure IV-2

Train à sustentation magnétique reliant
l'aéroport de Shanghai (Pudong) à la ville

Les 30 km de la ligne franchie par le Transrapid sont parcourus en moins de 8 minutes. Le Transrapid accélère de 0 à 300 km/h sur une distance de 5 km (les trains à grande vitesse classiques ont besoin de 20 km pour atteindre la même vitesse).

Deuxième exemple: Manutention de palette

Figure IV-3

Transpalétage de produit par un moteur linéaire synchrone

# V. Principe des moteurs Linéaire électrique:

## A. Physique Globale

Le principe de la loi de Laplace et de coulombs nous indique qu'une force existe lorsque d'une manière générale il y a interaction entre deux champs magnétiques.

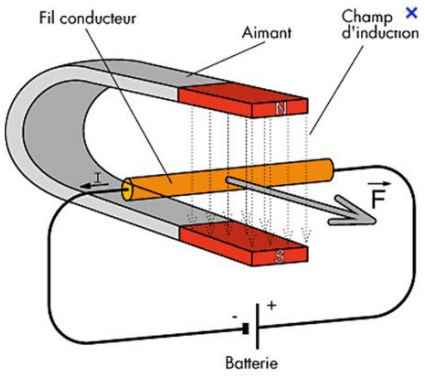



Figure V-1

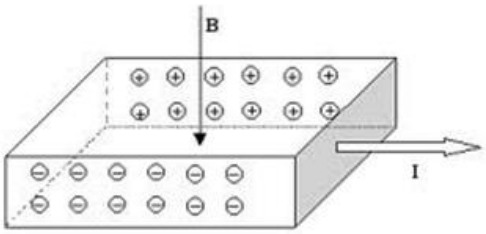



Figure V-2

**Force de Laplace**

**Force de Lorentz**

Un moteur linéaire électrique est basé sur le principe de la force de Laplace.

En s'appuyant sur la figure V-1, on comprend que tant que la barre se trouve dans le champ magnétique, celle-ci subira une force qui tendra à la déplacer suivant le sens du champ magnétique B et du courant I décrit par la relation.

Les caractéristiques de déplacement, à savoir la **force**, la **vitesse** et l'accélération sont des paramètres contrôlables par le courant I et/ou le champ d'induction B.

## B. Type de moteur linéaire

### 1. Moteur linéaire à courant continu avec aimant

Une possibilité très simple de créer un moteur linéaire peut donc se faire avec une alimentation de type courant continu. La figure V-3 illustre ce principe en ayant comme inducteur un aimant permanent.

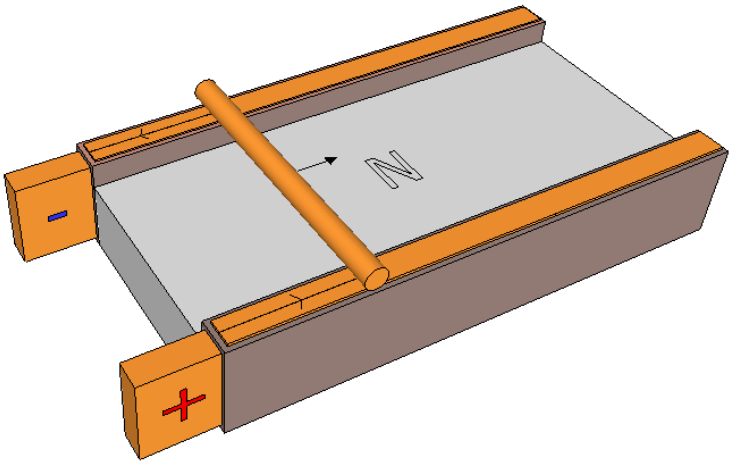



Figure V-3

Les deux bornes étant alimentées, le courant circulera dans la barre (contact par roulement idéalement). La force résultante va entrainer la barre dans le sens indiqué. L'inversion de la polarité inversera le sens de déplacement.

### 2. Moteur linéaire synchrone (MLS)

Il est possible de créer une machine linéaire synchrone en alimentant des enroulements statoriques par une tension alternative sinusoïdale triphasée. Les figures V-4 et V-5 représente une MLS.

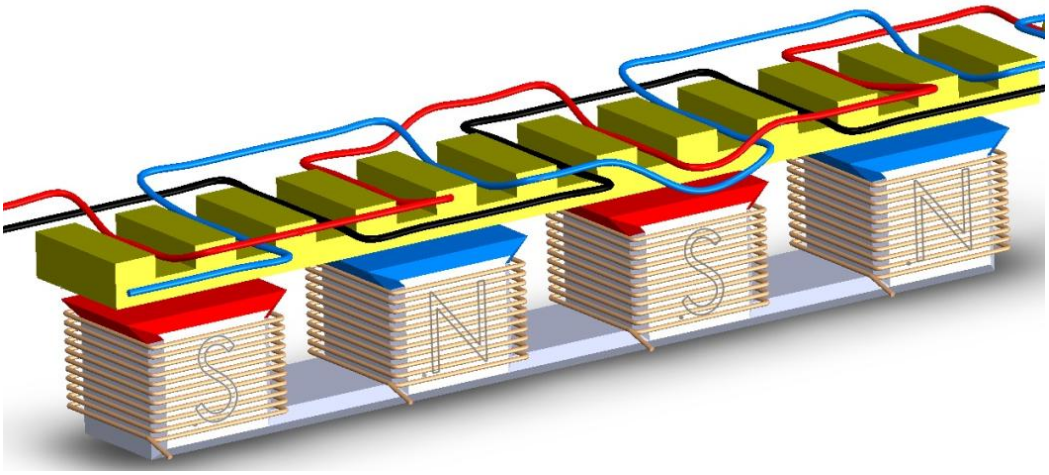



Figure V-4

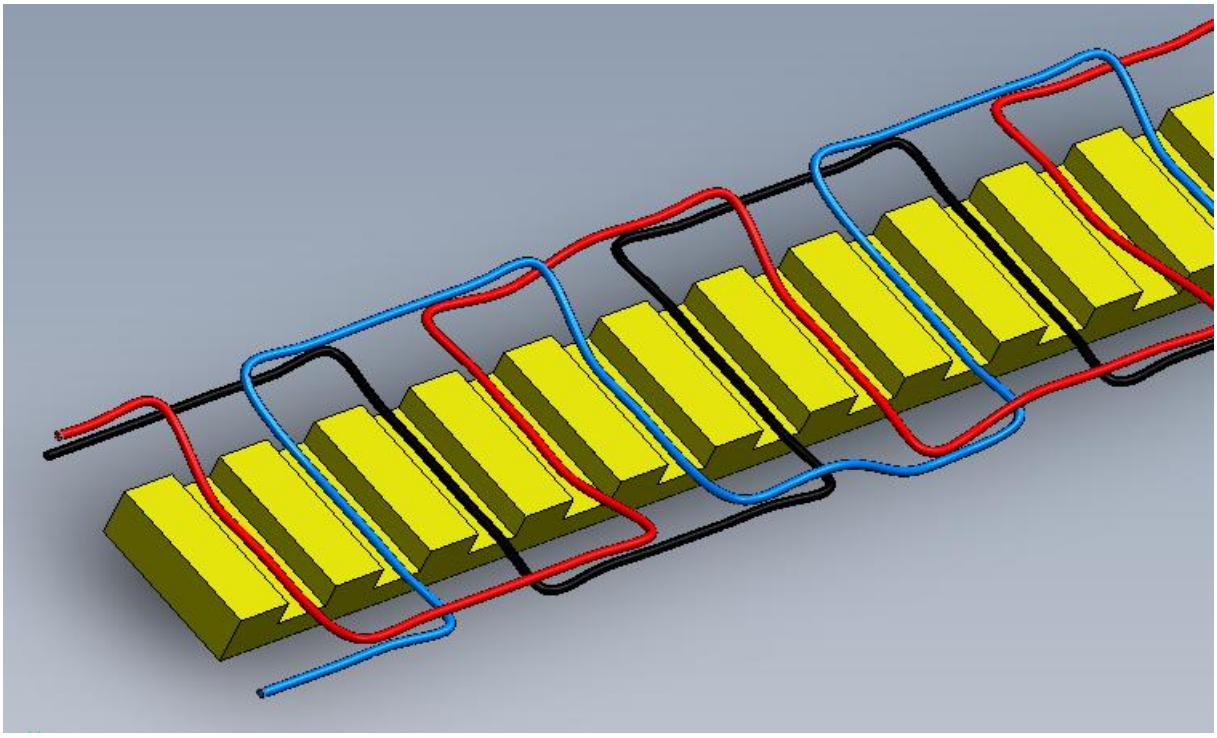


Figure V-5

(Pour des raisons de clarté le feuilletage n'est pas représenté sur la pièce mobile)

Les bobines statorique sont alimentées en courant continu. Leur sens d'enroulement (ou l'alimentation) est inversé consécutivement afin d'avoir un champ magnétique d'induction alterné.

Les enroulements triphasés placés dans les encoches de la partie mobile (qu'on appel mover) vont créés un champ magnétique « glissant ».

Ce type de dispositif linéaire fait l'objet de commercialisation notamment par la société TECHNOTION F La figure V-6 représente un modèle [B]


Figure V-6

Partie fixe : accouplement d'aimant

Partie mobile : Bobinage alimenté en triphasé

Force : 141 N

Vitesse : 2.7 à 6.6 m/s

### 3. Moteur asynchrone

Bien que des prototypes aient vues le jour sur ce type de moteur [C], il n'est pas à l'heure actuelle présent sur le marché industriel, ce qui en fait pour le moment sa particularité. L'inducteur est identique à celui de la machine synchrone, l'induit sera parcouru par des courants de Foucault. Le chapitre suivant en donne l'explication

# VI. Le moteur asynchrone linéaire

La figure VI-1 est une représentation du principe de fonctionnement permettant d'analyser le principe.

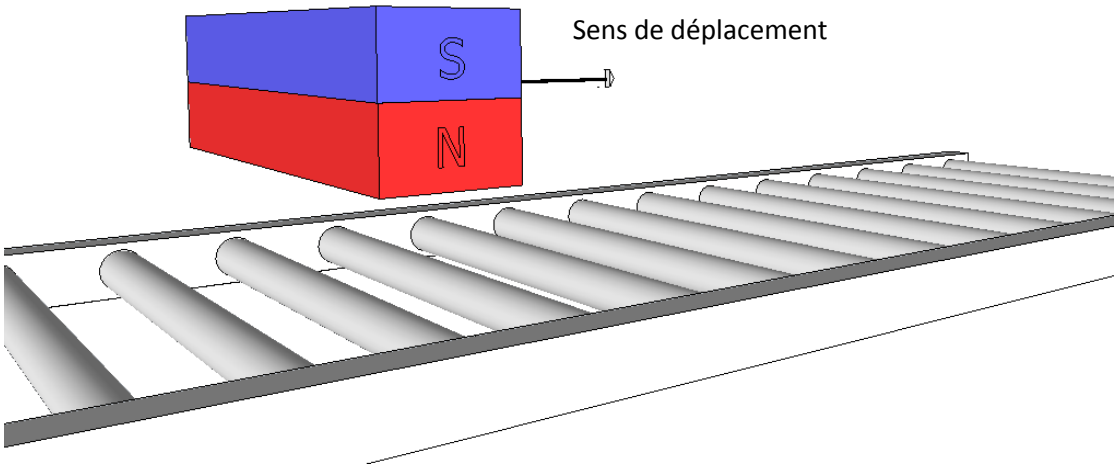



Figure VI-1

## A. Forces en applications :

### 1. Force linéaire :

Le flux de l'aimant coupe les barres conductrices suivant le sens de déplacement indiqué. Les barres sont alors soumises à un champ d'induction variable dépendant de la position et de la vitesse de déplacement de l'aimant.

En vertu de la loi de Lenz, les barres en avant de l'aimant dont le flux commence à les traverser, vont créer un courant qui va s'opposer à cette variation. On verra apparaitre alors un pole Nord dans la zone ou ce flux agit.

En vertu de la loi de Lenz, les barres en arrière de l'aimant dont le flux décroit au fur et à mesure du déplacement de l'aimant, se verront apparaitre un pôle Sud.

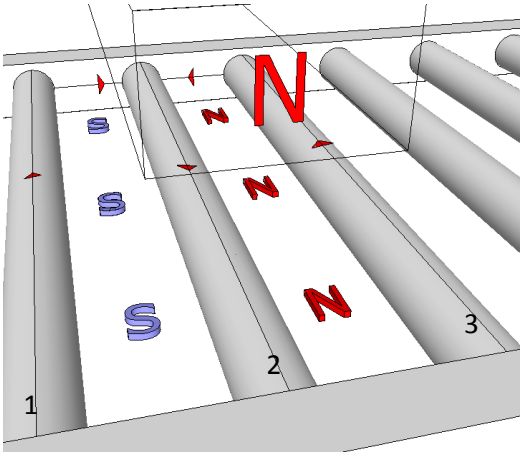



**Figure VI-2**

Représentation du courant et des pôles induit

Une force longitudinale existe tant que la « plaque » est en avance ou en retard par rapport au champ électromagnétique

## 2. Force de lévitation (répulsion) :

Supposons que, à un instant donné, le centre du pole N de l'aimant passe au dessus du conducteur 2. Le champ magnétique balayant ce conducteur y induit une tension qui est alors maximale. Si l'aimant se déplace lentement, le courant induit dans ce conducteur atteint sa valeur maximale en même temps que la tension.

Ce courant, revenant par les conducteurs 1 et 3 crée des pôles magnétiques n et s comme l'indique la figure.

On constate alors que, selon la loi de l'attraction et de répulsion, la partie avant de l'aimant est repoussée vers le haute et que la partie arrière est attirée vers le bas.

Comme la distribution des pôles n et s est symétrique par rapport au centre de l'aimant, les forces verticales d'attraction et de répulsion sont égales et la force résultantes est nulle ; il ne reste donc que la force de traction horizontale.(force externe)

Supposons maintenant que l'aimant se déplace très rapidement. A cause de l'inductance des conducteurs, le courant dans le conducteur 2 atteint sa valeur maximale une fraction de seconde après le maximum de tension induite.

Par conséquent, lorsque le courant dans le conducteur 2 est maximal, l'aimant se trouve déjà à une certaine distance en avant de ce conducteur.

Le courant, revenant par les conducteurs 1 et 3 crée encore des pôles n et s comme auparavant ; cependant, le pole N de l'aimant se trouve maintenant entièrement au dessus d'un pole n et il en résulte une force verticale importante qui repousse l'aimant mobile vers le haut.

C'est le principe de la sustentation magnétique.

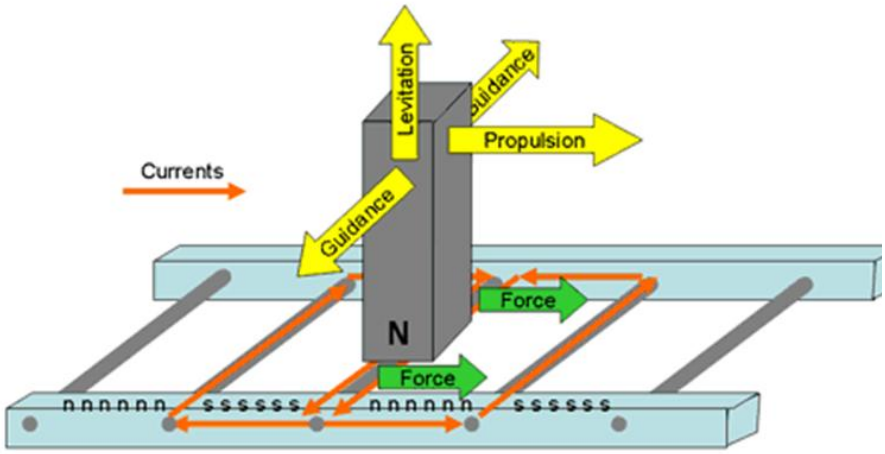



Figure VI-3

Illustration de l'ensemble des forces [B]

Selon les besoins, Il existe différentes techniques pour fabriquer un moteur linéaire asynchrone. Les figures VI-4 à VI-8 les illustrent.

Plusieurs possibilités s'offrent à nous quand au nombre de phases constituant l'inducteur et le type de bobinage pouvant être employé (enroulement en bobines par pôle, enroulement en bobines par pôle consécutifs…)

Chaque modèle de machine linéaire asynchrone (MAL) est constitué de bobinage triphasé

## B. Généralité sur les types de machine asynchrone linéaire possible

### 1. MAL avec induit en forme d'échelle

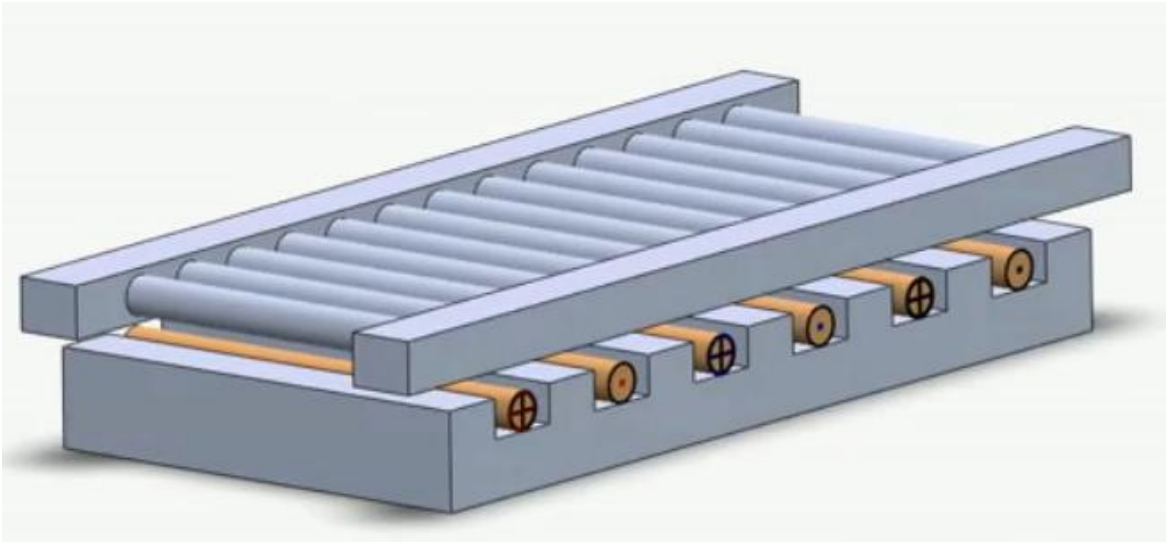


Figure VI-4

L'induit est tel que nous l'avions représenté à la figure VI-1. Il est en forme d'échelle. Sa matière peut être amagnétique (cuivre, aluminium..) ou magnétique (fer ou autres, attention dans ce cas à la force d'attraction magnétique).

En augmentant le nombre de barre (L'espacement entre elles diminuent donc proportionnellement) on augmente la quantité de flux coupé, d'où une amélioration des performances de la machine.

Aussi pour maximiser l'interaction entre le courant induit et le champ inducteur, la partie latérale se situera hors de la partie active du champ inducteur. Figure VI-5

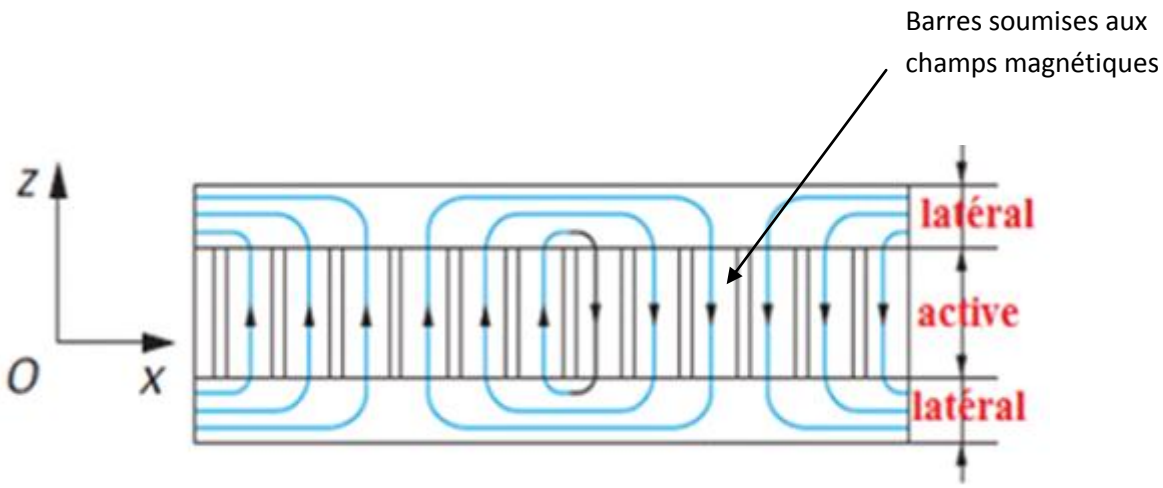



Figure VI-5

Vue de dessus de l'induit : distribution du courant

Ainsi les courants ont une composante longitudinale suivant la direction X dans la partie latérale, et une composante transversale suivant la direction Z dans la partie active.

## 2. MAL avec induit Massif

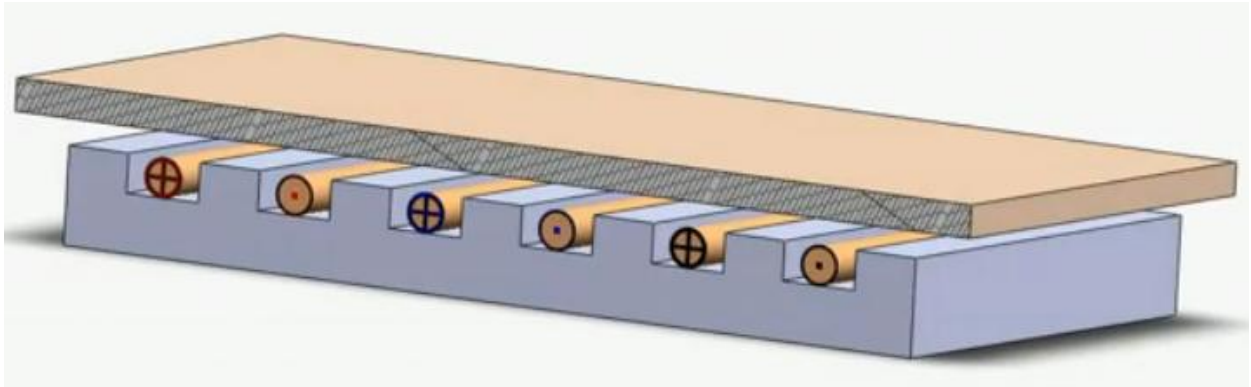


Figure VI-6

Même remarque que précédemment, pour maximiser l'interaction entre le courant induit et le champ inducteur, la partie latérale se situera hors de la partie active du champ inducteur. Figure VI-7

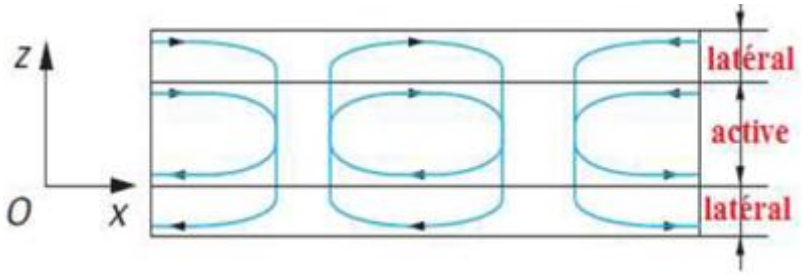



Figure VI-7

Vue de dessus de l'induit : distribution du courant

La composante utile du courant induit selon *Z* pour un secondaire massif diminue par rapport à un rotor en forme d'échelle, car le courant se reboucle également dans la partie active. Ainsi la force de poussée est diminuée.

### 3. MAL avec induit et circuit magnétique de fermeture

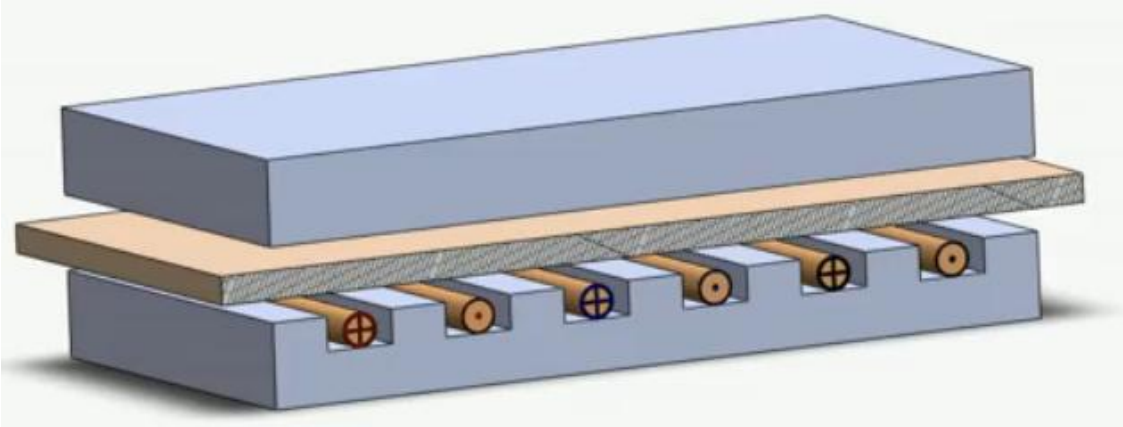


Figure VI-8

L'induit inférieur, qui est fixe, est composé d'un matériau amagnétique tandis que l'induit supérieur qui est fixe également est composé d'un matériau magnétique.

L'avantage de ce moteur réside dans le fait que la partie mobile est plus légère, donc on augmente les performances dynamiques.

Largeur finie - effet transversal – effet d'extrémité.

Les courants induits au secondaire sont des courants de Foucault qui ne sont pas distribués uniformément selon la direction Z. Cette distribution modifie les pertes et la répartition du champ. Cet effet est l'effet d'extrémité de largeur finie.

# VII. Présentation de la machine

## A. Généralité

En se basant sur la figure VI-8, imaginons que l'on relie les quatre extrémités de chaque constituant pour former un tube. Les propriétés physiques ne sont pas altérées et on obtient une machine de type tubulaire, celle sur lequel le travail se porte.

Au repos, le mover repose mécaniquement sur ses appuis (2 paliers lisses « Igus »). C'est seulement lorsque les bobines sont alimentées que la force de lévitation fait que la machine est sustentée naturellement. Il n'y a donc théoriquement pas de frottement en fonctionnement.

## B. Matériaux utilisés

**Les tôles** empilées entre chaque bobine sont faites avec un matériau ferromagnétique afin de canaliser le flux dans le sens voulu et afin d'éviter au maximum les courants de Foucault statoriques.

Matériau utilisé : M 270-50A AMFCE (Même matériau que pour une machine rotatif)

**Le tube** est fait d'un matériau paramagnétique conducteur, car il faut que :

- Le flux transversal pénètre le tube sans que celui-ci soit canalisé sur sa géométrie.
- Le flux transversal doit pouvoir créer des courants induits dans le matériau d'où la nécessité qu'il soit conducteur électrique.

Matériau utilisé : Aluminium

**La culasse** et l'arbre central doivent être faits d'un matériau ferromagnétique car ils doivent canaliser et refermer le flux.

Matériau utilisé: XC 18

**Conducteur** : Fil multibrin en cuivre, section 2*1mm

## C. Elément de montage

### 1. Bobinage :

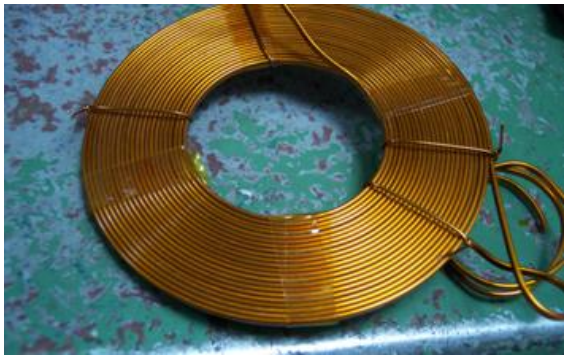


Figure VII-1

Une demie bobine est constitué de 22 spires

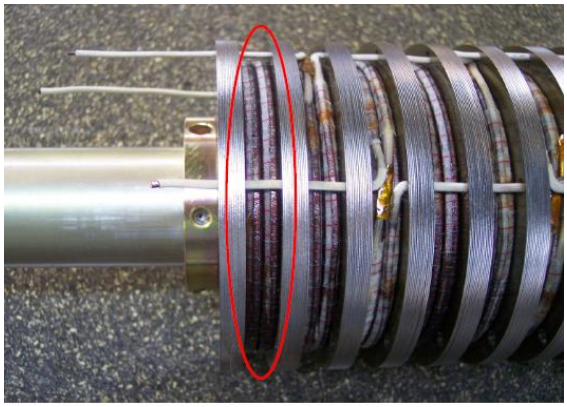


Figure VII-2

Chaque encoches (*15) est constituée de deux demie bobines assemblées entre elles (en conservant bien entendu le même sens d'enroulement pour ne pas annuler les ampères tours.)

## 2. Schéma de câblage des phases

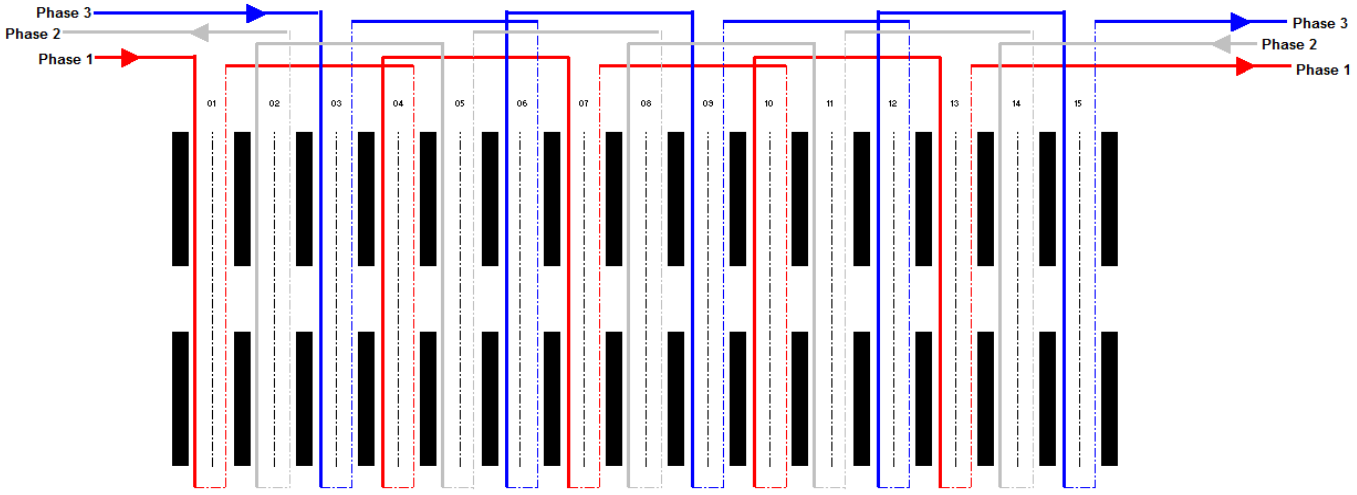



Figure VII-3

On comprend la nécessité d'inverser le sens du bobinage afin de créer alternativement des ampères tour positif et négatif.

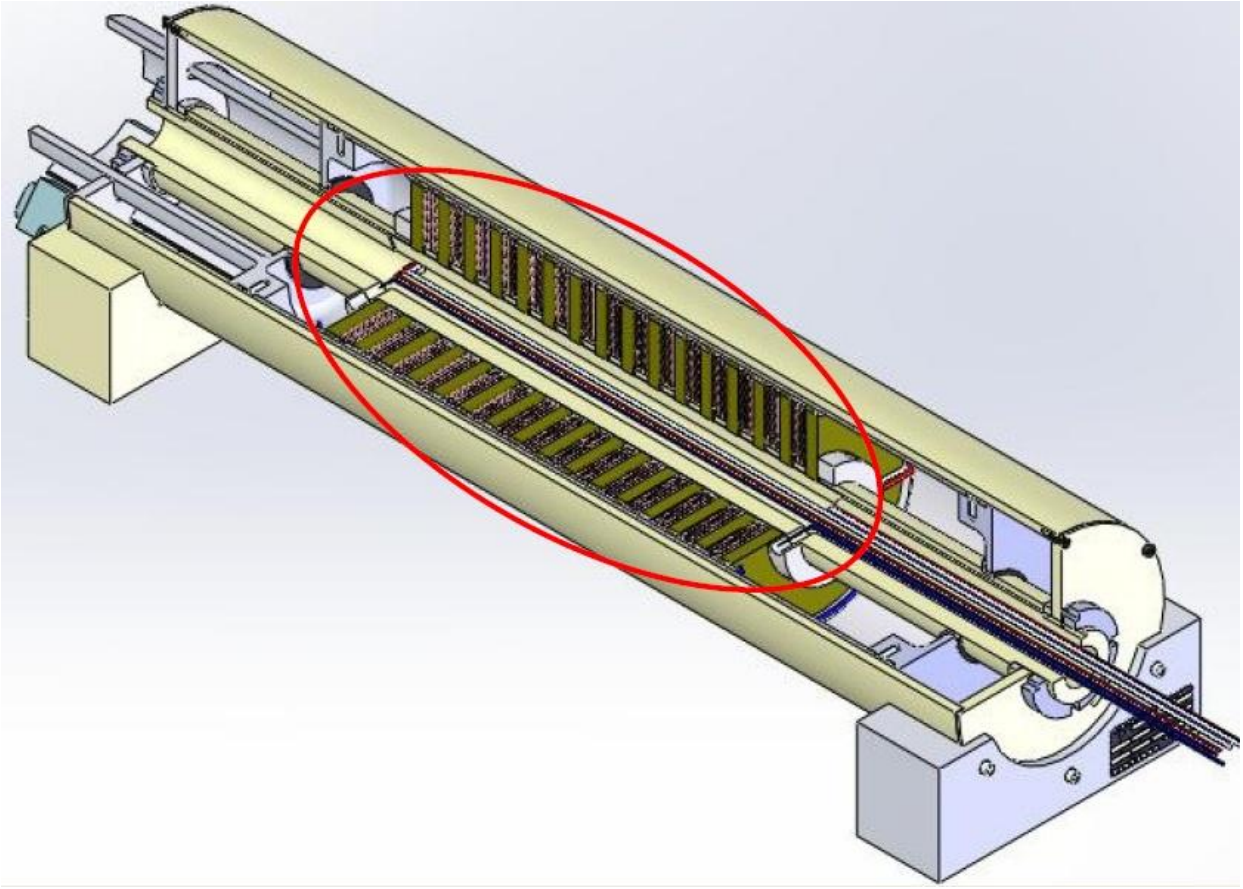


Figure VII-4

Vue en coupe de la machine

## D. Caractéristique du moteur

**Résistance et Inductance Rotorique :**

Rr théorique : 2,1 Ω (pour g=1) [D]

Lr estimé : 2,3 mH (Valeur estimé par rapport à une machine tournante)

**Resistance statorique : (mesurée avec une fréquence de 30 Hz)**

| | phase rouge | phase bleu | phase blanche |
|---|---|---|---|
| Rs (Ω) | 1,98 | 1,87 | 1,82 |

## 1. Phénomène de dissymétrie

Contrairement à une machine tournante qui ne présente pas de dissymétrie (pas de déséquilibre de phase), la machine linéaire présente des dissymétries liées aux effets d'extrémité qui se caractérisent en deux, à savoir :

- La phase qui se situe en extrémité par rapport au sens de déplacement du flux verra son inductance diminuée car le flux va se refermer dans l'air.
- Les mutuelles d'inductances ne sont pas identiques du fait que la surface en rapport avec le flux commun n'est pas identique entre phases [E].

**Inductance Statorique :**

(Le détail des mesures de la phase blanche se trouve en annexe):

| | phase rouge | phase bleu | phase blanche |
|---|---|---|---|
| ls (mH) | 39 | 35,7 | 35,9 |

**Mutuelle d'inductance statorique :**

Les mesures des mutuelles d'inductance entre les différentes phases ont été réalisées « doublement » en inter-changeant l'alimentation uniquement à titre d'expérimentation.

- Fréquence d'essai: 30 Hz

| | **phase alimentée** | **phase induite** | **phase induite** |
|---|---|---|---|
| | Rouge | Bleu | blanche |
| I (A) | 2,00 | | |
| U induit (V) | | 5,17 | 6,3 |
| m (sortie/entrée) | | 3,2 | 2,63 |
| inductance mutuelle (Henry) | | 0,0153 | 0,0167 |

| | **phase alimentée** | **phase induite** | **phase induite** |
|---|---|---|---|
| | Bleu | Rouge | Blanche |
| I (A) | 2,00 | | |
| U induit (V) | | 5,71 V | 6,26 V |
| m (sortie/entrée) | | 2,9 | 2,65 |
| inductance mutuelle (Henry) | | 0,0151 | 0,0166 |

| | phase alimentée | phase induite | phase induite |
|---|---|---|---|
| | Blanche | Bleu | Rouge |
| I (A) | 1,95 A | | |
| U induit (V) | | 6,18 V | 6,23 V |
| m (sortie/entrée) | | 2,68 | 2,66 |
| inductance mutuelle (Henry) | | 0,0168 | 0,0169 |

**Tableau VII.1**

Paramètres initiaux retenus :

Rs= 1,89 ohms

ls= 36 mH

Rr= 2.1 ohms

Lr= 0.0023Henry

Mutuelle stator/rotor = 0.014 [F]

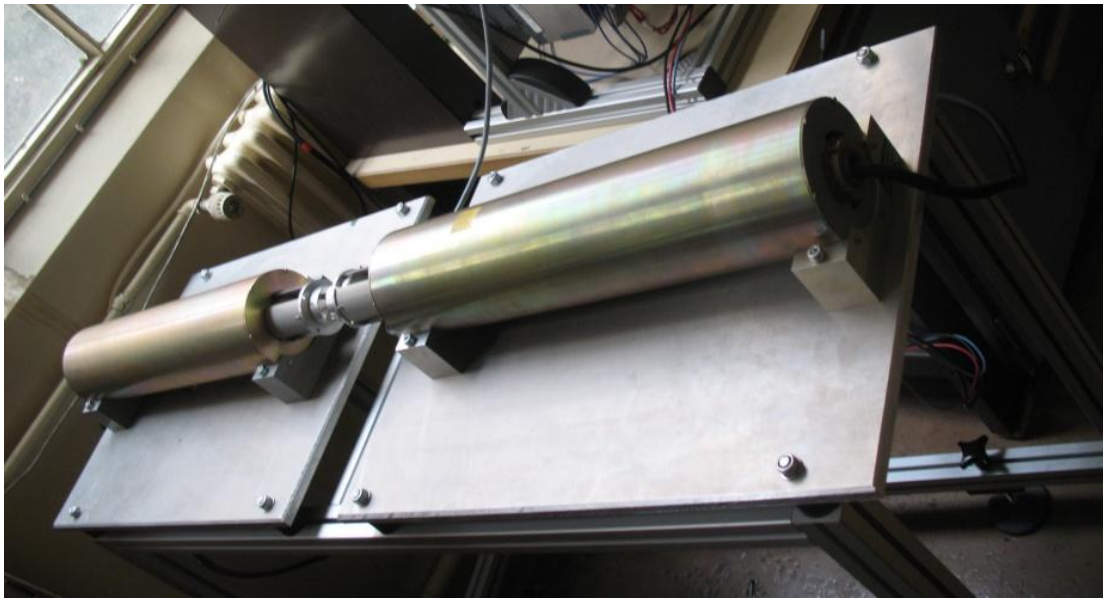


**Figure VII-5**

Photo des 2 moteurs linéaires montés sur le banc

# VIII. Analogie entre les grandeurs d'une machine tournante et d'une machine linéaire.

Outre le phénomène de dissymétrie mentionné précédemment qui différencie la machine linéaire d'une machine tournante, il convient à présent de définir l'ensemble des analogies.

## A. Machine tournante :

Dans une machine polyphasée tournante, les bobines rotoriques ou statoriques sont positionnées dans l'espace géométrique de manière uniforme.

- L'angle d'ouverture entre deux bobines de la même phase est de π.
- L'angle entre chacune d'elle est donné par la relation : —

p étant le nombre de paire de pôles et n le nombre de phases

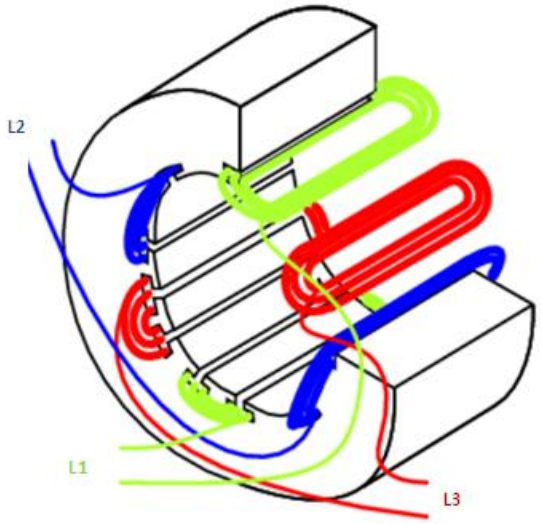



figure VIII-1

Enroulement statorique triphasé

Du point de vue des variables mécaniques, nous avons trois grandeurs existantes :

- L'angle du rotor : θr
- La vitesse de rotation du rotor :Ωr
- Le couple utile: Tu

Machine linéaire :

La distance entre deux bobines inductrices d'une même phase est appelée pas polaire. Elle se note τ (taux).

Du point de vue des variables mécaniques, nous avons trois grandeurs existantes :

- La position du mover: x
- La vitesse linéaire du battement (vitesse d'aller-retour): vs en m/s
- La force de déplacement: F

## B. Comparatif des grandeurs linéaires/ tournantes

La figure VIII-2 est une vue en coupe de la MASL mettant en évidence les caractéristiques mentionnées.

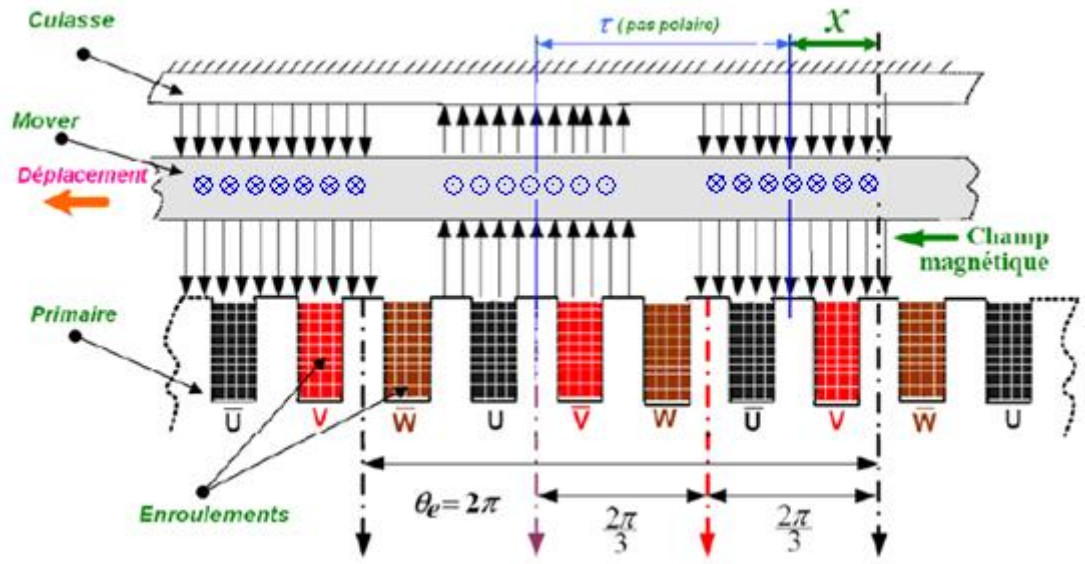



Figure VIII-2

Vue en coupe de la machine linéaire avec la représentation du champ d'induction

Tableau comparatif des grandeurs mécaniques et électriques :

| | Moteur rotatif | Moteur linéaire |
|---|---|---|
| **Position mécanique** | Θméca | x |
| **Position électrique** | θélec=P*Θméca | (π/τ)*x=Np*x |
| **Force/Couple** | Couple | Force longitudinale |
| **Ouverture entre deux bobines d'une même phase** | π | τ |

Tableau VIII.1

### C. Schéma équivalent de la Machine asynchrone linéaire :

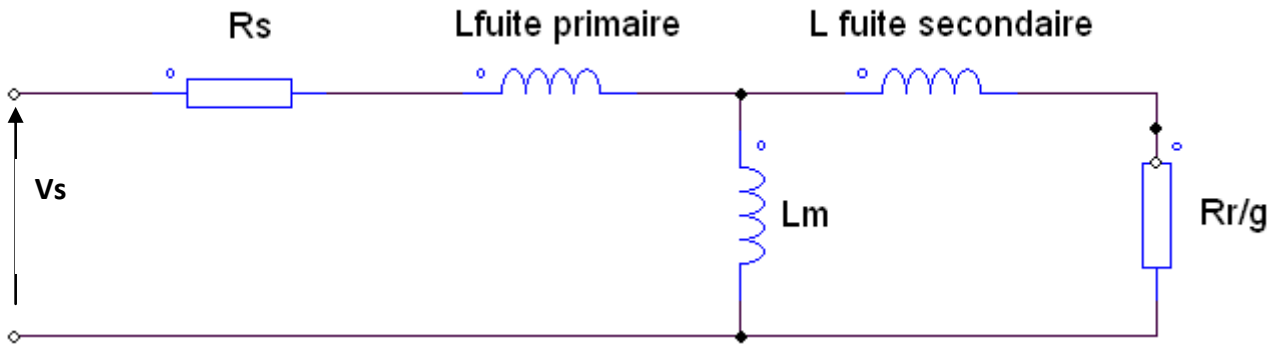



Figure VIII-3

Schéma monophasé équivalent de la MASL, les effets d'extrémités ne sont pas pris en compte, ainsi que les pertes fer.

Le schéma équivalent ci-dessus ne tient pas compte de la dissymétrie du champ magnétique des trois enroulements du primaire due aux effets d'extrémités [H].

La définition du glissement reste bien entendu la même pour une machine linéaire, mais étant donné que le régime dynamique n'est jamais stable du fait mouvement alterné, g variera constamment.

# IX. Physique du déplacement du mover

### A. Repérage préalable des phases

Il faut impérativement repérer les phases pour que l'ensemble du flux sur les pas polaires soit cohérent. La façon d'opérer est identique à la mesure des mutuelles d'inductances.

On alimente une phase de la machine avec une tension alternative. Le courant variable dans la phase en question va produire un flux variable qui va traverser les bobines des deux autres phases et ainsi produire une fem induite.

C'est le sens de la fem induite par rapport à la fem d'alimentation qui va déterminer les fils à connecter pour créer le point étoile.

Le schéma électrique de la figure IX-1 montre le sens de la tension induite dans le bobinage noir ainsi que les lignes de flux.

Si la tension induite est en opposition par rapport à la tension d'alimentation. Le potentiel bas est à connecter en étoile. Si le cas inverse se présente, c'est le potentiel haut à connecter en étoile

Le repérage de la troisième phase est identique.

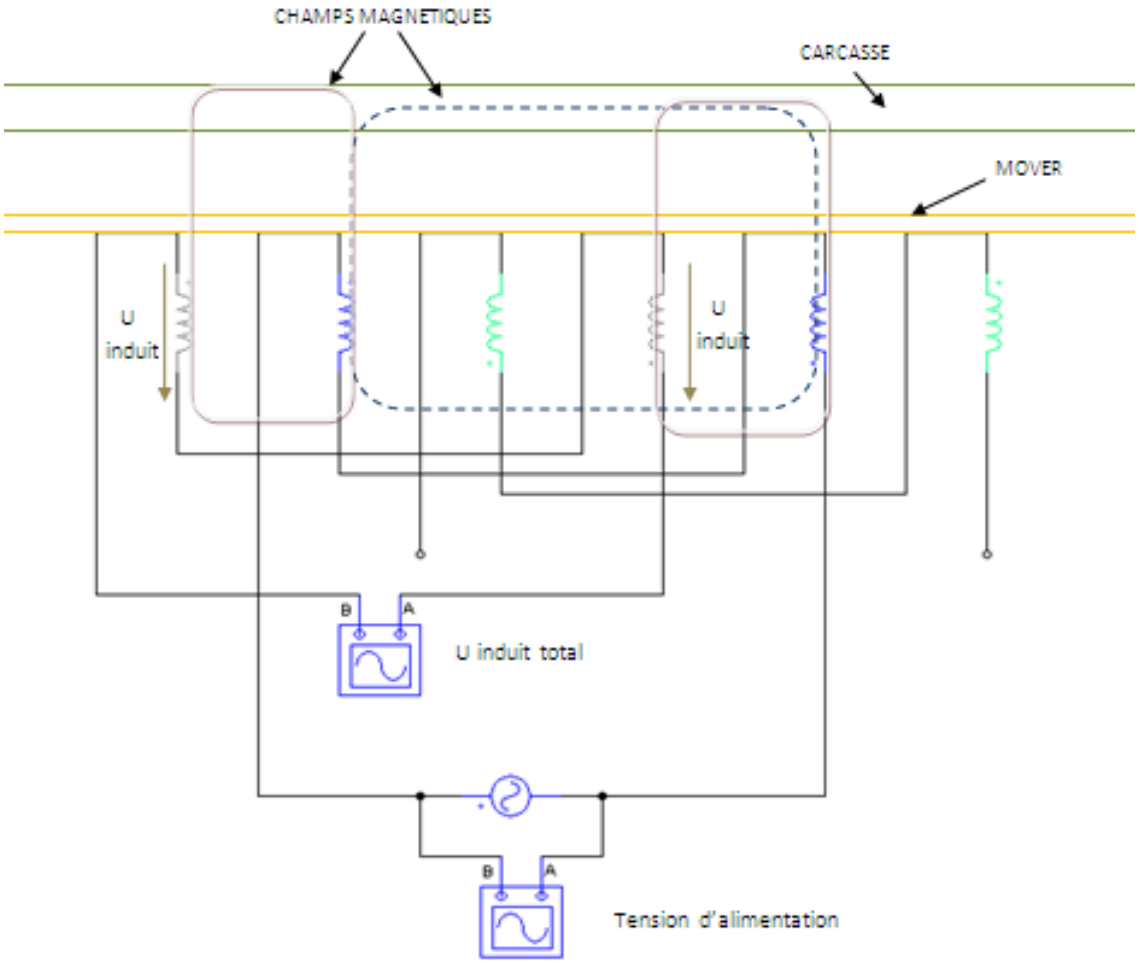



**Figure IX-1**

Synoptique du montage pour le repérage des phases. On alimente une phase parmi les trois et on observe sur les 2 autres phases la fem induite.

### B. Déplacement du champ magnétique

Afin d'étudier le sens de déplacement du champ magnétique (donc de la force), il convient de séquencer les tensions d'alimentations suivant leurs changements de signes.

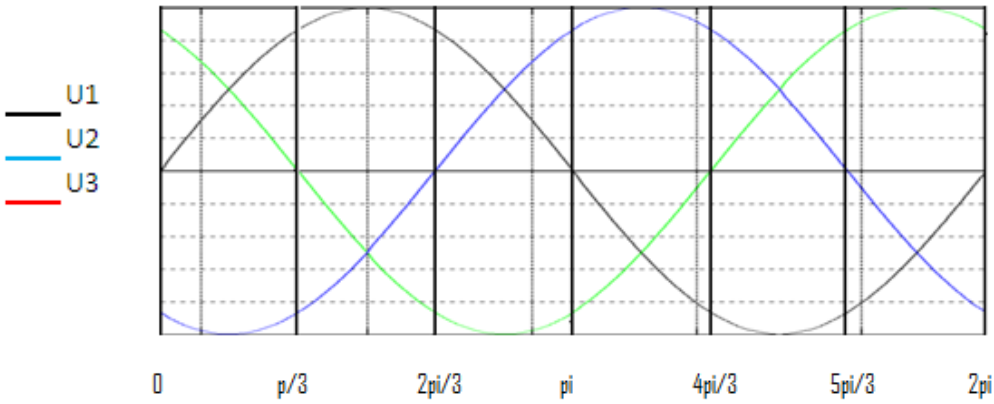



Figure IX-2 : Représentation des tensions triphasées

Pour des soucis de clarté, il ne sera représenté sur les trois figures suivantes qu'un ensemble de 3 pas polaires. Le champ magnétique est caractérisé par des flèches.

Séquence 1 (0 à π/3):

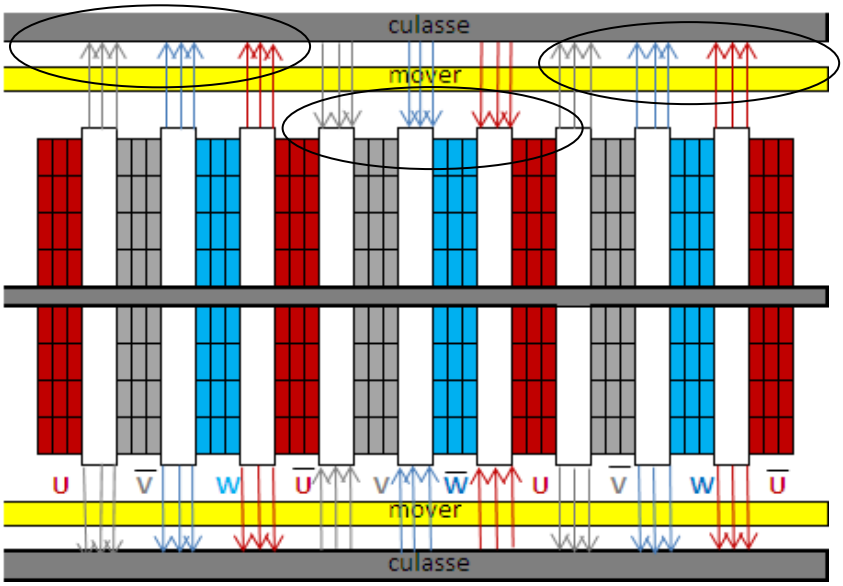



Figure IX-2 1

Séquence 2 (π/3 à 2π/3):

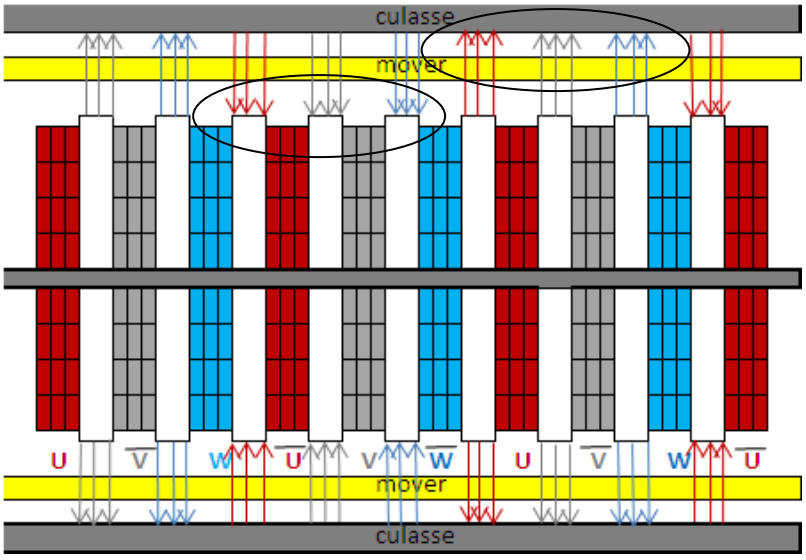



Figure IX-2 2

Séquence 3 (2π/3 à π):

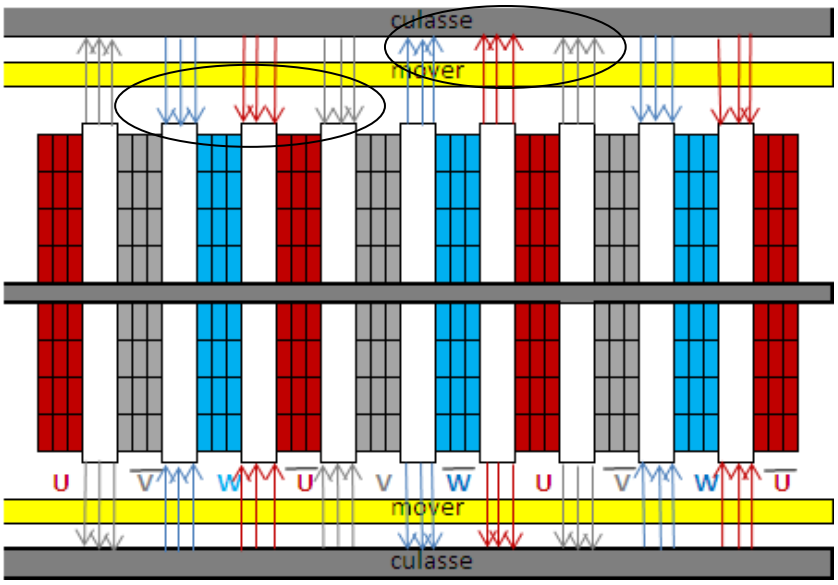



Figure IX-2 3

Le champ translate de la droite vers la gauche sur toute la demi-période. Il en est de même pour les séquences 4,5 et 6. A la Septième (2pi+pi/3) on aura la configuration inverse de la séquence 1.

La force de Laplace étant dirigée dans le même sens que le déplacement du champ, le mover se déplace donc de la droite vers la gauche.

## 1. Inversion du sens de déplacement :

Pour inverser le sens de la force, il faut que le champ se déplace de la gauche vers la droite. On comprend la nécessité de recourir à un pilotage spécifique d'un variateur pour effectuer ce procédé en grande dynamique (30 allers-retours par seconde).

# X. Matériel pour piloter les deux machines :

## A. B&R Automation

Parmi les différents variateurs existant sur le marché français (Leroy somer, Emerson, B&R Automation, Schneider Electric…) nous devons choisir un variateur permettant de contrôler l'axe suivant un profil de déplacement spécifique.

Nous avons opté pour la société B&R qui a été la seule a indiquer que son matériel pouvait répondre à notre cahier des charges, de plus cette société possède une antenne qui se trouve en Seine et Marne.

La société **B&R** (Bernecker + Rainer Industrie-Elektronik Ges.m.b.H.) est une société autrichienne fondée en 1979 par Erwin Bernecker et Josef Rainer en Autriche.

B&R conçoit, fabrique et commercialise des produits et solutions d'automatisation à destination des constructeurs de machines, des intégrateurs et des exploitants d'installations ou utilisateurs finaux. Ses services comprennent la formation, le développement d'applications, le support technique et le service après-vente.

## B. Matériels utilisés pour le banc d'essai

La commande du variateur se fera à l'aide d'Automation Studio qui est le logiciel de programmation (version SP 06).

Le matériel mis en œuvre est décrit ci dessous, à noter que le matériel nécessite un schéma de liaison à la terre de type IT ou alors un schéma TT avec un disjoncteur différentiel 300 mA immunisé. Ceci est nécessaire du fait du fort appel de courant homopolaire à la mise sous tension.

- DISJONCTEUR DIFFERENTIEL : Le disjoncteur doit être triphasé de type HP 300 mA. Un Relevé des caractéristiques des courants homopolaire à la mise sous tension se situe en annexe D.
- FILTER : Le but de ce filtre est de filtrer les harmoniques que renvoie l'ensemble du matériel sur le réseau.
- REGENERATION CHOKE : Bobines couplées pour permettre de renvoyer de l'énergie au réseau.
- POWER SUPPLY (rectifier) : Fonctionne en mode redresseur (380V AC en 750V DC).
- AUXILIARY SUPPLY : Bloc contenant un convertisseur 750V-24V pour l'alimentation des auxiliaires (Dans notre cas on alimente le power panel)
- INVERTER : Onduleur/variateur connecté au moteur
- POWER PANEL : Ecran d'interface homme machine et automate intégré
- MEMORY CARD : Contient les informations sur le programme et surtout sur la désignation des éléments qui lui sont connectés
- PC: Pc servant à se connecter à l'automate (transfert/modification du programme, visualisation des entrés - sorties en temps réel)

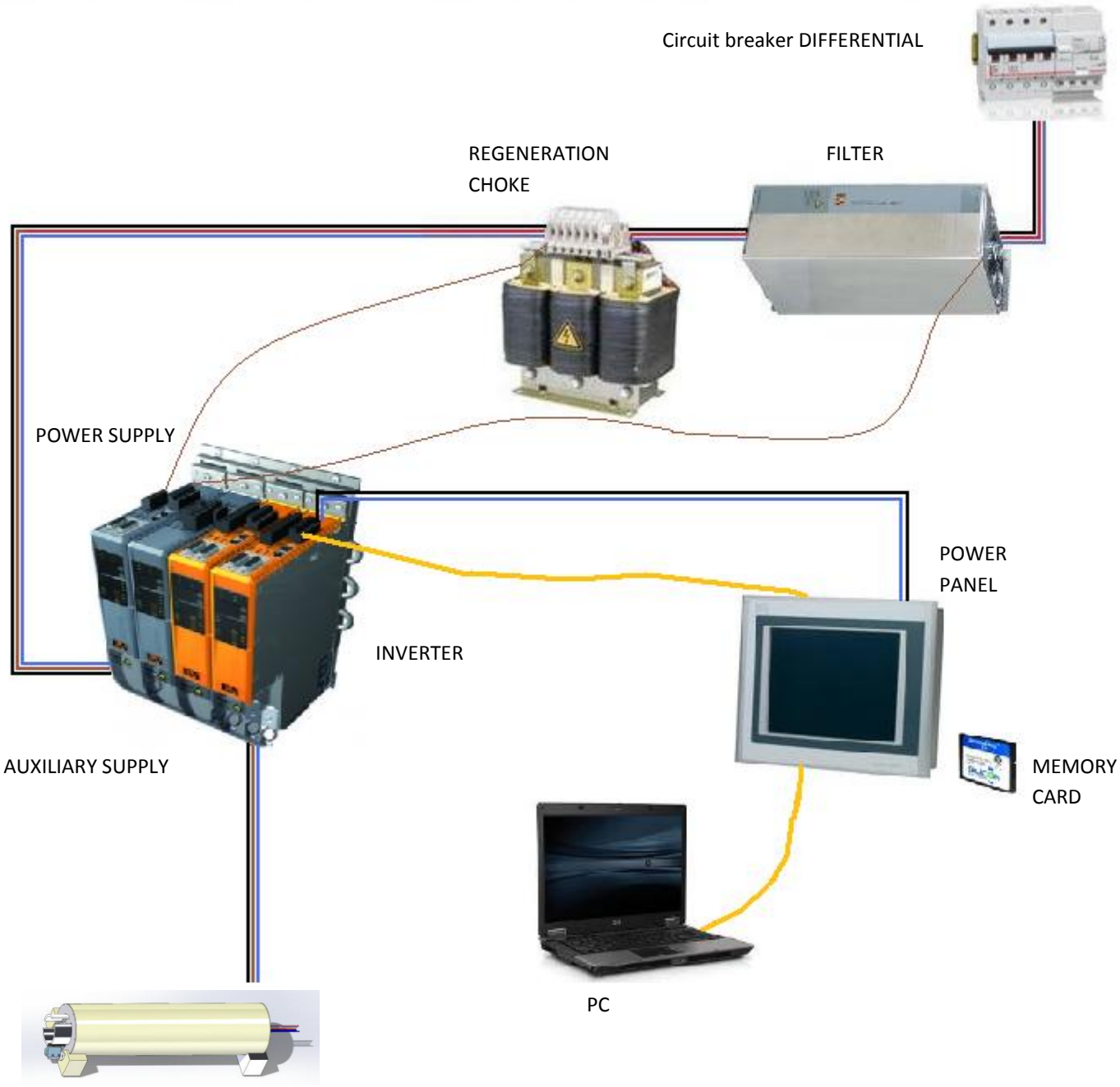



**Figure X-1**

Représentation de l'ensemble du montage

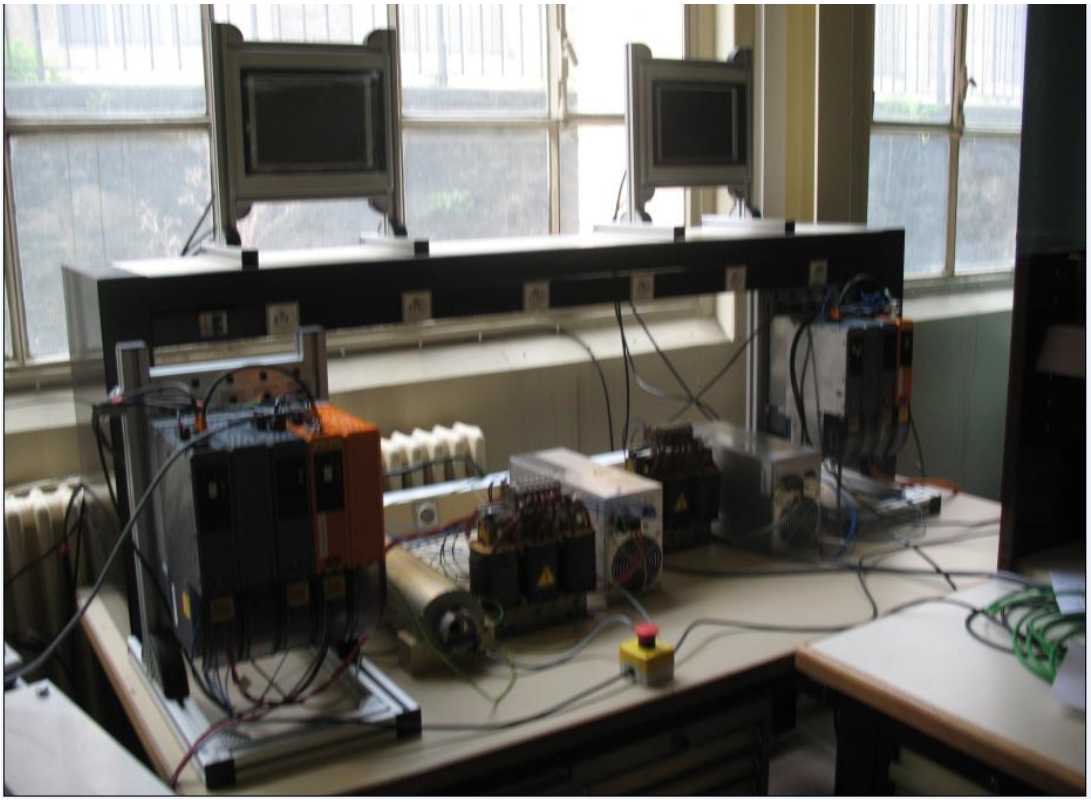


**Figure X-2**

Photo de l'ensemble du montage du contrôle commande avec un seul moteur linéaire en fonction au centre de la table.

## C. Synoptique électrique du matériel :

Le moteur absorbe de la puissance électrique réactive et active tandis que la génératrice fournie de la puissance électrique active et absorbe de la puissance réactive.

Le schéma ci-dessous représente les convertisseurs de puissance et le sens du transit des puissances.

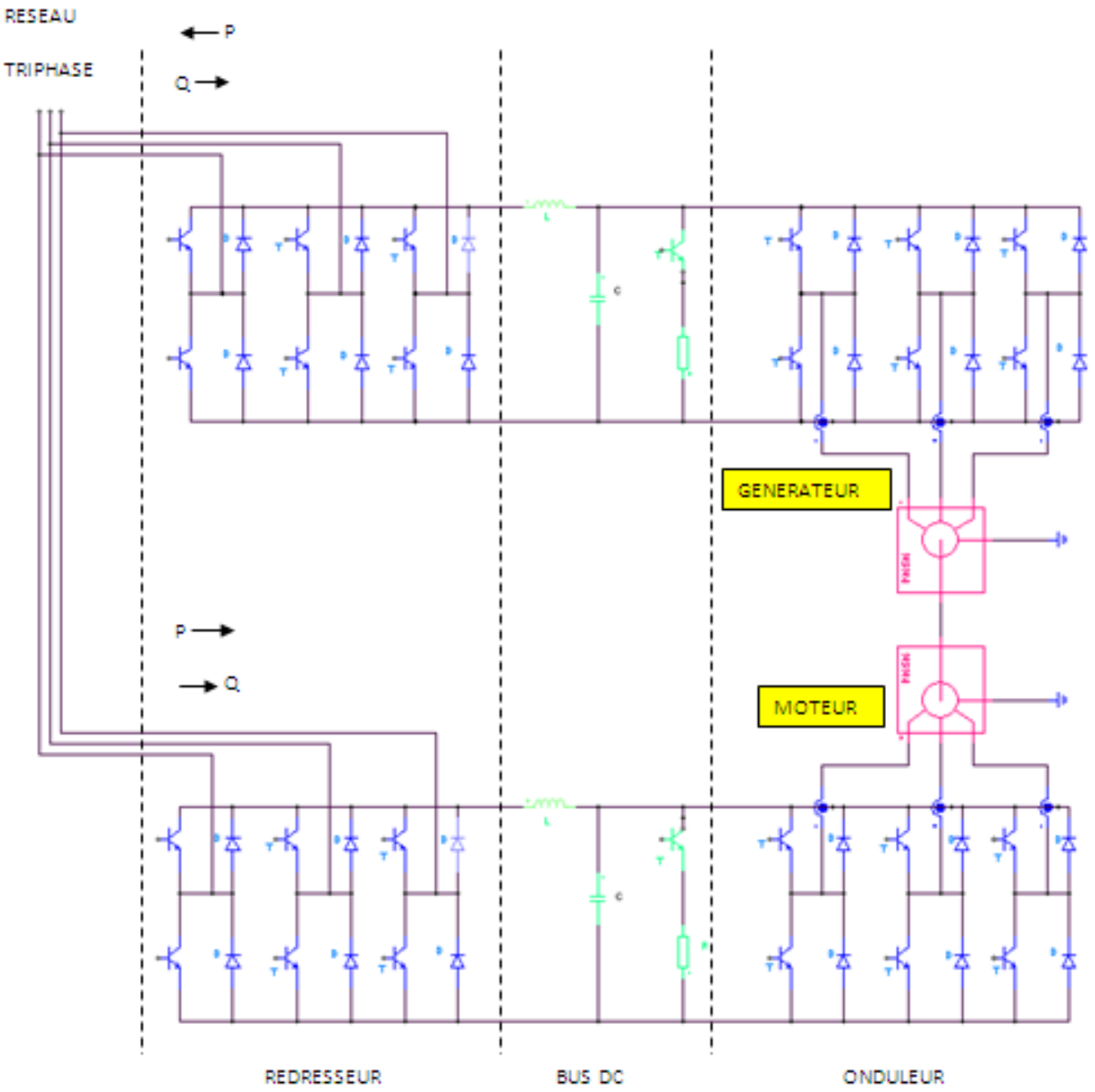



Figure X-3

Schéma électrique des modules.

# XI. Généralité sur la configuration des axes moteurs

## A. Première configuration :

Dans certaines applications, il est possible que l'on veuille qu'un ensemble de moteurs sur une chaine d'assemblage suivent un seul et même profil (une cadence par exemple) d'un autre moteur jouant ainsi l'office de « moteur maitre ».

**1er exemple :**

Sur la figure XI-1 est représenté le synoptique d'une découpe de tôle. Bien que le moteur qui déroule le rouleau de manière continue, des irrégularités peuvent exister suite au déroulement de la bobine. Ainsi, afin de maintenir les dimensions de coupe, une temporisation sur le moteur de coupe ne serait donc pas optimale.

Il est préférable dans ce type de procédé que le moteur effectuant la coupe (esclave) dépend de la quantité déroulé par le moteur dérouleur (maitre). Ainsi la précision est assurée.

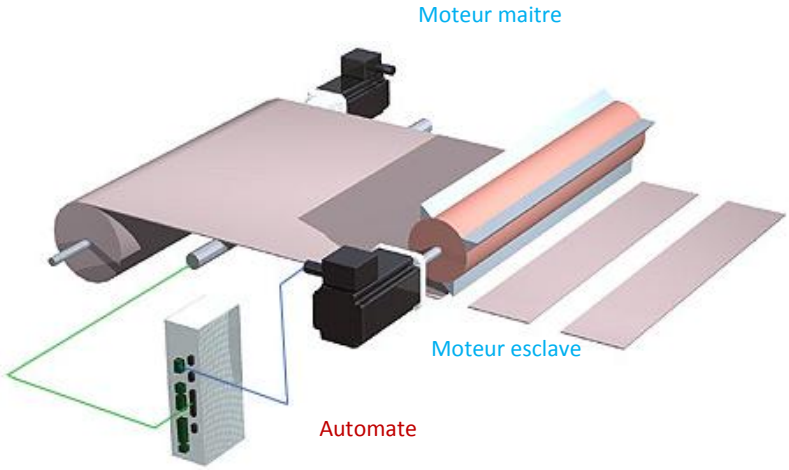



**Figure XI-1**

**2 ême exemple:**

L'exemple du pont de levage illustre également la nécessité du principe d'axe maitre / esclave. En effet, lors de l'avancement du pont, les deux portiques doivent avancer à la même vitesse, sous peine d'exercer des contraintes mécaniques sur l'ossature. Il suffira ainsi de configurer n'importe quel moteur en tant que maitre et les autres en tant qu'esclaves. (Figure XI-2).

Figure XI-2

## B. Deuxième configuration possible

Une deuxième configuration possible est de synchroniser un ou plusieurs moteurs par rapport à un profil de position, appelé « Cam ».

Définition d'une Cam:

- Graphique dont l'ordonné correspond à l'axe esclave qui est réel et dont l'abscisse correspond à l'axe maitre qui peut être réel ou virtuel. Le concept de virtuel signifie que l'axe est défini « informatiquement ». Les unités des deux axes ne sont pas définies. Elles peuvent donc représenter une position en degré ou en mètre selon l'application utilisée.

# XII. Logiciel de programmation de l'automate

La structure de programmation des automates est basée sur un code commun appelé PLC (API en français)

## A. Les « views » dans Automation Studio

Dans Automation Studio on trouvera trois fenêtres principales dans lesquelles se trouve(nt) le(s)programme(s) et toutes les configurations nécessaires telles que les paramètres et le matériel utilisé.

### 1) La vue logique

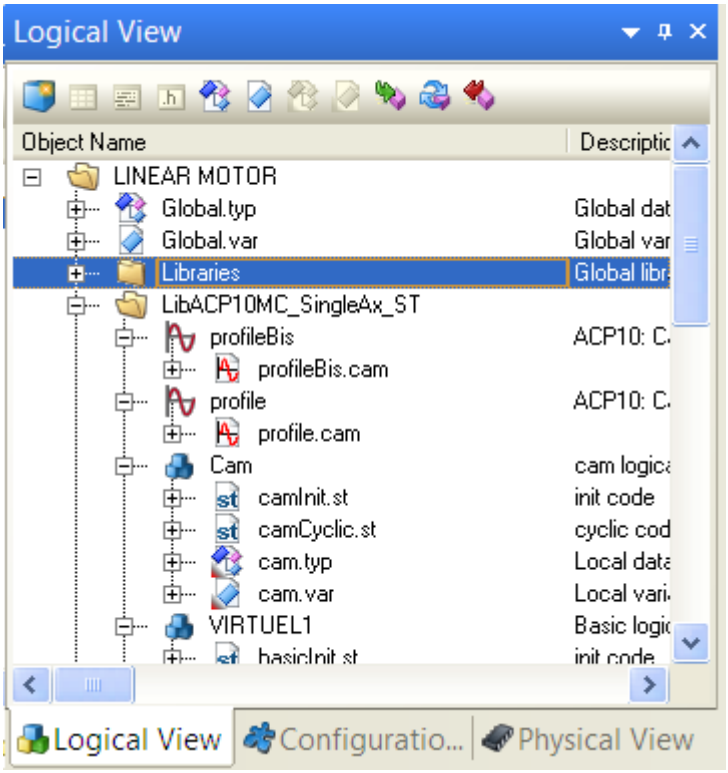



**Figure XII-1**

Copie d'écran sur Automation Studio de la Logical View

**Logical View :**

On trouvera dans cette fenêtre :

1° Le nom du programme général

2° Les variables - Global.var ou Global.typ

3° Les Librairies de la PLC open contenant toutes les fonctions dont on peut faire appel

4°

- La déclaration des packages
- les data Objects,
- Les variables locales. (var et type)
- Les programmes

2) La vue de Configuration

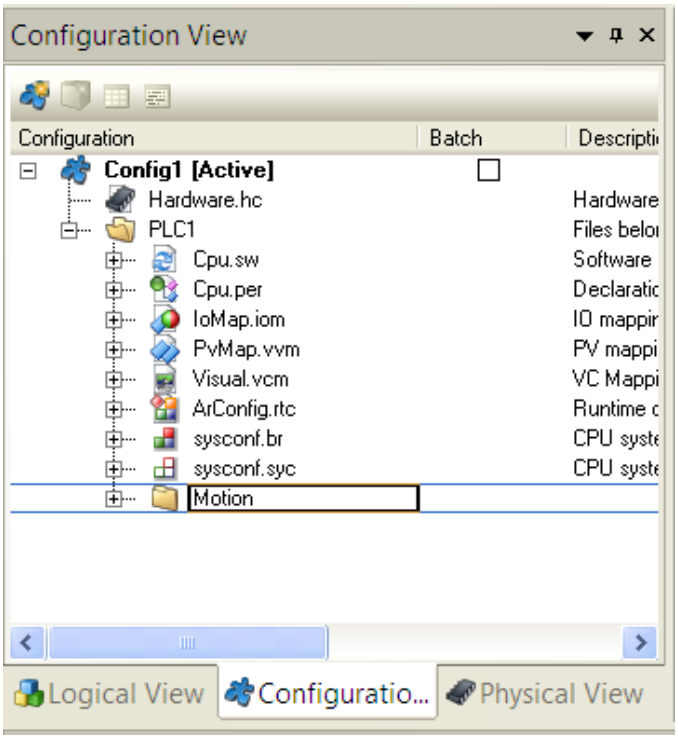



Figure XII-2

Copie d'écran sur Automation Studio de la Configuration View

**Configuration View :**

On trouvera dans cette fenêtre :

1° Les noms de la configuration faisant interface entre la vue logique et la vue physique.

Il existe deux types de configuration, (Une seule peut être active à la fois.)

- Différentes configurations faisant référence aux « hards » (matériel physique déclaré)
- Une configuration utilisée pour simuler le programme sur le PC (mode ARsim)

2) La vue Physique

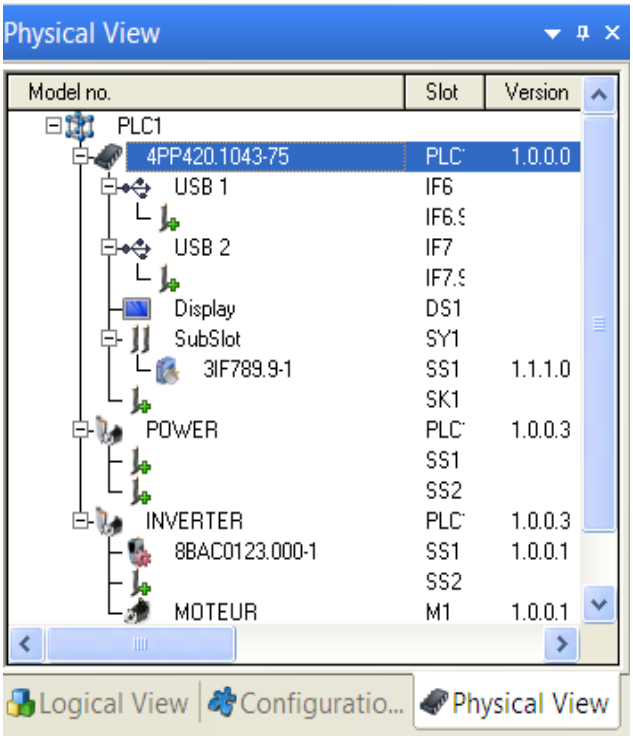



Figure XII-3

Copie d'écran sur Automation Studio de la Physical View

**Physical View :**

On trouvera dans cette fenêtre :

Tout le matériel déclaré contenu dans la vue active. La liste se présente sous forme d'arborescence

(L'ordre de déclaration n'a d'importance que si la connexion est de type série)

## B. Les librairies de la logical View :

Dans la librairie d'automation Studio, il y a neuf dossiers existants regroupant l'ensemble des blocs de fonctions utilisables dans un programme :

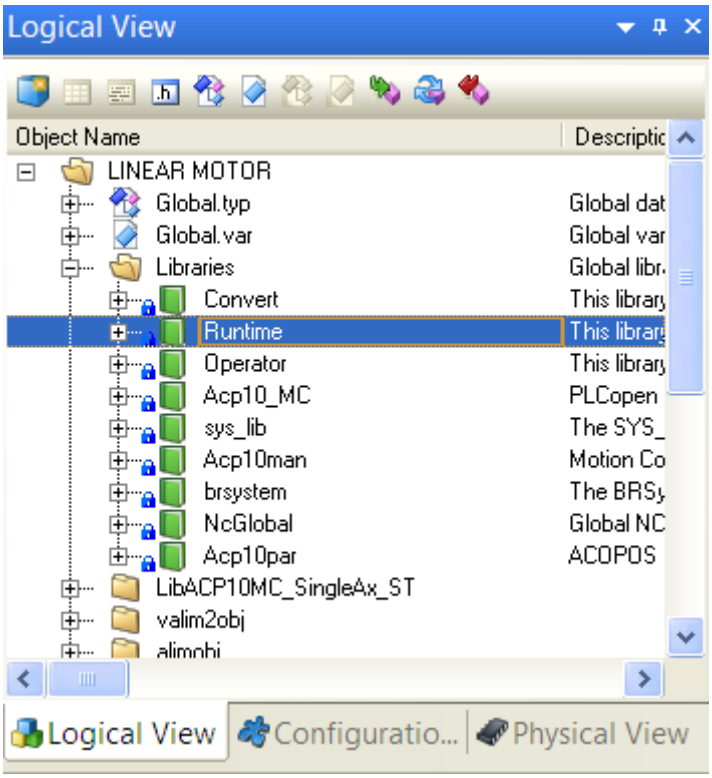



**Figure XII-4**

Copie d'écran sur Automation Studio de la Logical View

Classification des dossiers de la librairie d'Automation Studio :

| | Contenu | Exemple |
|---|---|---|
| Convert | Permet de convertir une entité en une autre entité | LREAL_TO_SINT<br>BOOL_TO_DT |
| Runtime | | |
| Operator | permet de faire tout types d'opérations entre les variables | TAN<br>DIV |
| Acp10_MC | blocs de fonctions applicable au control d'axe | MC_Stop<br>MC_ReadActualVelocity |
| sys_lib | Contient deux blocs de fonctions pour la conversion de bits. Les autres fonctions sont relative à la mémoire à la gestion du temps… | Byte2Bit<br>DA_read |
| Acp10man | Paramètrage des axes et des boucles de régulation des controllers | t_filter,kv |
| brsystem | fonction relatives à l'affichage de la mémoire disponible, de la pile automate… | |
| NcGlobal | Contient des fonctions NC général et constant | |
| Acp10par | Relatif aux paramètres existant avec indication des Par ID | DC bus,Positive switch End |

**Tableau XII.1**

## C. Déclaration et appel des variables dans un programme :

La figure XII.5 illustre le processus de déclaration

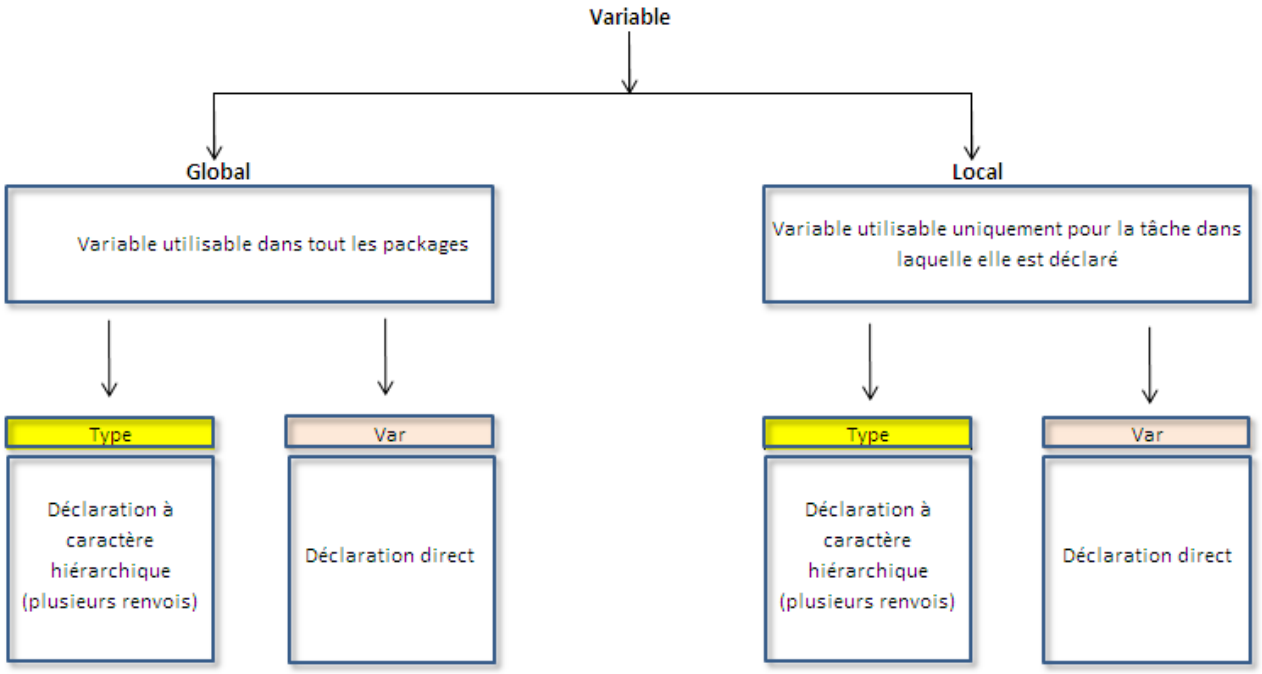



**Figure XII-5**

Représentation schématique de la déclaration des variables

Tous les variables déclarées en point « Typ » doivent être déclarées dans un premier temps par une variable de type point « Var ».

Exemple d'appel de la variable MechanicalPower :

CamControl.meca.MechanicalPower

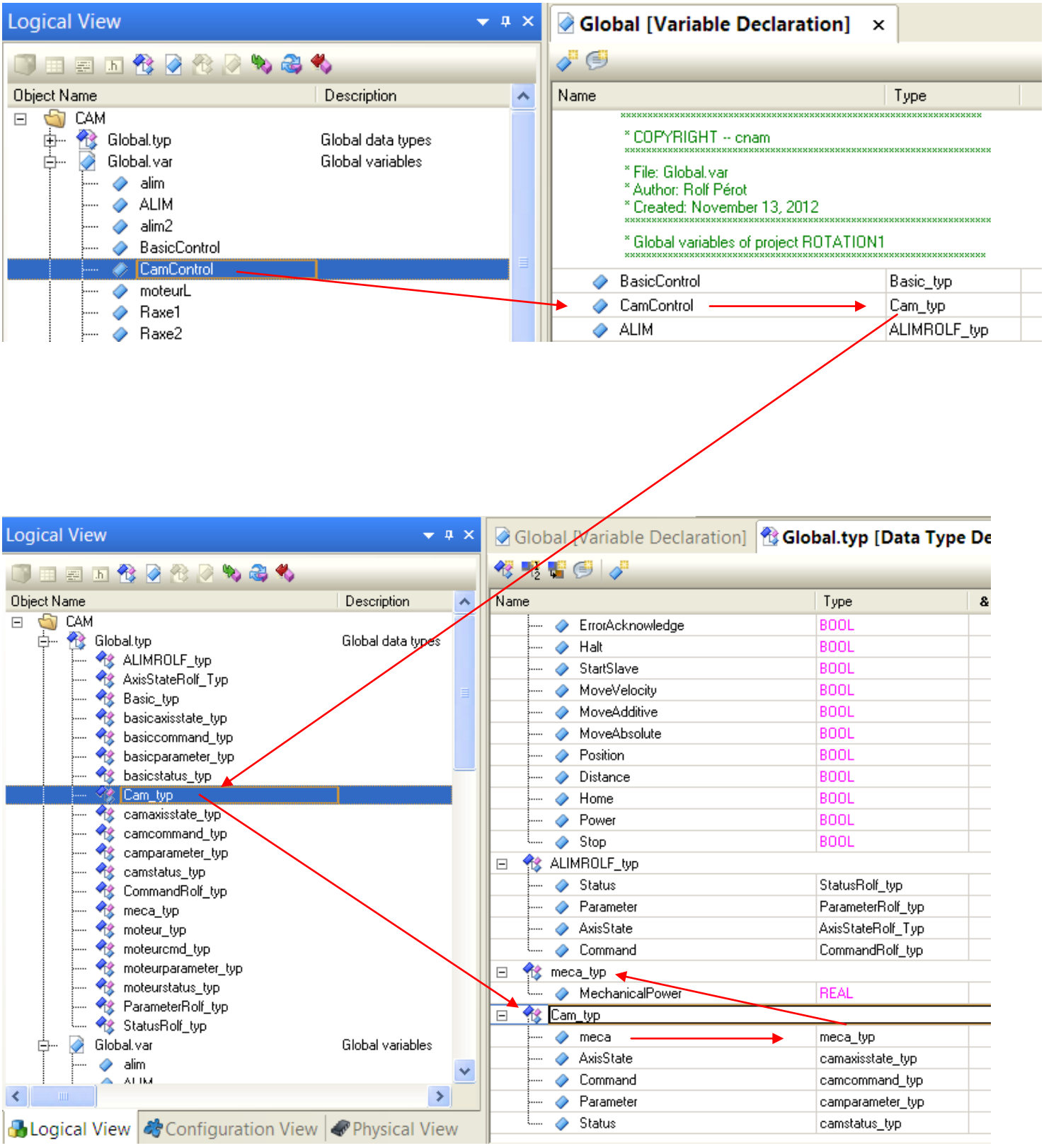



Figure XII-6

Représentation schématique de la déclaration et de l'appel de la variable Mechanical power par copie d'écran.

La variable « Typ » est une variable intermédiaire, elle permet de classifier et de structurer l'ensemble des variables

La figure ci-dessous explique la structure de déclaration est d'appel des variables en point « Typ ».

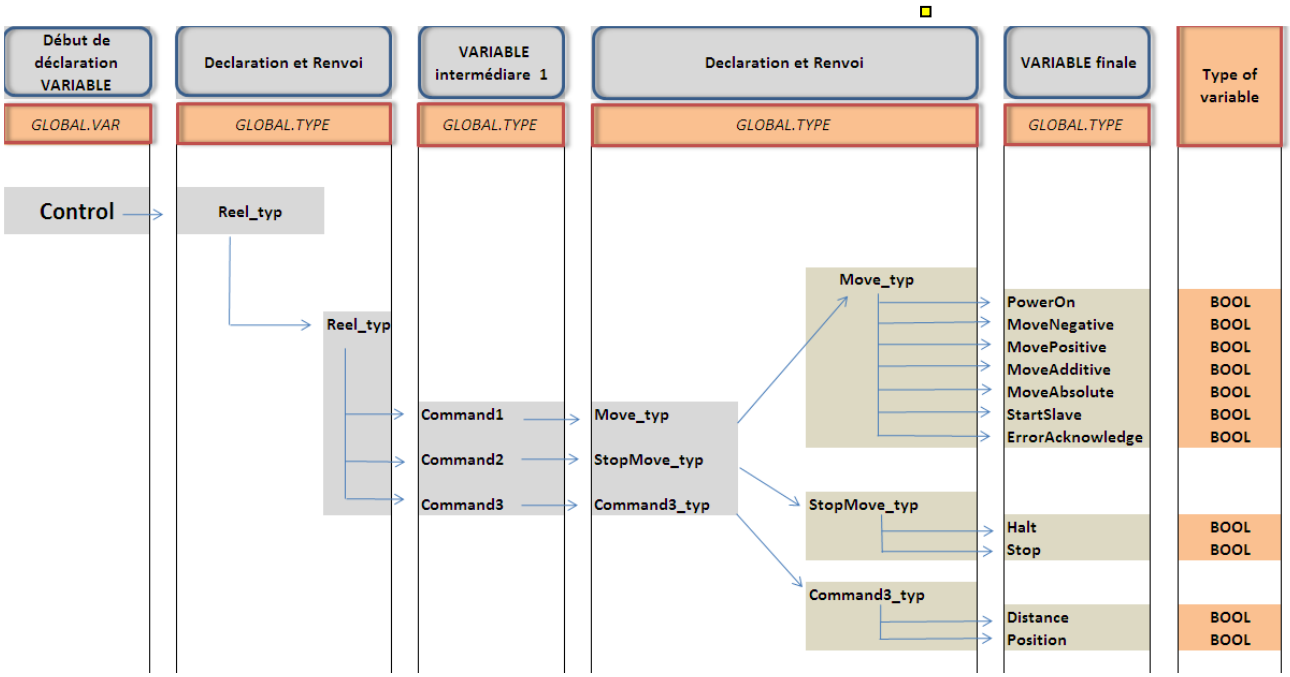



**Figure XII-7**

Représentation architecturale de déclaration et d'appel des variables

## D. Principe des blocs de la librairie Motion et de leurs utilisations:

Bloc de fonction MC_Power :

Ce bloc a pour but de mettre sur ON/OFF un axe quelconque, virtuel ou réel (Figure XII-8).

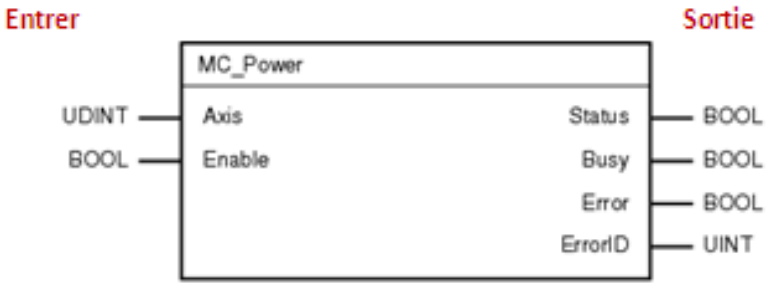



Figure XII-8

Description des entrées et des sorties du bloc de fonction MC_Power :

| Entrée/Sortie | Paramétre | Type de variable | Signification |
|---|---|---|---|
| **E** | Axis | UDINT | Axe à référencer |
| **E** | Enable | BOOL | Mise sur OFF ou sur ON du bloc |
| **S** | Status | BOOL | Bloc ON ou OFF |
| **S** | Busy | BOOL | Le bloc de fonction en en process |
| **S** | Error | BOOL | Une erreur est présente au sein du bloc |
| **S** | ErrorID | UINT | Numéro de l'erreur |

**Tableau XII.2**

Entrés /Sortie du bloc de fonction MC_Power

Déclaration du bloc : Le bloc ne peut pas avoir le même nom que l'entité, il faut lui attribuer un autre nom (de préférence similaire pour facilité la lecture du programme).

| Name | Type | & Reference | Constant | Retain | Value | Description [1] |
|---|---|---|---|---|---|---|
| MC_MoveAdditive_0 | MC_MoveAdditive | ☐ | ☐ | ☐ | | Function Block |
| MC_MoveVelocity_0 | MC_MoveVelocity | ☐ | ☐ | ☐ | | Function Block |
| MC_Power_0 | MC_Power | ☐ | ☐ | ☐ | | Function Block |

figure XII-9

Copie d'écran de la déclaration du bloc de fonction MC_Power

L'exploitation des entrés/sorties d'un bloc de n'importe quelles librairies de la motion, doit être structurée hiérarchiquement.

Il est possible d'utiliser de plusieurs langages de programmation :

- Structure and Text
- Ladder
- B&R Automation Basic
- Instruction List
- Function Bloc Diagram
- Continuous Function Chart

## E. Mise en place d'une « Cam »

Dans le reste de ce chapitre il sera uniquement question d'axe maitre virtuel et d'axe esclave réel.

### 1. Procédure générale pour la mise en place d'une Cam pour un axe de moteur tournant:

#### *a) Définir la résolution de l'axe réel*

| | | | |
|---|---|---|---|
| scaling | | | Scaling |
| load | | | Load |
| units | 360 | Units | Units at the load |
| rev_motor | 1 | | Motor revolutions |

Figure XII-10

Copie d'écran d'une partie de la configuration de l'axe réel

Nous avons défini ici que, 360 unités équivalent à 1 tours. Soit une résolution de 360 unités par tour.

#### *b) Définir la vitesse de l'axe maitre*

Etant donné que l'on se cantonne à un axe maitre virtuel, on peut attribuer comme unité ce qui a le plus de sens physique. Soit, par la logique, on s'adonnera la même unité que sur l'axe réel : des degrés.

L'unité de l'axe maitre  se définie  dans le programme correspondant au virtuel.

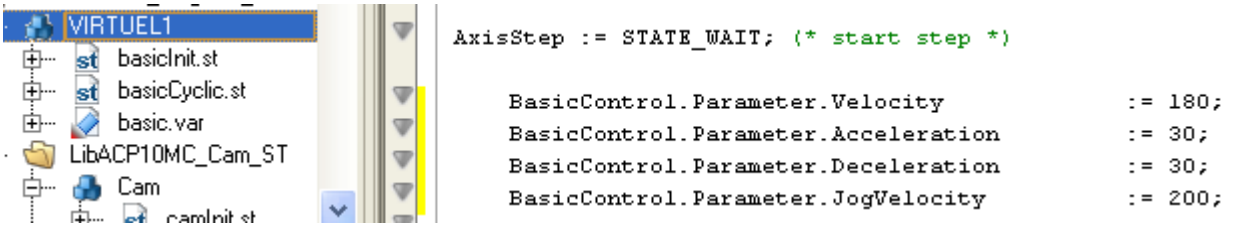



Figure XII-11

Copie d'écran d'une partie du programme initial de la tâche virtuel

Vitesse : 180 unités par seconde

Accélération effectué au démarrage : 30 unités par seconde²
Décélération effectué au démarrage : 30 unités par seconde²

### *c) Définir le profil de position souhaité et les unités*

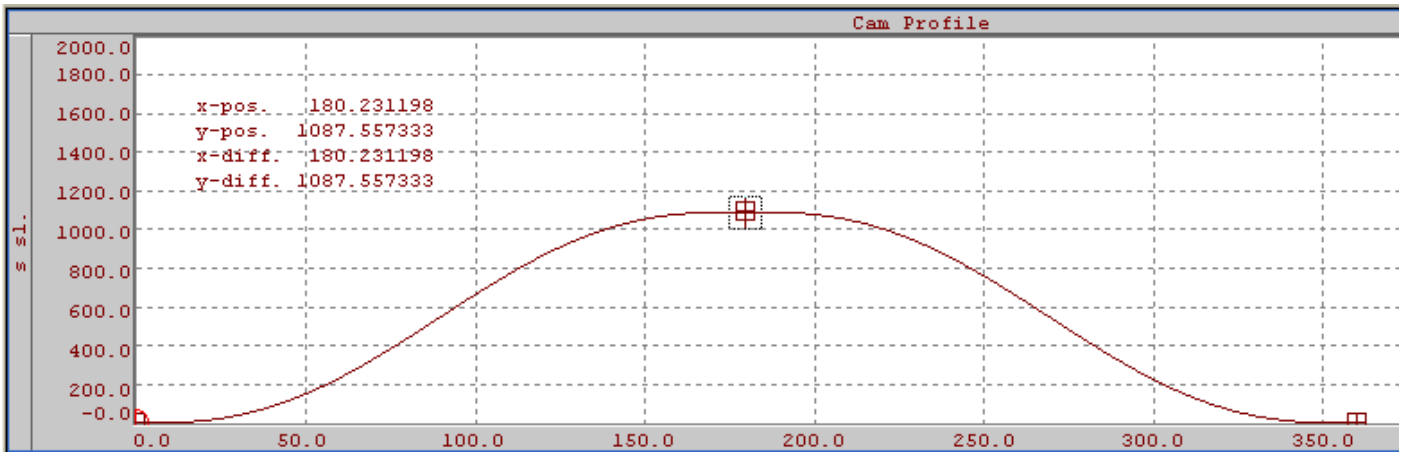



figure XII-12

Profil de Cam permettant de faire un aller retour sur un axe

La vitesse de l'axe maitre étant de 180 unités par seconde, on aura donc sur une période de fonctionnement deux tours effectués par ce dernier.

Lorsque le maitre aura atteint 180 unités (1s) l'axe réel aura fait 1080 unités, soit trois tours. Le profil fait que l'axe réel tourne ensuite dans l'autre sens ce qui ramène l'axe réel à sa position d'origine.

Le bloc de fonction MC_CamIn permet de renseigner et paramétrer l'axe réel et l'axe virtuel et d'effectuer la synchronisation des deux.

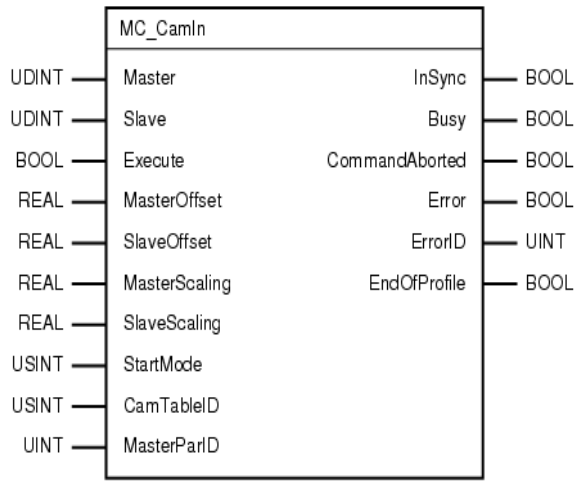



**Figure XII-13**

Bloc de fonction MC_CamIn

Paramétrage du bloc MC_CamIn :

**Parameters**

| Class | I/O | Parameter | Data Type | Description |
|---|---|---|---|---|
| B | IN | Master | UDINT | The master axis reference handle. |
| B | IN | Slave | UDINT | The slave axis reference handle. |
| B | IN | Execute | BOOL | Start at rising edge. |
| E | IN | MasterOffset | REAL | Offset on master axis [Units of master]. |
| E | IN | SlaveOffset | REAL | Offset on slave axis [Units of slave]. |
| E | IN | MasterScaling | REAL | Factor for the master profile. |
| E | IN | SlaveScaling | REAL | Factor for the slave.profile. |
| E | IN | StartMode | USINT | Start mode:<br>mcABSOLUTE ... 0<br>mcRELATIVE ... 1<br>mcDIRECT ..... 7 |
| E | IN | CamTableID | USINT | Identifier of cam table, output of MC_CamTableSelect.<br>Predefined 1:1 cam profile (master and slave length = 1)<br>mcLINEAR_CAM_PERIODIC<br>mcLINEAR_CAM_NON_PERIODIC |
| V | IN | MasterParID | UINT | Use this ParID from the master axis in stead of the set position, 0...use the set position. |
| B | OUT | InSync | BOOL | Cam is engaged for the first time. |
| E | OUT | Busy | BOOL | Set when "Execute" is set, will remain SET until function block is aborted by another command. |
| E | OUT | CommandAborted | BOOL | Function block is aborted by another command. |
| B | OUT | Error | BOOL | Error occurred within function block. |
| E | OUT | ErrorID | UINT | Error number |
| E | OUT | EndOfProfile | BOOL | Pulsed output signaling the cyclic end of the cam profile. |

**Tableau XII.3**

Caractéristiques du bloc de fonction MC_CamIn.

## F. Mode dans lequel le variateur peut se trouver

Le variateur peut être dans l'une des configurations cité ci-dessous en fonction de la commande.

Standstill :

Le processeur n'exécute pas de mouvement et est prêt pour une commande de position/vitesse

Errorstop:

Le processeur est en erreur.

Homing :

Le processeur exécute une procédure de « homing ». (Détermination de la position de l'axe)

Disabled

Le processeur est en position OFF

Discret Motion:

Le processeur exécute un mouvement avec une cible. Le mouvement a donc une position définie à atteindre

Continuous Motion

Le processeur exécute un mouvement sans cible. Le mouvement n'a pas de fin définie

Synchronized Motion :

Le processeur exécute une synchronisation d'un axe esclave avec un axe réel

La figure XII-14 est une représentation de l'ensemble des états du variateur suivant le bloc de fonction appelé

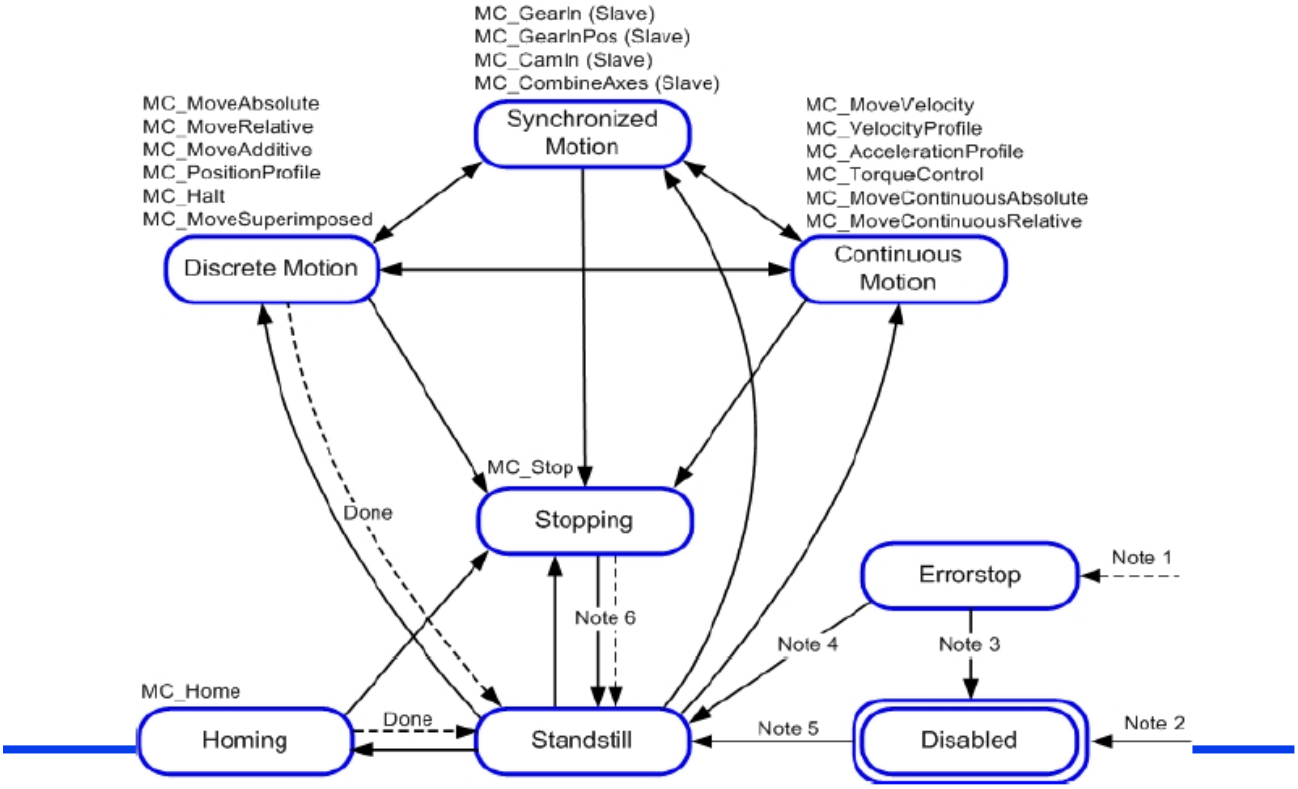



Figure XII-14

Principe de fonctionnement de la motion.

## G. Grafcet

### 1. Grafcet de commande

La figure XII-15 représente le grafcet de la commande des deux axes. (Format A3 en annexe)

Etape en rouge : action concernant l'axe virtuel
Etape en jaune : action concernant l'axe réel
Partie encadré : fonctionnalité présente dans le programme mais non activé.

1) Encadré bleu

- Le bloc de fonction MC_CamOut est utilisé pour désengager l'axe réel du virtuel. Ainsi l'axe réel au moment du désengagement continuera à « avancer/tourner » à une vitesse constante qui est celle figurant à l'instant du découplage. Pour une machine linéaire effectuant des mouvements de va et vient, ce bloc de fonction ne convient pas.
- Le bloc de fonction MC_Stop est utilisé pour stopper l'axe réel. Dans le programme, c'est l'axe virtuel qui est stoppé ce qui revient physiquement au même.

2) Encadré orange: Ces blocs sont également désactivés dans le programme.
MC_Move_Absolute ramène l'axe à une position désirée.
MC_Move_Additive déplace l'axe selon une distance désirée.

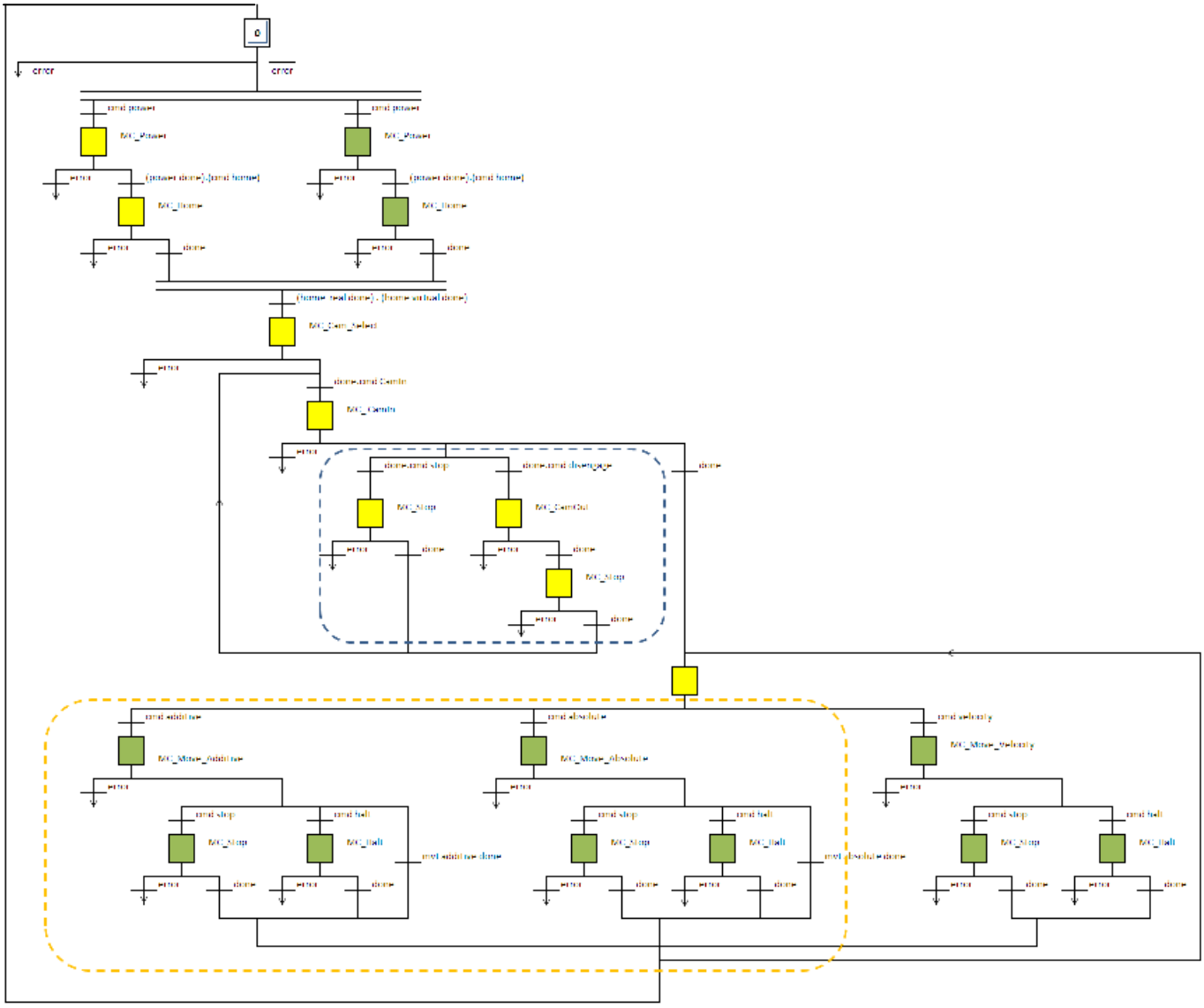


Figure XII-15

## 2. Logigramme de la gestion des erreurs

Lors de l'appel d'un bloc de fonction une erreur peut survenir. Cette erreur peut être unique ou suivie d'une erreur au sein du variateur (erreur interne) ce qui entrainera la mise hors tension du moteur.

La lecture des erreurs internes se fait par le bloc de fonction MC_ReadAxisError. Ce bloc doit toujours être lu cycliquement.

La gestion globale des erreurs consiste dans un premier temps à distinguer les deux types d'erreur et de les « reseter » l'une après l'autre jusqu'à la dernière pour que le système puisse être à nouveau opérationnel.

La représentation sous forme de grafcet étant complexe, la gestion des erreurs sera représentée sous forme de logigramme.

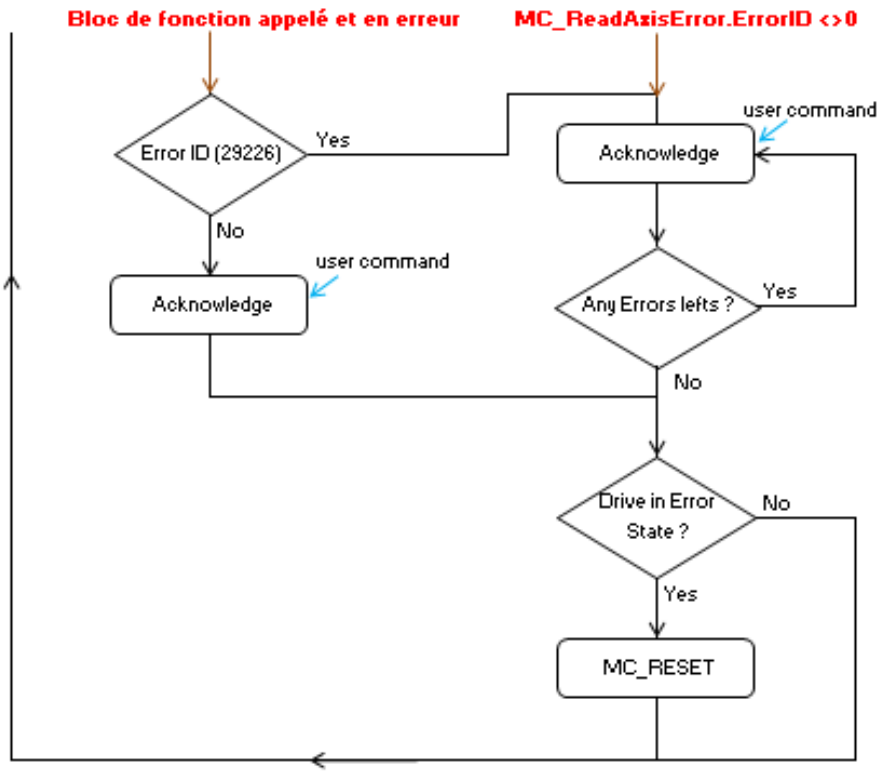



Figure XII-16

# XIII. Commande intégrée par le variateur, architecture des boucles de régulation

Le logiciel offre une commande scalaire ou une commande vectorielle indirecte. Etant donné que c'est la position du mover qui est à piloter, la commande vectorielle s'impose.

## A. Définition de la commande vectorielle :

La commande vectorielle consiste à piloter la position de l'arbre à chaque instant, ce qui requiert de connaitre la position du flux dans l'entrefer.
Une solution consiste alors à mesurer par le biais de capteurs le flux résultant (phase et amplitude) de manière directe. Cette solution a l'avantage d'être précise mais en revanche, la fragilité de la sonde soumise à la température de la machine et aux vibrations rend l'application de cette solution moins attractive.

La deuxième solution, qui est la commande vectorielle indirecte, consiste à connaître de façon précise la position du rotor pour pouvoir déterminer la position du flux rotorique.

## 1. Schéma général de commande avec le découplage des axes d et q

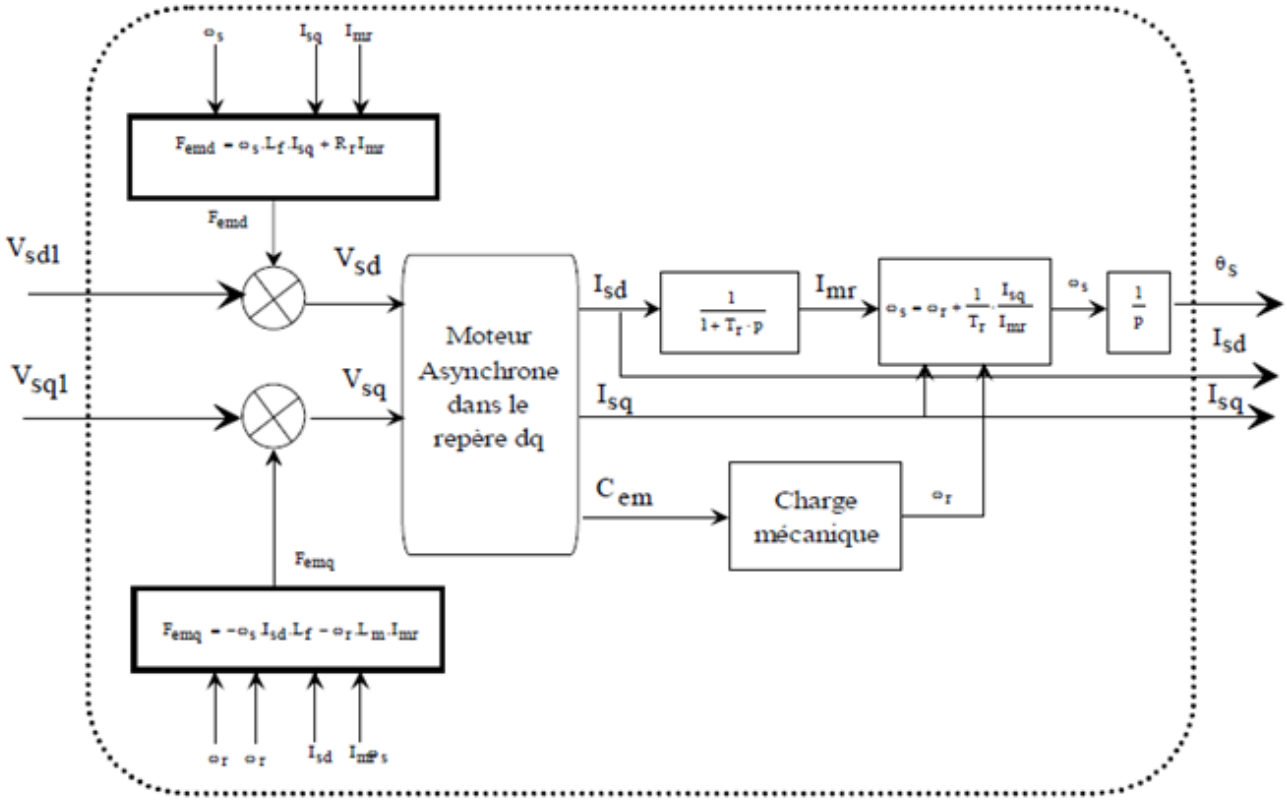



Figure XIII-1

Note sur les courants diphasé isq et isd :

Dans notre situation les consignes de isd et isq seront toujours variables dans le temps étant donné que nous somme dans un régime transitoire permanent.

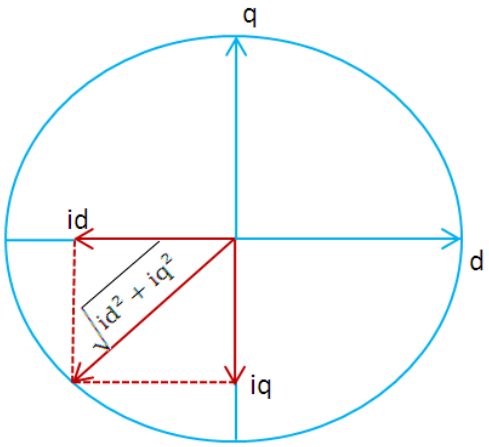



figure XIII-2

Diagramme des courants diphasés

I_CURRENT_RATED

## Boucle de régulation

On retrouve sur le schéma de la figure XIII-3 les différentes boucles de régulation en cascade.

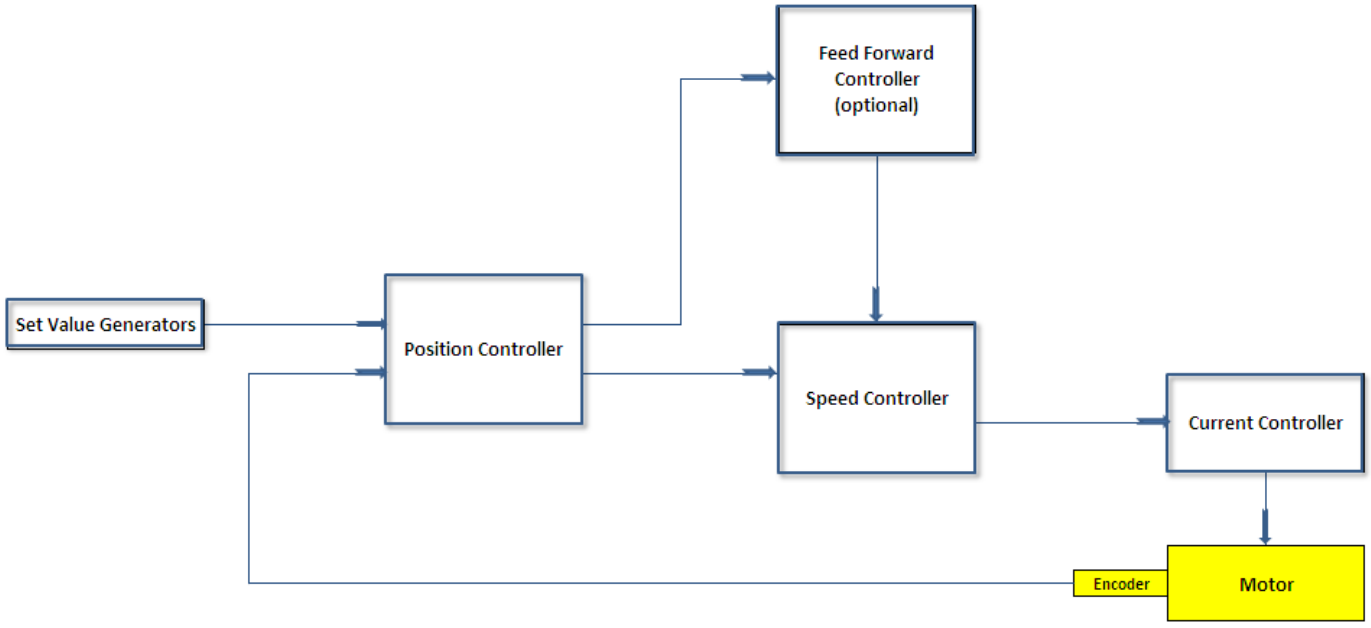



**Figure XIII-3**

Modèle des boucles d'asservissement

Set Values Generator : Les points esclaves du profil de cam sont lus et transmis à la boucle de position suivant la vitesse du maitre définie dans le programme.

Current Controller: Ce bloc est complètement inaccessible à l'utilisateur. Aucune visualisation ni aucun control ne peut être effectué. Il s'agit du pilotage des bras de l'onduleur avec en autre la gestion des temps morts.

1) La boucle de position (Position Controller) :

La boucle de position se trouve en amont des autres boucles car elle effectue la comparaison entre la position mesurée et la position désirée donnée par le bloc Set Value Generators. Une vitesse de consigne est donc déduite.

2) La boucle de vitesse (Speed controller) :

Cette boucle permet de contrôler la vitesse maximale

3) La boucle de courant :

La boucle de courant est optionnelle, elle consiste à contrôler le couple.

La valeur des paramètres internes est automatiquement calculée par le variateur en procédant à une commande spécifique. Ces coefficients sont accessibles et peuvent être modifiables. Il est cependant impossible de visualiser le signal en entrée et en sortie d'un bloc en question.

## 2. Exemple de représentation développée de structure interne du variateur

En rouge sont représentés les paramètres modifiables.

ANTI WIND UP : présent dans la boucle de position

L'anti wind up à pour but de d'annuler l'effet intégrateur lorsque le signal en sortie du sommateur dépasse la limite imposée par le saturateur p_max. La figure suivante montre le principe du anti-wind up dans un PID.

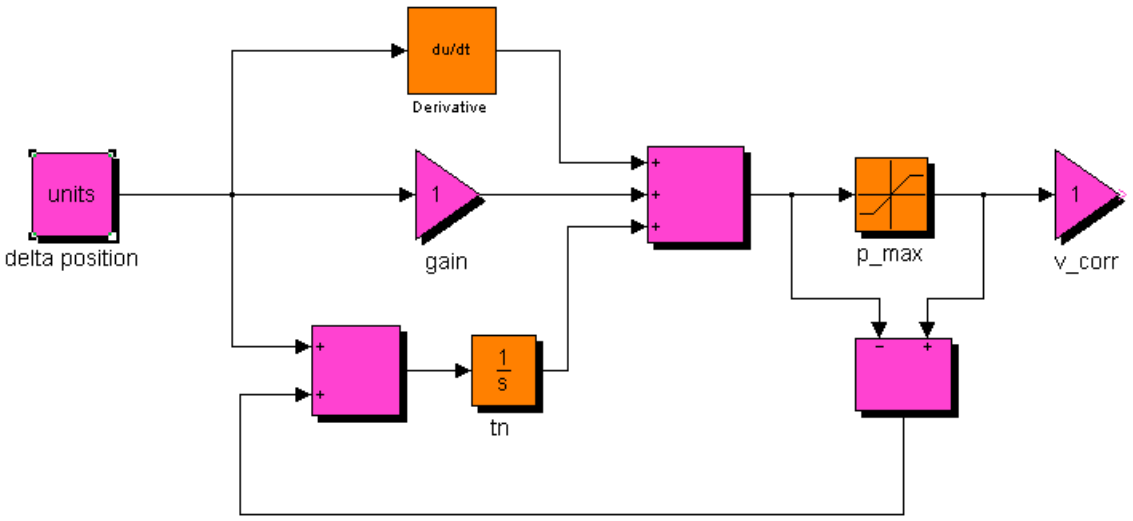



Figure XIII-4

ISQ FILTER : présent dans la boucle de vitesse

Ces trois filtres agissent uniquement sur isq (image du couple). Chacun des filtres peut être utilisé selon plusieurs configuration tel que :

- Filtre passe bas.

- Filtre rejecteur de bande.

- Limiteur de courant

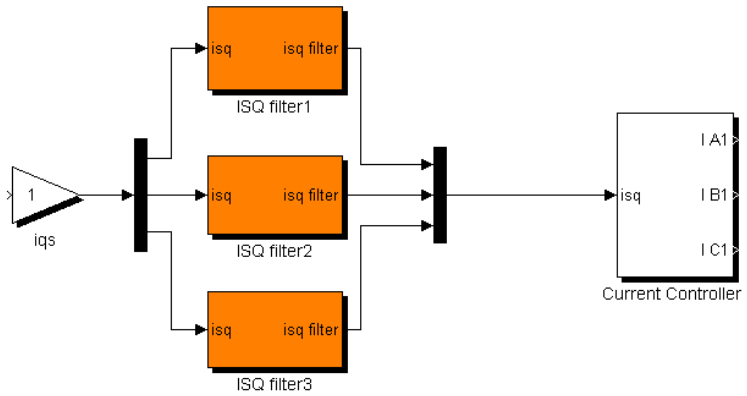



Figure XIII-5

# XIV. Mise en application du programme : Essais préalables, première constatation

Un travail important à été consacré pour la mise en œuvre du programme (programme en annexe 1.4). L'acquisition de la méthodologie de la programmation et celle du logiciel s'est finalement traduite comme l'occupation principale de ce mémoire.

Les premiers relevés de mesures ont été faits en paramétrant les paramètres moteurs avec ceux déterminés au paragraphe VII D.

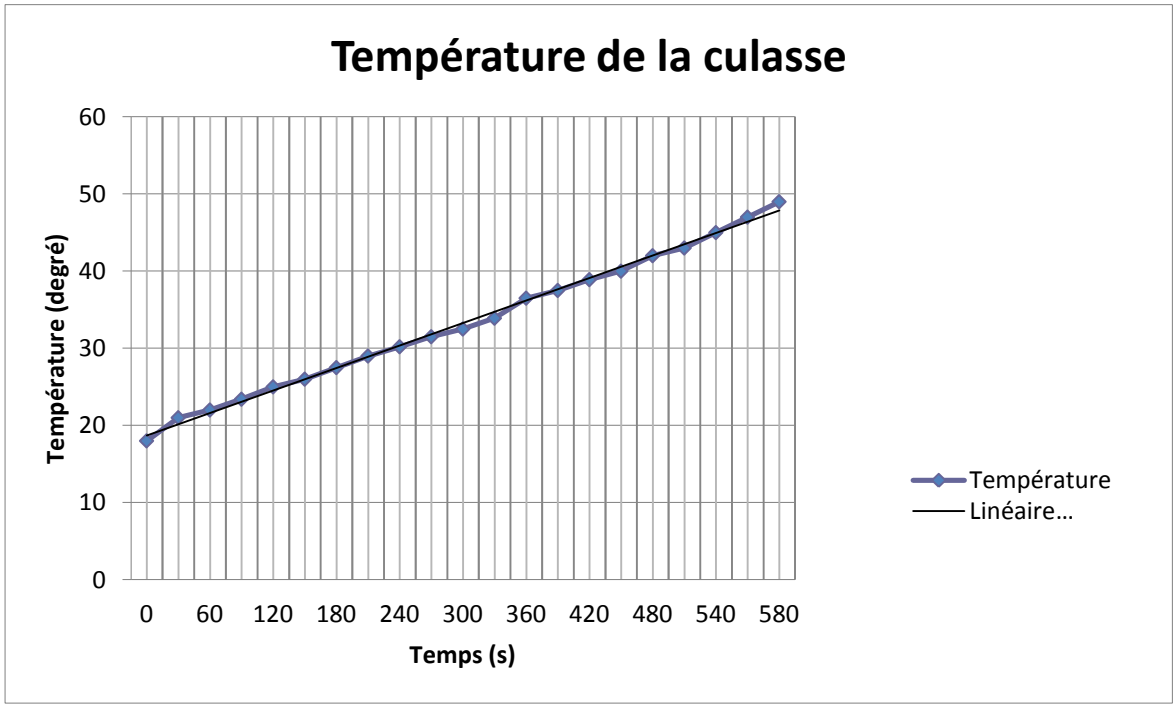
Ainsi des mesures ont été réalisées pour différentes fréquences de battement et de distance de déplacement du mover avec le programme.

1 ère constatation :

L'élévation de la température au niveau de la culasse par rapport à une machine tournante asynchrone de même puissance est élevée.

La figure XIV-1 représente un relevé de température en surface pour f=1 Hz et d= 40mm. Les mesures ont été effectuées jusqu'à 50 degrés pour ne pas endommager le moteur, mais l'élévation de température croit linéairement bien au-delà de cette valeur.

Remarque : La constante de temps thermique fait que l'élévation continue pendant un certain temps après la mise à l'arrêt.




Graphique XIV.1

Elévation de la température de la culasse. (T° Ambiant =18°)

Taux d'augementation de la témpérature : τ=0,05 degré/seconde

Cet effet était quelque peut prévisible car il est du aux courants de Foucault qui circulent dans la culasse non feuilletée. Bien que nous somme qu'au stade des premières mesures, il faudrait que le refroidissement se fasse tout comme le projet de la cogénération le stipulé au début (par un échangeur à eau).

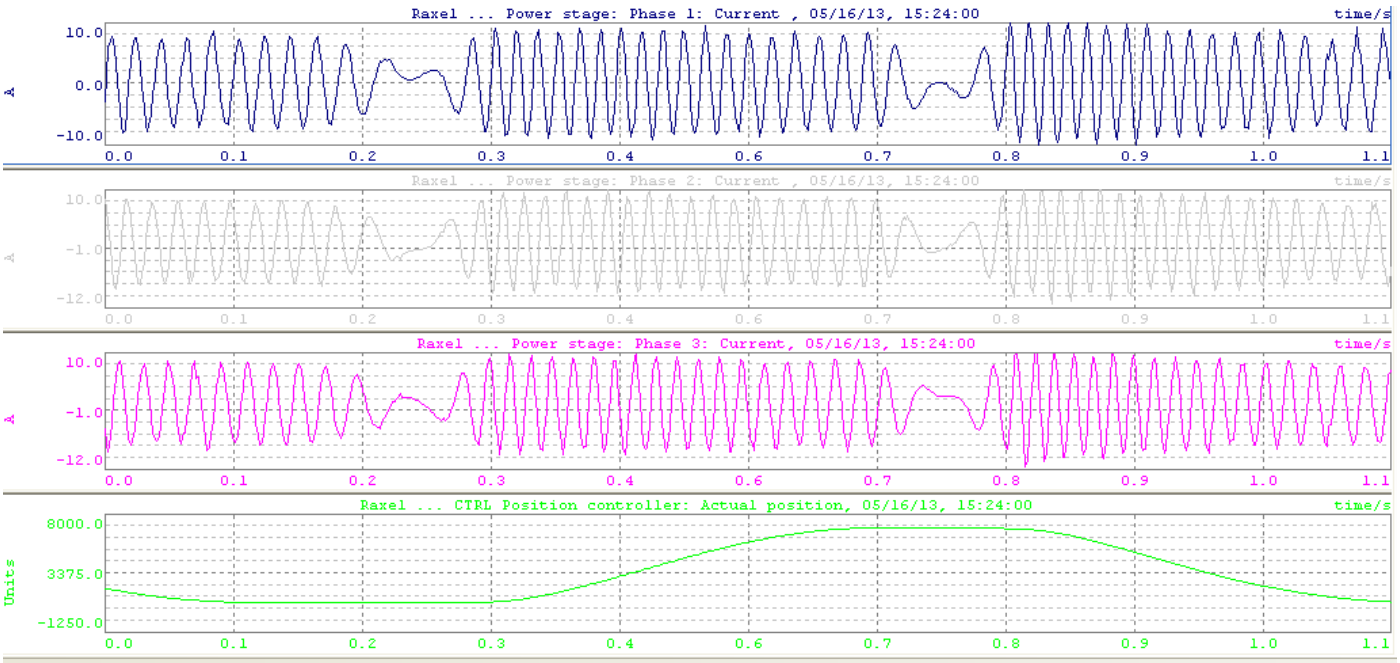
A défaut, si le projet de cogénération ne prend plus en compte la production de chauffage « direct », il faudrait pallier ce problème d'échauffement pour un fonctionnement nominal continu.

On peut alors opter pour deux solutions semblables.

- Feuilleter la culasse et le moyeu sur la longueur tout comme une machine tournante.
- Utiliser un matériau de type ferrite
- « Sillonner » ou « fendre » la culasse sur la longueur

Les deux premières solutions semblent les plus appropriées.

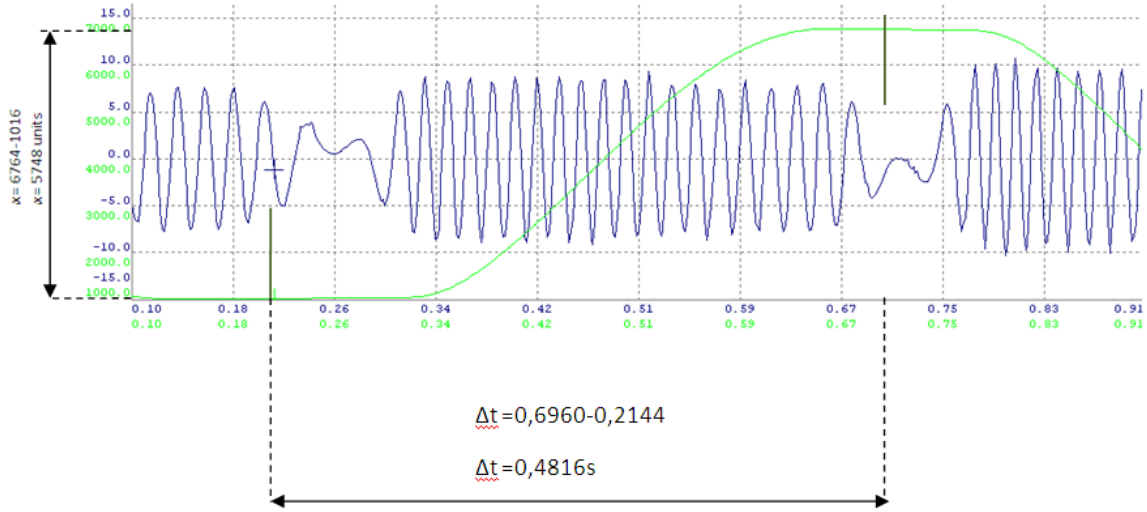
Le graphique XIV-2 est un relevé des courants absorbés et de la position du mover ( fméca =1 Hz d=40mm).




**Graphique XIV.2**

Respectivement i1, i2, i3, position (en concordance des temps)

Le graphique ci-dessous est un zoom sur le courant absorbé dans la phase 1.




**Graphique XIV.3**

Représentation du courant dans la phase 1 et de la position du mover sur une demi-période mécanique (0.5s)

Ieff sur la période sinusoïdale : 5 A

Ieff sur l'ensemble de la période : 5,5 A

Deuxième constatation :

De par l'étude sommaire des courants, on peut déduire que leur fréquence est élevée par rapport à la vitesse mécanique, ce qui se traduit par un glissement et des courants de Foucault important. (Les courants de Foucault sont proportionnels au carré de la fréquence des courants, et du champ d'induction.)

# XV. Influence des paramètres, relevés et interprétations

L'inertie du mover fait que des courants très intenses doivent être injectés dans les enroulements au moment du changement de sens. Ainsi le profil de Cam a été travaillé pour que les dérivées de position soient plus « douces ».

Profil de position

Profil de la dérivé de la positon


**Figure XV-1**

Profils de vitesse et de position du « générateur »

## A. Mesure du glissement pour différentes vitesses mécaniques :

Détermination de la vitesse du champ magnétique en m/s pour une fréquence de 1 Hz (fréquence de référence pour les calculs suivant).

Pour f=1Hz, T= 1s

Mesures et calcul du glissement :

| fréquence mécanique* (Hz) | fréquence électrique (Hz) | Vitesse du champ magnétique (m/s) | Vitesse mécanique (m/s) | glissement (%) |
|---|---|---|---|---|
| 1 | 100 | 9,6 | 0,08 | 99,17 |
| 2 | 166 | 15,936 | 0,16 | 99,00 |
| 5 | 200 | 19,2 | 0,4 | 97,92 |

**Tableau XV.1**

Constatation :

Le glissement est bien supérieur par rapport à une machine conventionnelle tournante (2% à 6%) comme on pouvait s'y attendre, mais il demeure au-delà des valeurs théoriques.
Il est important de rechercher à réduire ce glissement (idéalement 60 %) pour minimiser les pertes fer, optimiser la force et surtout pour pouvoir régénérer de l'énergie active avec le deuxième moteur en opposition.

La figure ci-dessous représente la force théorique en fonction du glissement.

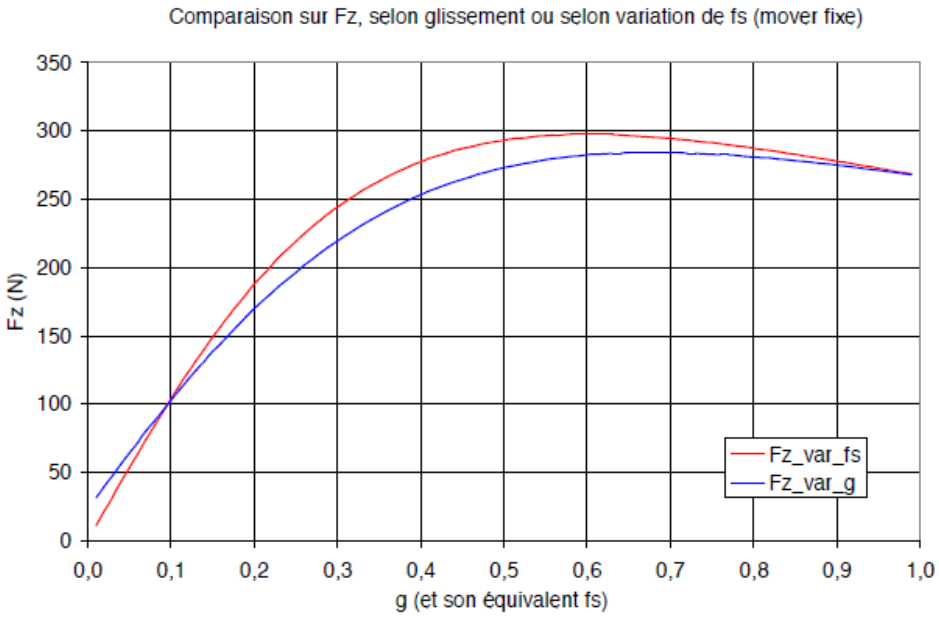



**Figure XV-2 :** Force en fonction du glissement [G]

Une méthode serait de corriger/modifier les paramètres du variateur, mais étant donné que l'on doit faire face à un système dont on ne peut agir que sur certain paramètres accessibles (propriété intellectuelle / protection) nous sommes confrontés à trouver une autre méthode.

Il est néanmoins bon de rappeler que le variateur B&R Automation est à caractère industriel, et qu'il n'est pas à la base conçu pour les machines linéaires asynchrones, bien que l'analogie ait été effectuée. (L'auto détermination des paramètres inconnus ne donne aucun résultat cohérent).

Ainsi nous pouvons à priori essayer de varier les paramètres de R rotorique et de L rotorique étant donné que ceux-ci découlent de calculs et d'estimations.

## B. Recherche d'un « optimum » par la variation des paramètres Rr et Lr

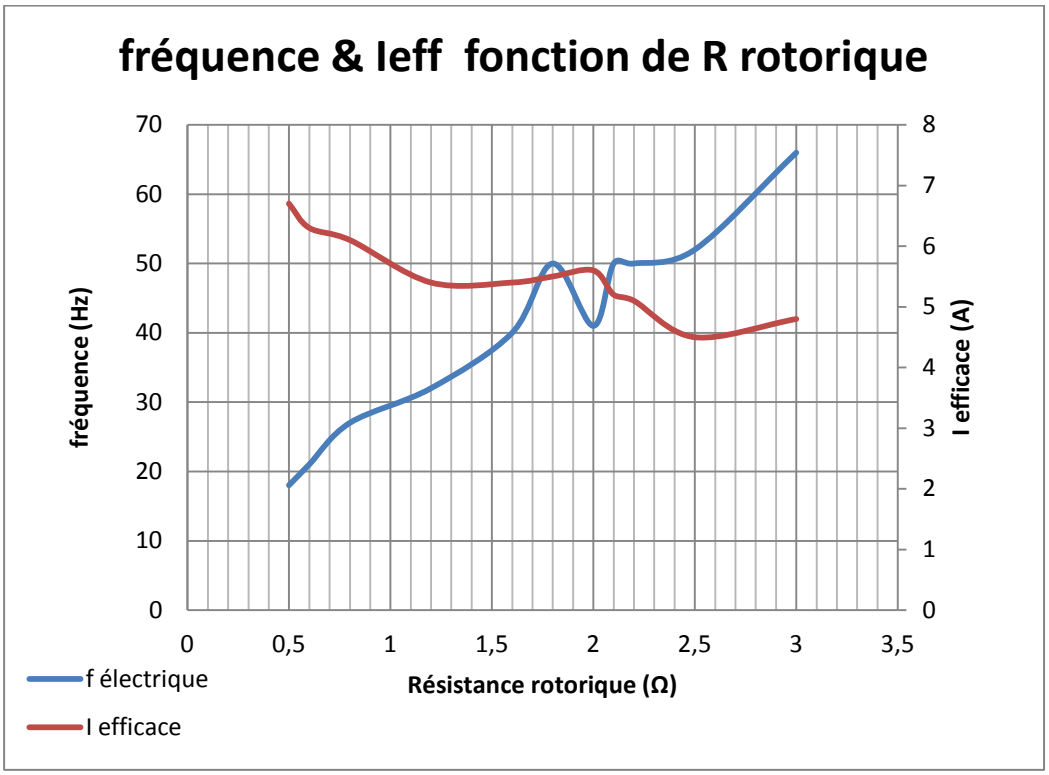
| Motor parameters | | | | |
|---|---|---|---|---|
| MOTOR_VOLTAGE_RATED | 48 | 400 | V | Motor: Rated voltage |
| MOTOR_VOLTAGE_CONST | 49 | 0 | mV*min | Motor: Voltage constant |
| MOTOR_SPEED_RATED | 50 | 6250 | 1/min | Motor: Rated speed |
| MOTOR_SPEED_MAX | 51 | 12500 | 1/min | Motor: Maximum speed |
| MOTOR_TORQ_STALL | 52 | 1 | Nm | Motor: Stall torque |
| MOTOR_TORQ_RATED | 53 | 1 | Nm | Motor: Rated torque |
| MOTOR_TORQ_MAX | 54 | 3 | Nm | Motor: Peak torque |
| MOTOR_TORQ_CONST | 55 | 0 | Nm/A | Motor: Torque constant |
| MOTOR_CURR_STALL | 56 | 17 | A | Motor: Stall current |
| MOTOR_CURR_RATED | 57 | 17 | A | Motor: Rated current |
| MOTOR_CURR_MAX | 58 | 19.69 | A | Motor: Peak current |
| MOTOR_WIND_CROSS_SECT | 59 | 0 | mm² | Motor: Line cross section |
| MOTOR_STATOR_RESISTANCE | 60 | 1.58 | Ohm | Motor: Stator resistance*2.638855 |
| MOTOR_STATOR_INDUCTANCE | 61 | 0.0327 | Henry | Motor: Stator inductance*0.023195 |
| MOTOR_INERTIA | 62 | 0.0001167 | kgm² | Motor: Moment of inertia |
| MOTOR_COMMUT_OFFSET | 63 | 0 | rad | Motor: Commutation offset |
| MOTOR_ROTOR_RESISTANCE | 76 | 2,1 | Ohm | *2,6388555 |
| MOTOR_ROTOR_INDUCTANCE | 77 | 0.023195 | Henry | Motor: Rotor inductance |
| MOTOR_MUTUAL_INDUCTANCE | 78 | 0.014 | Henry | Motor: Mutual inductance*0.036761 |
| MOTOR_MAGNETIZING_CURR | 79 | 1 | A | Motor: Magnetizing current |
| MOTOR_TAU_THERM | 849 | 0 | s | Motor: Thermal time constant |

**Figure XV-3**

Dans les essais qui suivent, les fréquences sont celles mesurées sur la partie linéaire de la courbe de position.

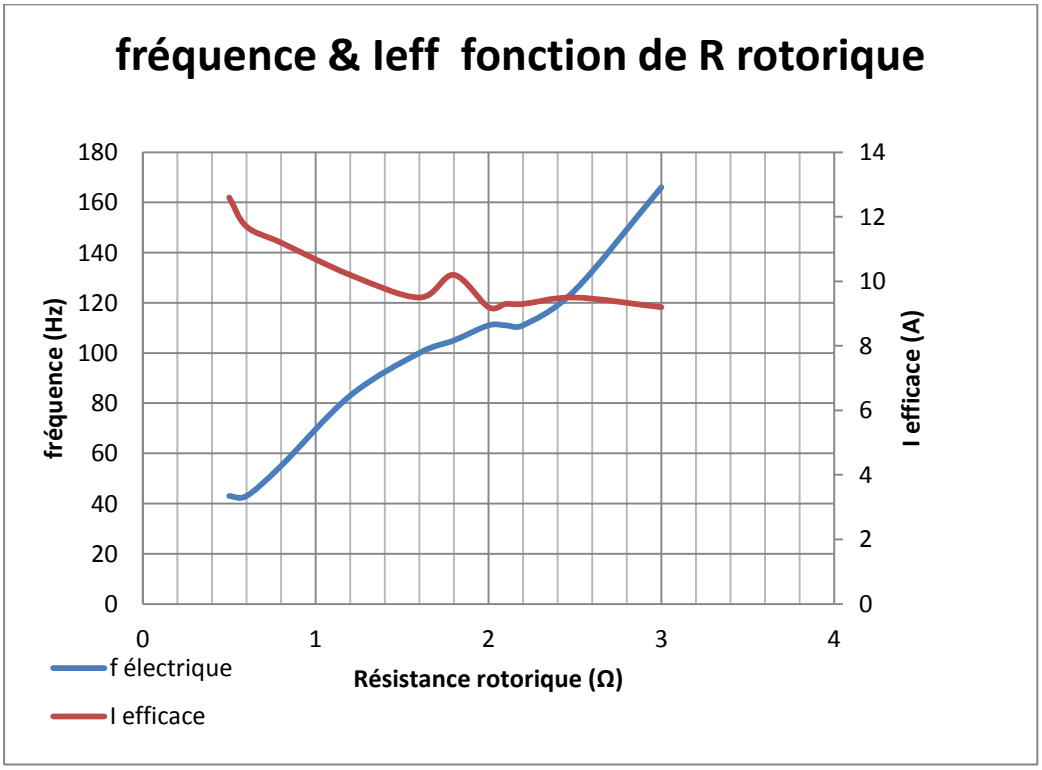
### 1. Variation de Rr

Essai pour une fréquence mécanique de 1 Hz sur une course de 40 mm



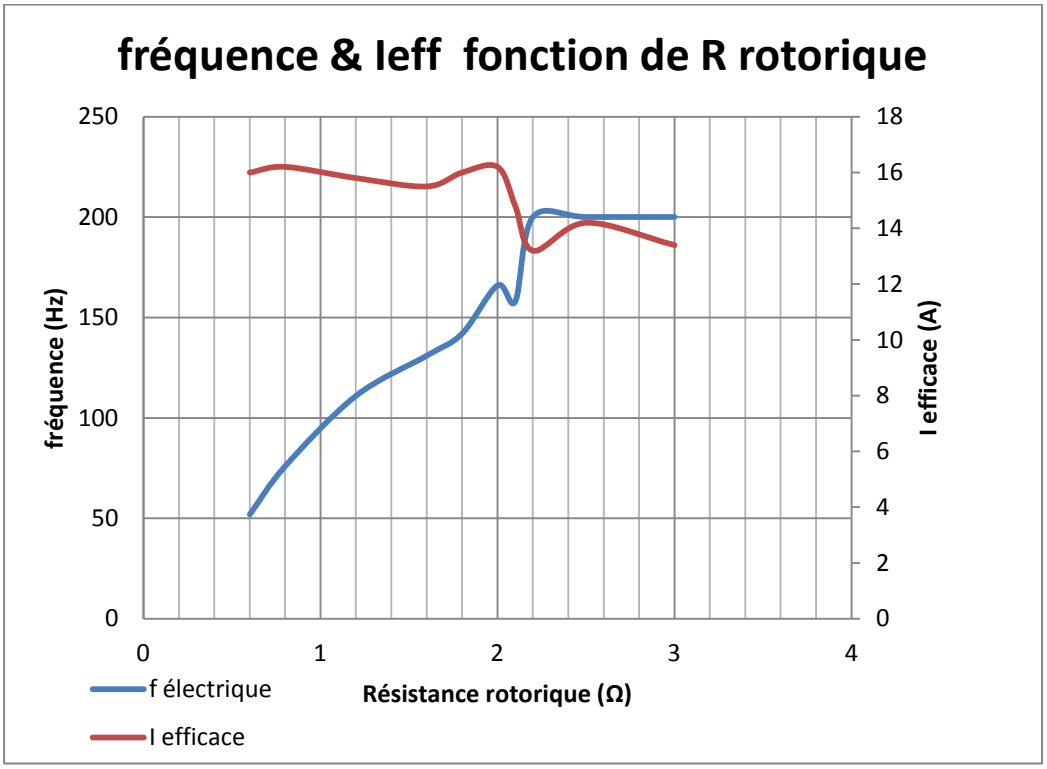

Graphique XV.1

Essai pour une fréquence mécanique de 5 Hz sur une course de 40 mm




Graphique XV.2

Essai pour une fréquence mécanique de 10 Hz sur une course de 40 mm

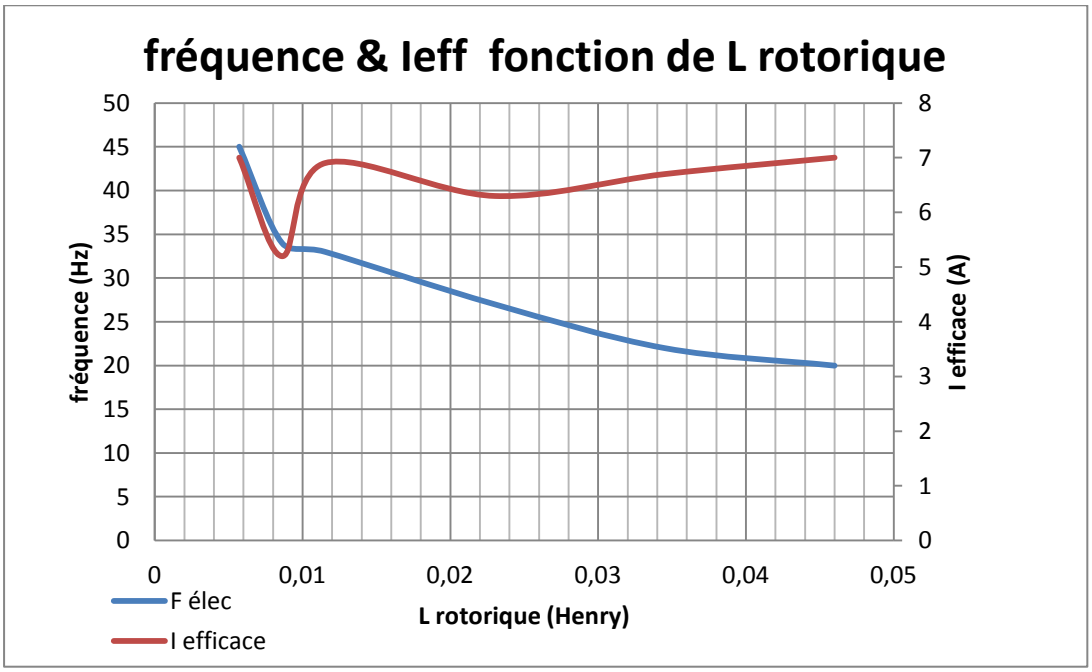



Graphique XV.3

Les mesures ont étaient effectuées sur une plage allant de 0.5 à 3 ohms. En étant hors de cette plage, le fonctionnement sera limité à 10 Hz, car les courants crêtes deviennent importants, il est cependant possible d'aller au delà. On peut à présent essayer de varier L'inductance rotorique en se basant sur une résistance rotorique de 0.8 Ω

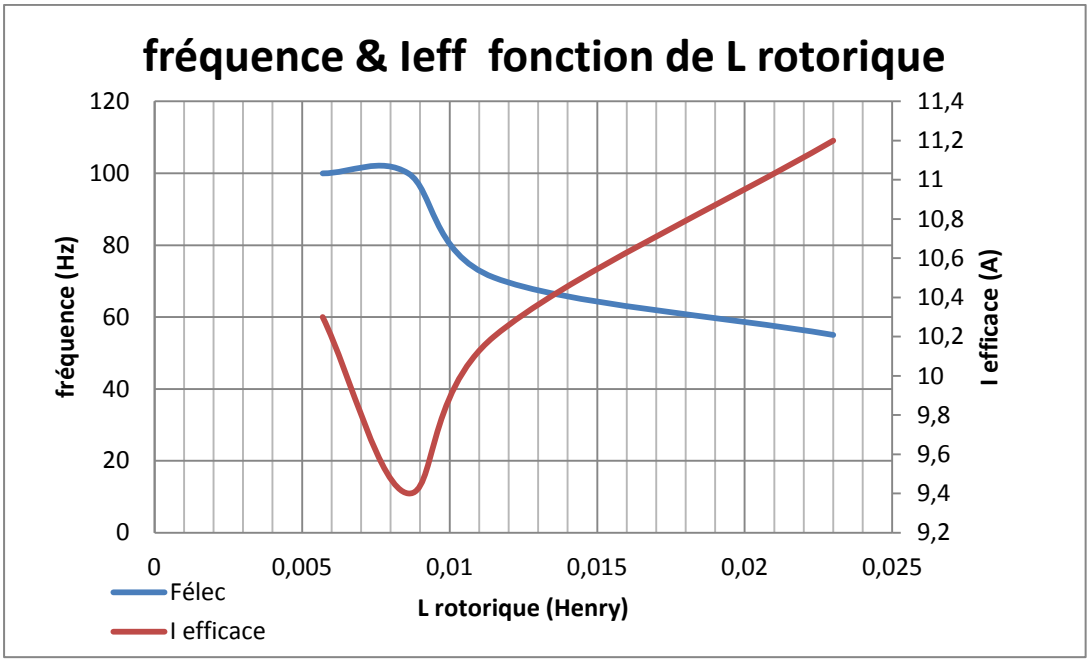
### 2. Variation de Lr

Essai pour une fréquence mécanique de 1 Hz sur une course de 40 mm



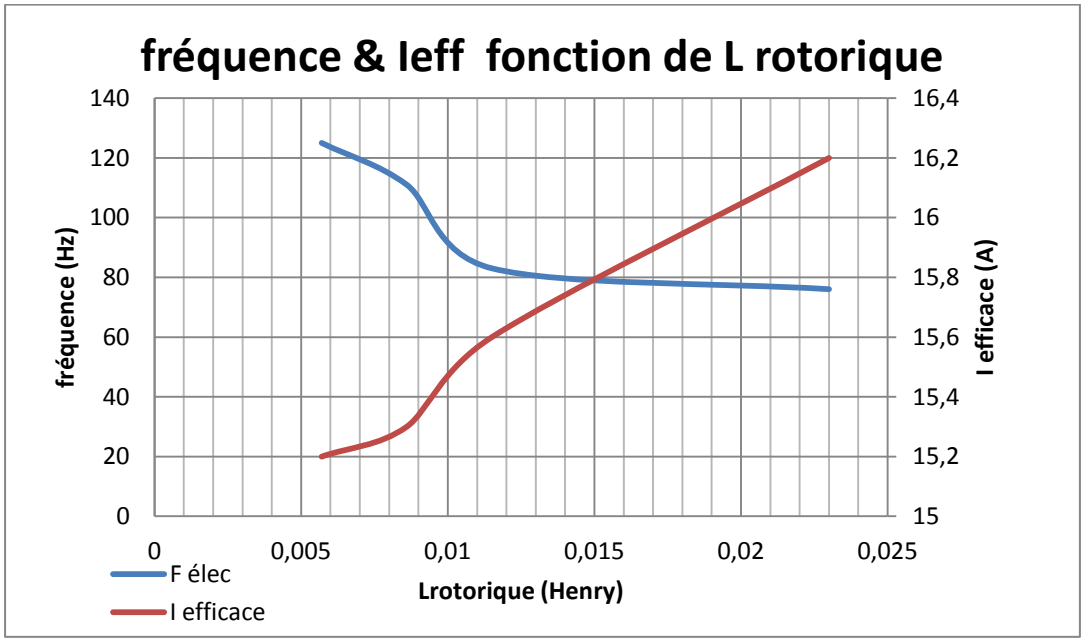

Graphique XV.4

Essai pour une fréquence mécanique de 5 Hz sur une course de 40 mm




Graphique XV.5

Essai pour une fréquence mécanique de 10 Hz sur une course de 40 mm

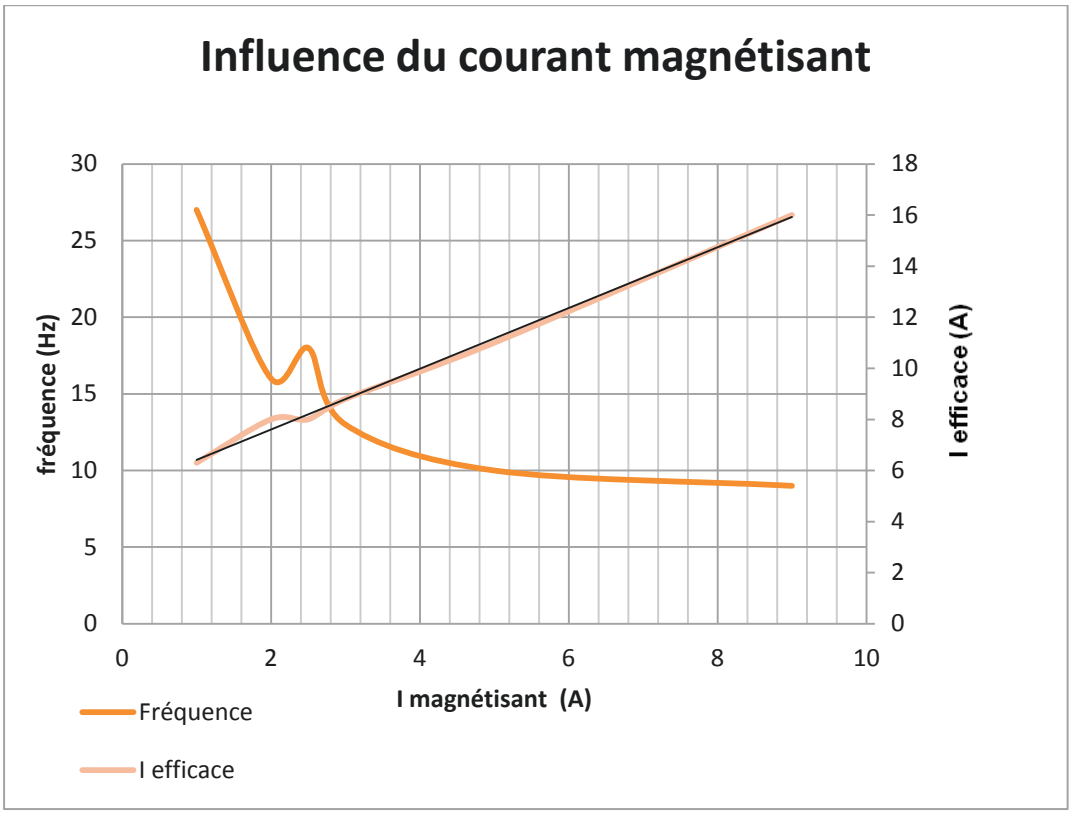



Graphique XV.6

La même remarque s'applique lors de la variation de Rr. En étant hors de cette plage le fonctionnement ne peut être assuré au-delà de 10 Hz.

### 3. Variation du courant magnétisant

La variation du courant magnétisant influe sur la fréquence des courants absorbés.




Graphique XV.7

A ce stade d'étude on s'aperçoit bien qu'il y a un problème de commande. Les résultats obtenus confirment que la commande n'est pas adaptée.

Il conviendra de rechercher ces « anomalies ». On peut cependant faire un ensemble de relevé des courants, et traiter ensuite la partie sur la régénération afin de voir si le problème est similaire.

(En conservant Rr =0.8 ohms et Lr=0.023 Henry et I magnétisant à 2,5 ampères).

(La variation des autres paramètres tels que le couple ou la vitesse nominale n'apporte pas de différences notables.)

Mesure des nouveaux glissements avec les paramètres retenus :

| fréquence mécanique* (Hz) | fréquence électrique (Hz) | Vitesse du champ magnétique (m/s) | Vitesse mécanique (m/s) | glissement (%) |
|---|---|---|---|---|
| 1 | 27 | 2,592 | 0,08 | 96,91 |
| 2 | 44 | 4,224 | 0,16 | 96,21 |
| 5 | 55 | 5,28 | 0,4 | 92,42 |

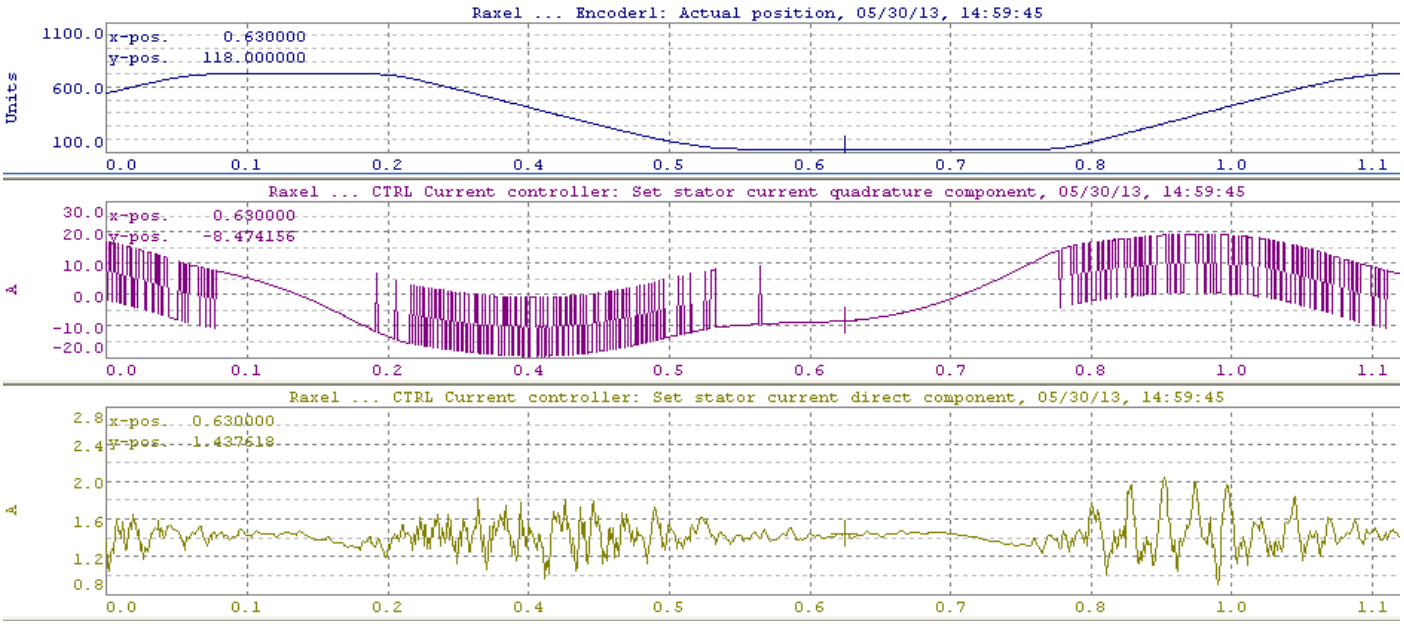
Tableau XV.2

Le nouveau glissement est à peine inférieur au glissement avec les paramètres initiaux.

## C. Visualisation des courants

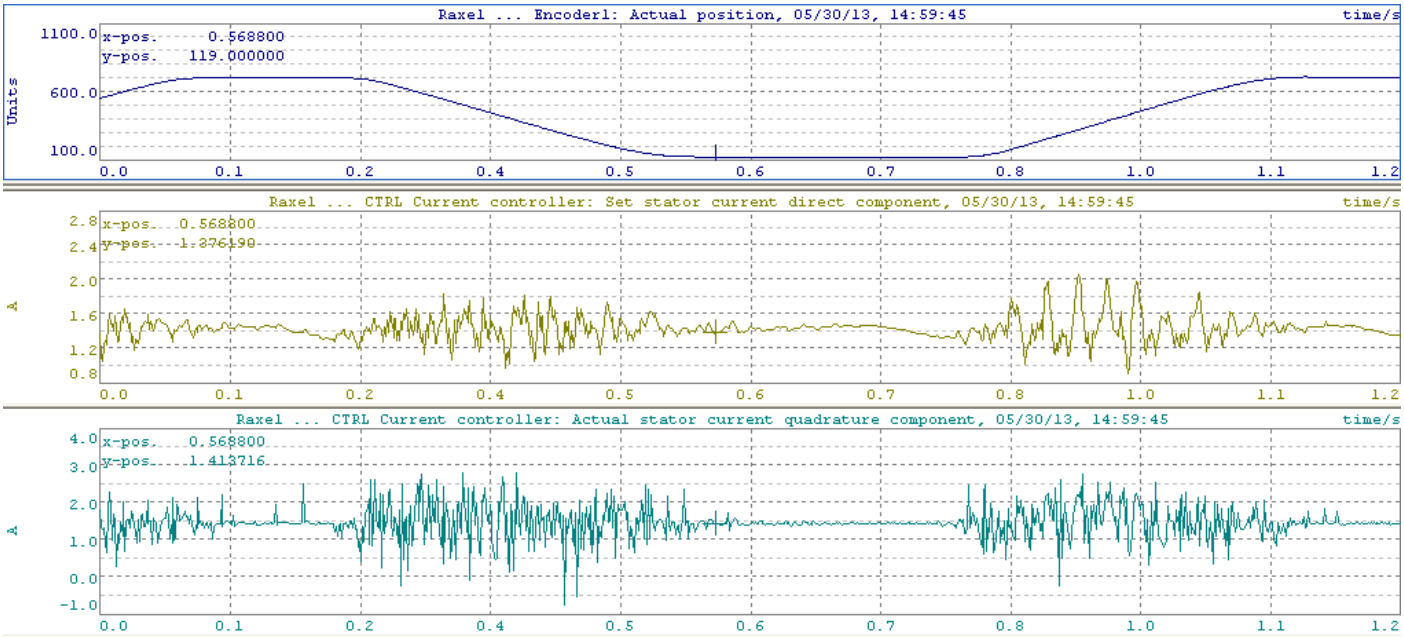
### 1. Courant de consigne d-q

Les consignes sont calculées en fonction des paramètres moteurs et du profil de cam.




Graphique XV.8

### 2. Courant réel mesuré




Graphique XV.9

## D. Calcul des pertes statorique

Puisque le logiciel permet de visualiser les courants dans le repère diphasé d-q, on peut se permettre de calculer les pertes joules statorique par la relation suivante :

Les caractéristiques suivantes sont données pour une fréquence de 1 Hz avec un déplacement de 40mm.

**Position du mover**

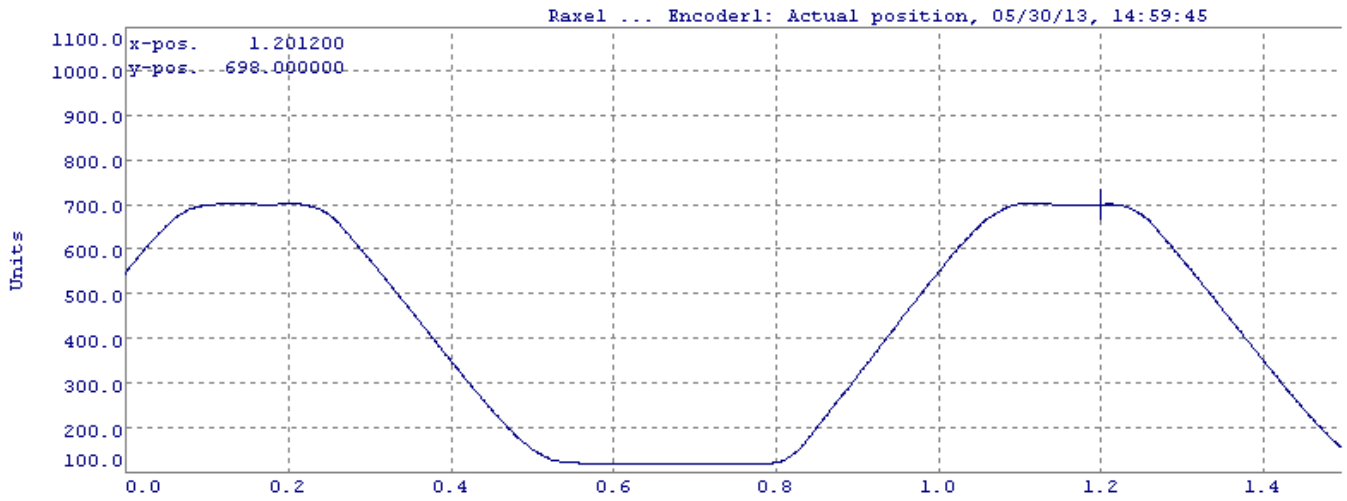



Figure XV-4

**Courant Id**

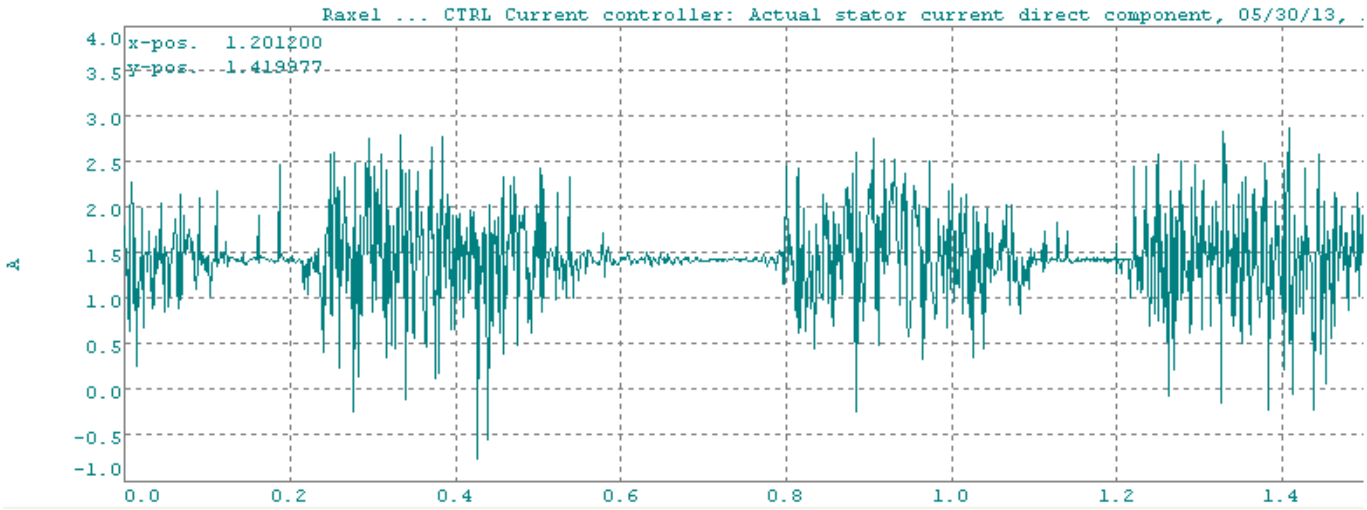



Figure XV-5

**Courant Iq**

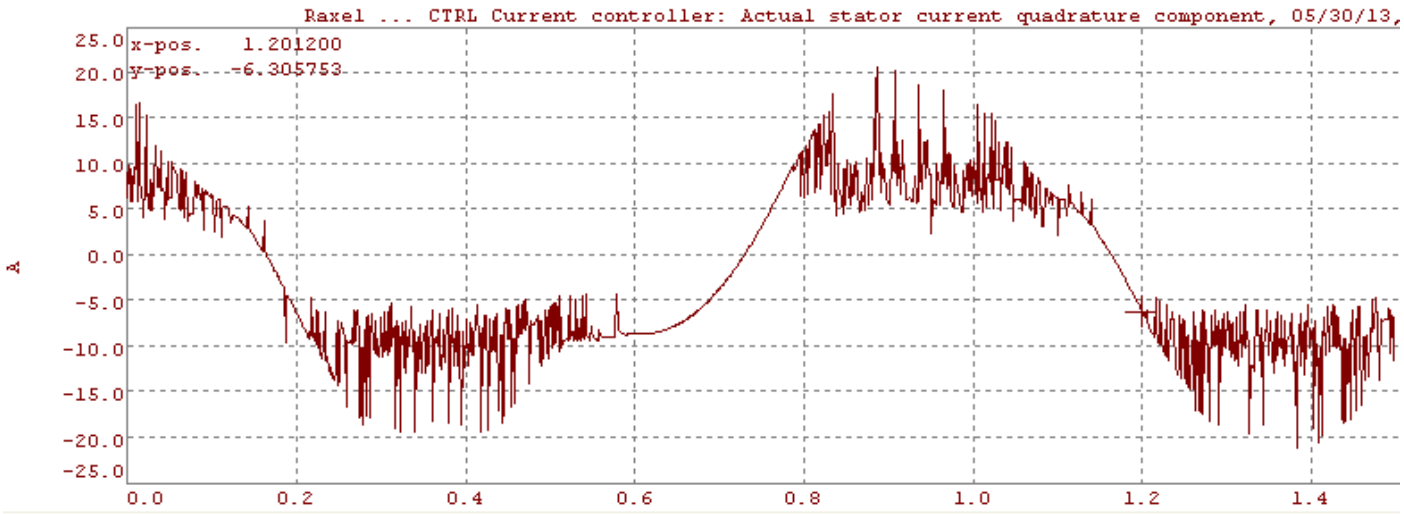



Figure XV-6

**Perte joule statorique**

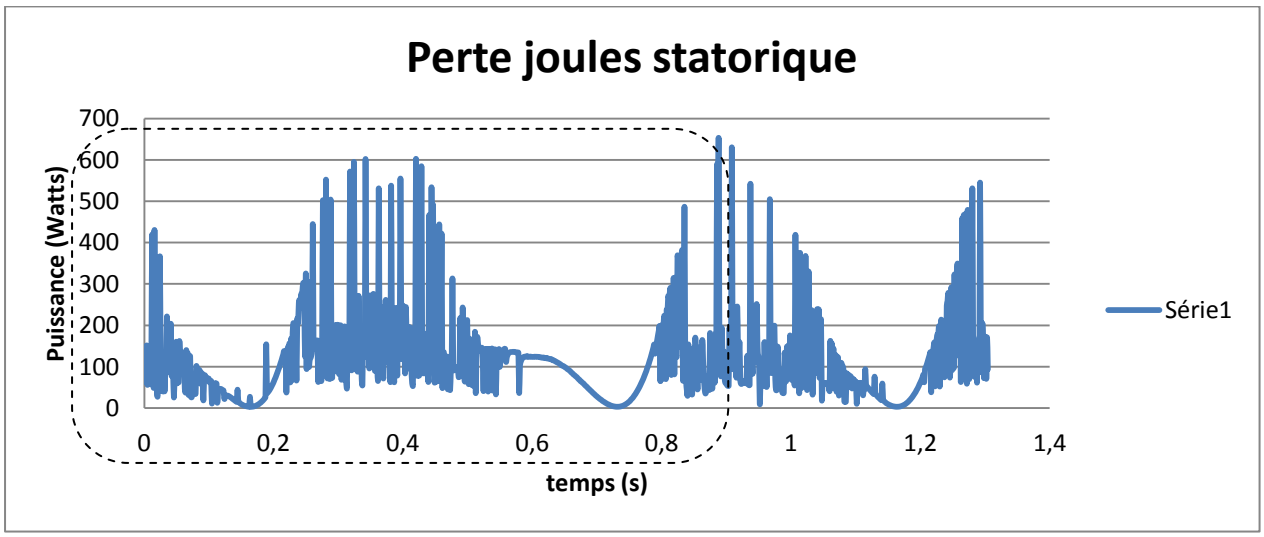



Figure XV-7

Les pertes joules sont de 113,7 Watts sur une période alors qu'elles étaient de 596 Watts avant la recherche d'un R et Lr « optimal »

# XVI. Regénération

## A. Généralité

En mode moteur, le champ statorique que l'on peut qualifier de « translatoire » est supérieur au déplacement mécanique du mover. Ainsi le flux statorique induit des courants dans le mover qui à sont tours produit un flux ayant une polarité magnétique opposé au stator.

La génération d'électricité n'est possible que si le glissement devient négatif, autrement dit si la fréquence mécanique du mover dépasse celle du champ inducteur. Le courant principalement réactif absorbé au stator induira toujours des courants dans le rotor, mais du fait que le flux rotorique à présent coupe les bobines statorique, un courant active est ainsi produit le rendant générateur.

Ainsi en mode générateur, le courant apparent de ligne se découposera donc en un courant actif et un courant réactif tout comme mode moteur

Etant donné que la machine est connectée au réseau via un variateur électronique, il n'est pas nécessaire d'affecter un banc de capacité pour fournir la puissance réactive nécessaire au fonctionnement.

La puissance délivrable par la génératrice est donné par la relation :[I]

$$\underline{\quad} \quad \underline{\qquad}$$

Avec cosϕg : cosϕ generateur

Avec cosϕm : cosϕ moteur

## B. Mesures

Le variateur pilotant le générateur doit avoir son parId 238 activé à 2, sinon le générateur s'opposera systématiquement aux variations mécaniques.

(Les paramètres « moteur » du moteur ont était remodifiés suite à un couple plus important du à la mise sous tension de la génératrice.)

Les relevés de la figure XVI-1.a et XVI-1.b représente les courants absorbés sur les trois phases du générateur pour une fréquence de 2 Hertz mécanique (La position du mover est donnés dans la figure XVI-1.b).

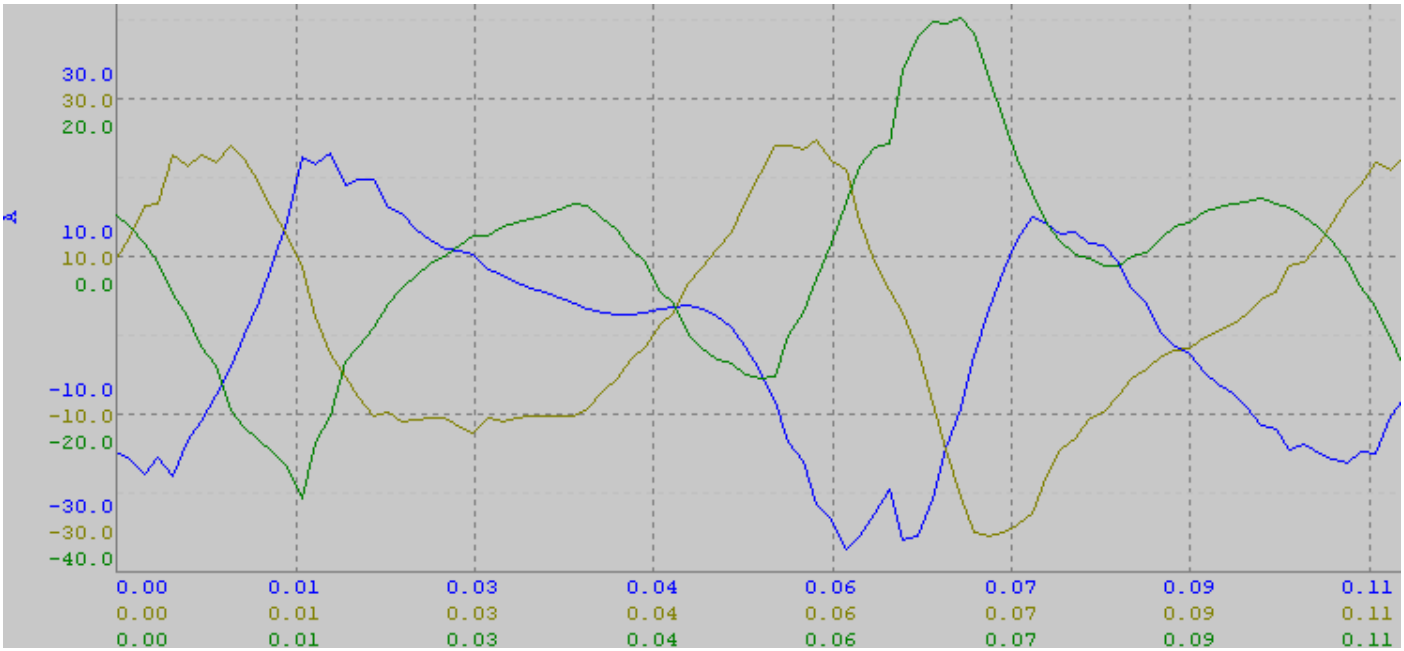



Figure XVI-1.a

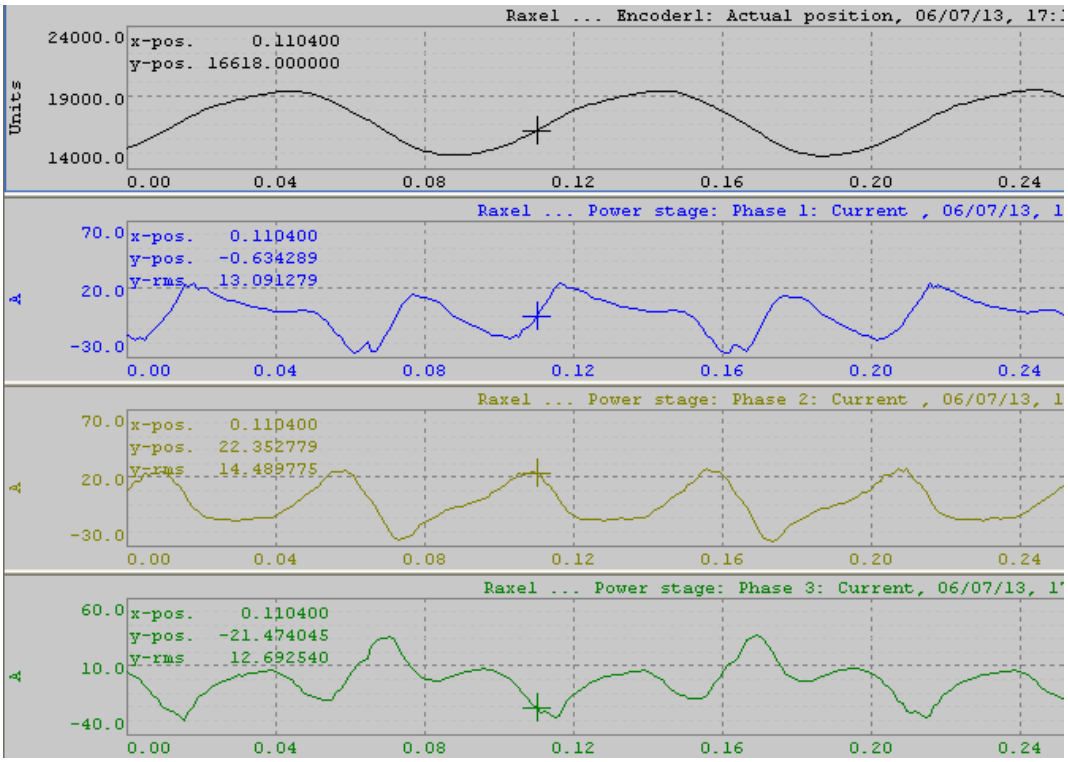



Figure XVI-2.b

On constate que les courants ont une fréquence électrique de 10 Hertz alors qu'il devrait être au maximum à 2 Hertz. Nous sommes donc confrontés au même problème que lors du fonctionnement moteur. Le glissement ne convient pas.

# XVII. Conclusions et Perspectives

L'objectif principal de ce mémoire d'ingénieur Cnam était de développer et mettre en œuvre un banc d'essai comprenant 2 machines asynchrones linéaires montées en opposition pilotées par du matériel industriel. Ce premier objectif a été atteint.

Le second objectif était de développer le logiciel qui permet de piloter les 2 machines, l'un en moteur et l'autre en générateur. Ce second objectif a été atteint.

Le troisième objectif était de générer de l'énergie électrique via la machine asynchrone utilisée comme génératrice. Cet objectif n'a pas pu être validé.

En synthèse, nous avons pu développer un banc d'essai qui permettre d'étudier les machines linéaires et de comparer les modélisations réalisées lors des précédant travaux en régime dynamique. En effet, actuellement seul les modèles statiques ont été validés. Nous avons mis en œuvre des moyens d'essai qui peuvent aussi servir à mieux comprendre le fonctionnement de ces machines asynchrones linéaires utilisées en hautes dynamiques. Nous avons notamment pu faire fonctionner le banc à des fréquences de plus de plus de 20 Hz sur des courses de 10mm.

Le matériel BR automation utilisé pour le contrôle commande de nos machines asynchrones linéaires n'est pas parfaitement adapté. Il faudrait que nous puissions modifier des paramètres liés au control commande et notamment pouvoir avoir des paramètres variables en fonction de la vitesse. Ce matériel est cependant très « puissant » en terme de possibilité. Il nous a permis grâce à ses nombreuses fonctionnalités de faire des mesures qui auraient été délicates par des appareils externes au système. Nous avons par exemple accès aux courants instantanés, aux courants dans le repère de Park, etc… en fonction de la position.

Concernant les machines construites par la société Servat, mis à part des problèmes d'isolements qui ont été rapidement détectés et réglés, les machines sont fiables. Le principe de sustentation indiqué en début de mémoire est vérifié et permet de soulager les guidages linéaires. Lors du fonctionnement, qui n'est pas optimisé comme nous l'avons indiqué (problème de glissements trop importants), la machine utilisée en moteur est sujet à des pertes fer et joule importante, la température importante n'a pas gêné le fonctionnement. Cela prouve qu'elle peut être utilisée en milieu chaud comme un cogénérateur.

Concernant l'aspect « système », le banc a permis de soulever un problème de contrôle commande. Nous avons mesuré des glissements trop importants, nous avons suspecté des problèmes de paramètres « moteur » à entrer dans le variateur. La variation de ces paramètres impacte effectivement le glissement et les performances globales de la machine. Nous n'avons pas pu actuellement définir les paramètres optimaux. Nous avons aussi pu grâce au fonctionnement en générateur mettre en avant un problème de définition du pas polaire vis à vis codeur et des paramètres correspondant dans le variateur.

Dans ce projet, beaucoup de temps a été nécessaire à l'appropriation et la formation sur le matériel du contrôle commande. Quelques problèmes mécaniques et électriques sur le banc ont aussi pris plus de temps que prévu.

Globalement, ce projet m'a permis d'apprendre beaucoup et d'assimiler des notions apprises à l'occasion des cours du Cnam (électromagnétisme, machines, commande vectorielle, etc…).

La poursuite de ces travaux devra porter sur l'optimisation du fonctionnement moteur et sur l'étude de la régénération. Le banc mis en place doit le permettre, ce mémoire doit aussi aider à faciliter à la prise en main d'autre utilisateur même si la formation sur le matériel BR Automation est indispensable selon moi.

La partie moteur et contrôle commande, une fois optimisée, pourra être utilisée comme vérin électrique. Une application est la création de gaz alterné, par exemple notre partenaire FEMTO a besoin d'étudier des flux de gaz

alternés dans des régénérateurs de moteur Stirling. Actuellement ils utilisent un système bielle manivelle qui ne permet pas de régler aisément la course.

Le SATIE continue à travailler sur l'optimisation de système de génération électrique linéaire, ce banc pourra servir à caler les modèles.

# XVIII. BIBLIOGRAPHIE

[A] http://www.rense.com/The Incredible Genius Of Eric Laithwaite

[B] http://www.directindustry.fr/prod/tecnotion-bv/moteurs-electriques-lineaires-haute-puissance-sans - fer-dc-synchrone-28399-225914.html

[C] Technique de l'ingénieur : Moteurs électriques à mouvement linéaire et composé. D3700-13

[D] Thèse Pierre Francois : Contribution à la modélisation électromagnétique d'un générateur linéaire à induction appliquée à un micro-cogénérateur Stirling à piston libre P 112

[E] Thèse Pierre Francois : Contribution à la modélisation électromagnétique d'un générateur linéaire à induction appliquée à un micro-cogénérateur Stirling à piston libre P 168

[F,G] Thèse Pierre Francois : Contribution à la modélisation électromagnétique d'un générateur linéaire à induction appliquée à un micro-cogénérateur Stirling à piston libre P 113,P98

[H] Mémoire Abderrachid Hani : DIMENSIONNEMENT ET REALISATION D'UN ACTIONNEUR LINAIRE TUBULAIRE ASYNCHRONE DE 1Kw, DEFINITION DE SA COMMANDE P44

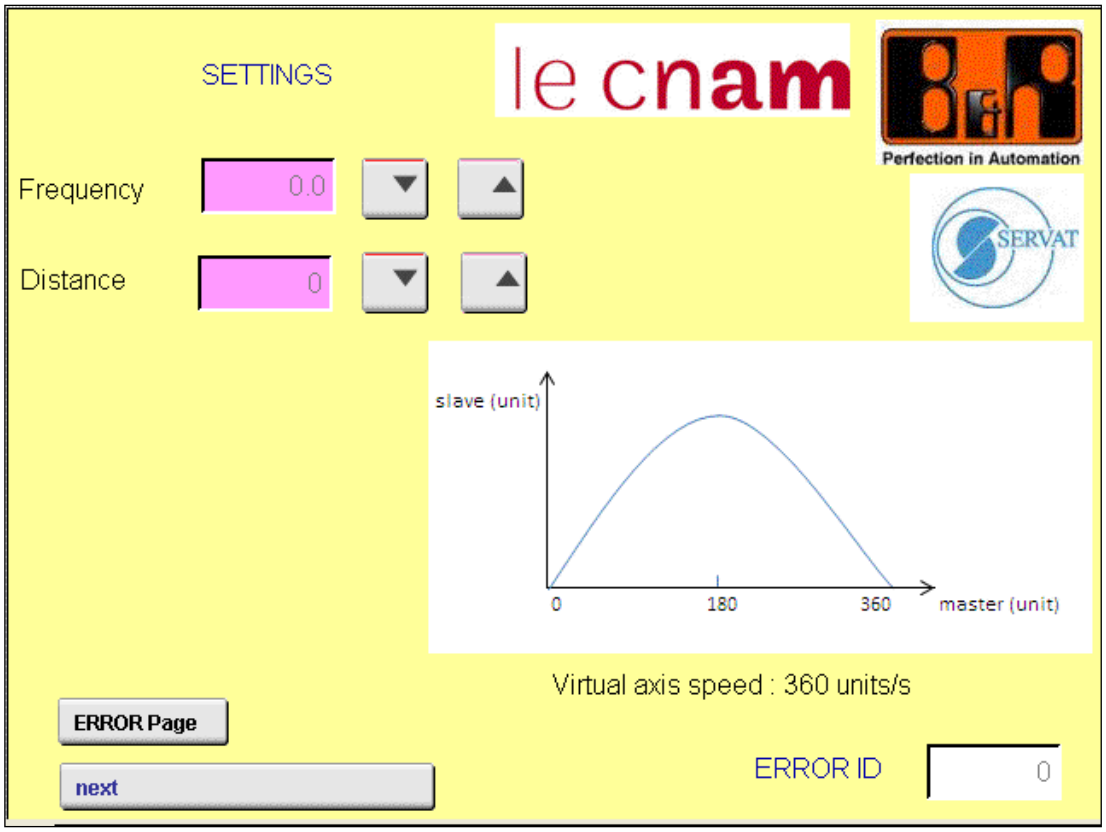
[I] Technique de l'Ingénieur : Génératrice Asynchrone D452-4

# XIX. ANNEXE

## A. Grafcet de commande

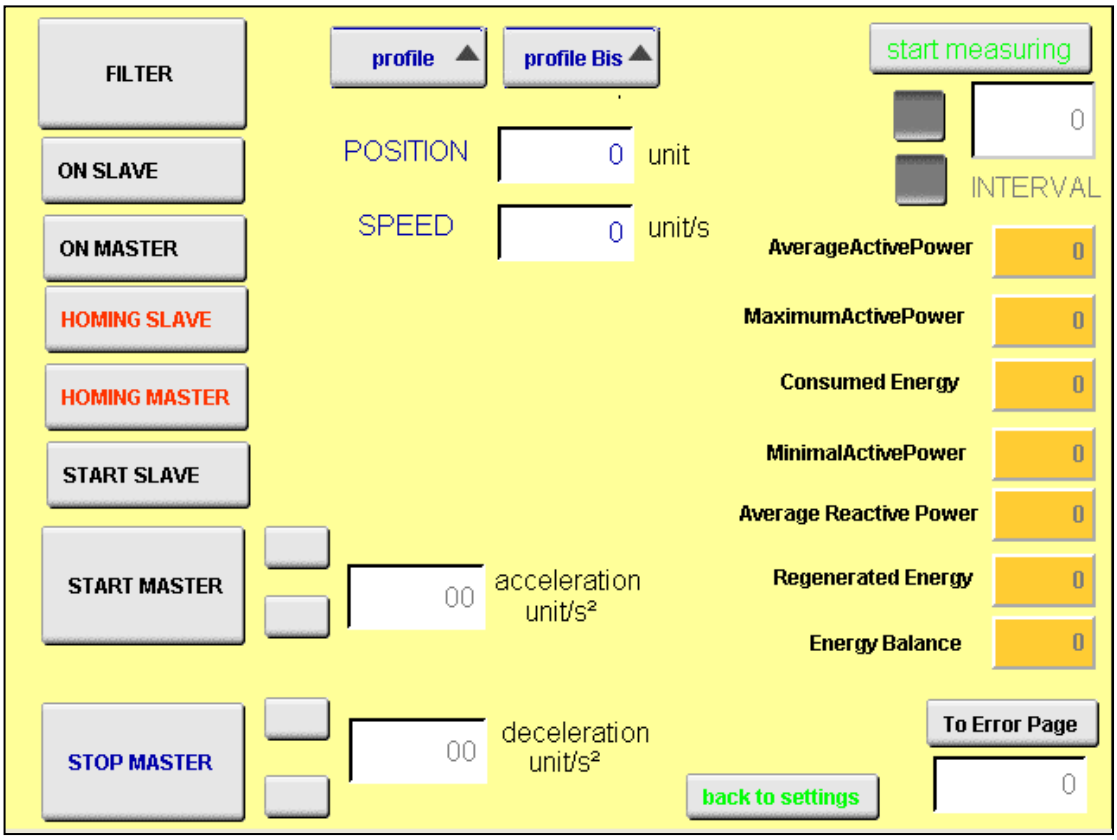
## B. Programme du moteur linéaire

### 1. Power Panel (interface)




Réglage des paramètres de distance et de fréquence

Figure XIX-1




Page de mise en fonctionnement du moteur linéaire avec affichage des mesures

Figure XIX-2


Page d'erreur

Figure XIX-3

## 2. PROGRAMME AXE REEL

**PROGRAM_INIT**

// This part in only read once, not cyclicaly

(* get axis object *)

Axis1Obj := ADR(Vaxe1); (*Master Axis*)

Axis2Obj := ADR(Raxe1); (*Slave Axis*)

AxisStep := STATE_WAIT; (* start step *)

**END_PROGRAM**

**PROGRAM _CYCLIC**

(*************************************************************

Control Sequence

```
**********************************************************)

(*************** CHECK FOR GENERAL AXIS ERROR *****************)
        IF ((MC_ReadAxisError_0.AxisErrorID <> 0) AND (MC_ReadAxisError_0.Valid = TRUE)) THEN
                AxisStep := STATE_ERROR_AXIS;
                (***************** CHECK IF POWER SHOULD BE OFF ******************)
        ELSIF (CamControl.Command.Power = FALSE) THEN
                IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE)) THEN
                        AxisStep := STATE_ERROR_RESET;
                ELSE
                        AxisStep := STATE_WAIT;
                END_IF
        END_IF
CASE AxisStep OF

                (******************* WAIT *************************)
                STATE_WAIT: (* STATE: Wait *)
                        IF (CamControl.Command.Power = TRUE) THEN
                                AxisStep := STATE_POWER_ON;
                        ELSE
                                MC_Power_0.Enable := FALSE;
                        END_IF
                        (* reset all fb execute inputs we use *)

                        MC_Home_0.Execute := FALSE;
                        MC_CamTableSelect_0.Execute := FALSE;
                        MC_CamIn_0.Execute := FALSE;
                        MC_Halt_0.Execute := FALSE;
////              MC_CamOut_0.Execute := FALSE; (*Can't be used for any linear with a back and forth motion*)
                        MC_ReadAxisError_0.Acknowledge := FALSE;
```

```
                MC_Reset_0.Execute := FALSE;
                (* reset user commands *)
                CamControl.Command.Home := FALSE;
                CamControl.Command.StartSlave := FALSE;
                CamControl.Status.ErrorID := 0;

        (******************** POWER ON **********************)
        STATE_POWER_ON: (* STATE: Power on *)
                MC_Power_0.Enable := TRUE;
                IF (MC_Power_0.Status = TRUE) THEN
                        AxisStep := STATE_READY;
                END_IF
                (* if a power error occured go to error state *)
                IF (MC_Power_0.Error = TRUE) THEN
                        CamControl.Status.ErrorID := MC_Power_0.ErrorID;
                        AxisStep := STATE_ERROR;
                END_IF
        (******************** READY **********************)
        STATE_READY: (* STATE: Waiting for commands *)
                        IF (CamControl.Command.Home = TRUE)THEN
                        CamControl.Command.Home := FALSE;
                        AxisStep := STATE_HOME;
                        ELSIF (CamControl.Command.StartSlave = TRUE) THEN
                        CamControl.Command.StartSlave := FALSE;
                        AxisStep := STATE_CAM_SELECT;

(*Can't be used for any linear with a back and forth motion*)
//              ELSIF (CamControl.Command.CamOut= TRUE) THEN
//                        CamControl.Command.CamOut := FALSE;
//                      AxisStep := STATE_CAM_STOP;
```

```
        END_IF
(******************** HOME **********************)
STATE_HOME: (* STATE: start homing process *)
        MC_Home_0.Position := CamControl.Parameter.HomePosition;
        MC_Home_0.HomingMode := mcHOME_DIRECT;
        MC_Home_0.Execute := TRUE;

        IF (MC_Home_0.Done = TRUE) THEN
                MC_Home_0.Execute := FALSE;
                AxisStep := STATE_READY;
        END_IF
        (* if a homing error occured go to error state *)
        IF (MC_Home_0.Error = TRUE) THEN
                MC_Home_0.Execute := FALSE;
                CamControl.Status.ErrorID := MC_Home_0.ErrorID;
                AxisStep := STATE_ERROR;
        END_IF
(******************** SELECT CAM TABLE **********************)
STATE_CAM_SELECT: (* STATE: Select the CAM *)
        //MC_CamTableSelect_0.CamTable := 'profile';
        MC_CamTableSelect_0.CamTable := moteurL.status.profile;
        MC_CamTableSelect_0.Periodic := mcPERIODIC;
        MC_CamTableSelect_0.Execute := TRUE;
        (* wait for select done *)
        IF (MC_CamTableSelect_0.Done = TRUE) THEN
                CamTableID := MC_CamTableSelect_0.CamTableID;
                MC_CamTableSelect_0.Execute := FALSE;
                AxisStep := STATE_CAM_START;
        END_IF
        (* check if error occured *)
```

```
                    IF (MC_CamTableSelect_0.Error = TRUE) THEN
                            CamControl.Status.ErrorID := MC_CamTableSelect_0.ErrorID;
                            MC_CamTableSelect_0.Execute := FALSE;
                            AxisStep := STATE_ERROR;
                    END_IF
            (******************** START CAM MOVEMENT **********************)
            STATE_CAM_START: (* STATE: Start electronic gear coupling *)
                    MC_CamIn_0.MasterOffset := 0;
                    MC_CamIn_0.SlaveOffset := 0;
                    MC_CamIn_0.MasterScaling := moteurL.status.MasterScaling;
                    MC_CamIn_0.SlaveScaling := moteurL.status.SlaveScaling;
                    MC_CamIn_0.StartMode := CamControl.Parameter.StartMode;
                    MC_CamIn_0.CamTableID := CamTableID;
                    MC_CamIn_0.Execute := TRUE;
                    (* wait for CAM stop *)

//      IF (CamControl.Command.DisengageSlave = TRUE) THEN
//                            CamControl.Command.DisengageSlave := FALSE;
//                            MC_CamIn_0.Execute := FALSE;
//                            AxisStep := STATE_CAM_STOP;
//                END_IF
                    (* check if error occured *)
                    IF (MC_CamIn_0.Error = TRUE) THEN
                            CamControl.Status.ErrorID := MC_CamIn_0.ErrorID;
                            MC_CamIn_0.Execute := FALSE;
                            AxisStep := STATE_ERROR;
                    END_IF

(*Can't be used for any linear with a back and forth motion*)
//              (******************** STOP CAM MOVEMENT **********************)
```

```
//              STATE_CAM_STOP: (* STATE: Stop electronic gear coupling *)
//                      MC_CamOut_0.Execute := TRUE;
//                      (* check if coupling is stopped *)
//                      IF (MC_CamOut_0.Done = TRUE) THEN
//                              MC_CamOut_0.Execute := FALSE;
//                              AxisStep := STATE_READY;
//                      END_IF
//                      (* check if error occured *)
//                      IF (MC_CamOut_0.Error = TRUE) THEN
//                              CamControl.Status.ErrorID := MC_CamOut_0.ErrorID;
//                              MC_CamOut_0.Execute := FALSE;
//                              AxisStep := STATE_ERROR;
//                      END_IF
                (********************* FB-ERROR OCCURED ************************)
                STATE_ERROR: (* STATE: Error *)
                        (* check if FB indicates an axis error *)
                        IF (MC_ReadAxisError_0.AxisErrorCount<>0) THEN
                                AxisStep := STATE_ERROR_AXIS;
                        ELSE
                                IF (CamControl.Command.ErrorAcknowledge = TRUE) THEN
                                        CamControl.Command.ErrorAcknowledge := FALSE;
                                        CamControl.Status.ErrorID := FALSE;
                                        (* reset axis if it is in axis state ErrorStop *)
                        IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE)) THEN
                                                AxisStep := STATE_ERROR_RESET;
                                        ELSE
                                                AxisStep := STATE_WAIT;
                                        END_IF
                                END_IF
                        END_IF
```

```
(******************** AXIS-ERROR OCCURED ************************)
STATE_ERROR_AXIS: (* STATE: Axis Error *)
        IF (MC_ReadAxisError_0.Valid = TRUE) THEN
                IF (MC_ReadAxisError_0.AxisErrorID <> 0) THEN
                        CamControl.Status.ErrorID := MC_ReadAxisError_0.AxisErrorID;
                END_IF
                MC_ReadAxisError_0.Acknowledge := FALSE;
                IF (CamControl.Command.ErrorAcknowledge = TRUE) THEN
                        CamControl.Command.ErrorAcknowledge := FALSE;
                        (* acknowledge axis error *)
                        IF (MC_ReadAxisError_0.AxisErrorID <> 0) THEN
                                MC_ReadAxisError_0.Acknowledge := TRUE;
                        END_IF
                END_IF
                IF (MC_ReadAxisError_0.AxisErrorCount = 0) THEN
                        (* reset axis if it is in axis state ErrorStop *)
                        CamControl.Status.ErrorID := 0;
        IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE)) THEN
                                AxisStep := STATE_ERROR_RESET;
                        ELSE
                                AxisStep := STATE_WAIT;
                        END_IF
                END_IF
        END_IF
(******************** RESET DONE ************************)
STATE_ERROR_RESET: (* STATE: Wait for reset done *)
        MC_Reset_0.Execute := TRUE;
        (* reset MC_Power.Enable if this FB is in Error*)
        IF (MC_Power_0.Error = TRUE) THEN
                MC_Power_0.Enable := FALSE;
```

```
                    END_IF
                    IF(MC_Reset_0.Done = TRUE)THEN
                        MC_Reset_0.Execute := FALSE;
                        AxisStep := STATE_WAIT;
                    END_IF
        END_CASE

(*************************************************************
    Function Block Calls
*************************************************************)
(************************* MC_POWER ***************************)
        MC_Power_0.Axis := Axis2Obj; (* pointer to axis *)
        MC_Power_0();
        (************************* MC_HOME ****************************)
        MC_Home_0.Axis := Axis2Obj;
        MC_Home_0();
        (********************** MC_CAMTABLESELECT ************************)
        MC_CamTableSelect_0.Master:= Axis1Obj;
        MC_CamTableSelect_0.Slave:= Axis2Obj;
        MC_CamTableSelect_0();
        (********************** MC_CAMIN ************************)
        MC_CamIn_0.Master := Axis1Obj;
        MC_CamIn_0.Slave := Axis2Obj;
        MC_CamIn_0();
        (**************************MC_HALT*****************************)
        MC_Halt_0.Axis := Axis2Obj;
        MC_Halt_0();

//      (************************* MC_CAM_OUT ****************************)
//      MC_CamOut_0.Slave := Axis2Obj;
```

```
//      MC_Stop_0();
        (************************* MC_RESET ***************************)
        MC_Reset_0.Axis := Axis2Obj;
        MC_Reset_0();
        (************************MC_READAXISERROR **********************)
        MC_ReadAxisError_0.Enable := NOT(MC_ReadAxisError_0.Error);
        MC_ReadAxisError_0.Axis := Axis2Obj;
        MC_ReadAxisError_0.DataAddress := ADR(CamControl.Status.ErrorText);
        MC_ReadAxisError_0.DataLength := SIZEOF(CamControl.Status.ErrorText);
        MC_ReadAxisError_0.DataObjectName := 'acp10etxen';
        MC_ReadAxisError_0();
        (************************ MC_READSTATUS ************************)
        MC_ReadStatus_0.Enable := NOT(MC_ReadStatus_0.Error);
        MC_ReadStatus_0.Axis := Axis2Obj;
        MC_ReadStatus_0();
        CamControl.AxisState.Disabled           := MC_ReadStatus_0.Disabled;
        CamControl.AxisState.StandStill         := MC_ReadStatus_0.StandStill;
        CamControl.AxisState.Stopping           := MC_ReadStatus_0.Stopping;
        CamControl.AxisState.Homing             := MC_ReadStatus_0.Homing;
        CamControl.AxisState.DiscreteMotion     := MC_ReadStatus_0.DiscreteMotion;
        CamControl.AxisState.ContinuousMotion   := MC_ReadStatus_0.ContinuousMotion;
        CamControl.AxisState.SynchronizedMotion := MC_ReadStatus_0.SynchronizedMotion;
        CamControl.AxisState.ErrorStop          := MC_ReadStatus_0.Errorstop;
        (********************MC_BR_READDRIVESTATUS***********************)
        MC_BR_ReadDriveStatus_0.Enable := NOT(MC_BR_ReadDriveStatus_0.Error);
        MC_BR_ReadDriveStatus_0.Axis := Axis2Obj;
        MC_BR_ReadDriveStatus_0.AdrDriveStatus := ADR(CamControl.Status.DriveStatus);
        MC_BR_ReadDriveStatus_0();
        (******************** MC_READACTUALPOSITION *********************)
        MC_ReadActualPosition_0.Enable := (NOT(MC_ReadActualPosition_0.Error));
```

```
MC_ReadActualPosition_0.Axis := Axis2Obj;
MC_ReadActualPosition_0();
IF(MC_ReadActualPosition_0.Valid = TRUE)THEN
        CamControl.Status.ActPosition := MC_ReadActualPosition_0.Position;
END_IF
(******************** MC_READACTUALTORQUE **********************)
MC_ReadActualTorque_0.Enable := (NOT(MC_ReadActualTorque_0.Error));
MC_ReadActualTorque_0.Axis := Axis2Obj;
MC_ReadActualTorque_0();
IF(MC_ReadActualTorque_0.Valid = TRUE)THEN
        CamControl.Status.ActTorque := MC_ReadActualTorque_0.Torque;
END_IF
(******************** MC_READACTUALVELOCITY **********************)
MC_ReadActualVelocity_0.Enable := (NOT(MC_ReadActualVelocity_0.Error));
MC_ReadActualVelocity_0.Axis := Axis2Obj;
MC_ReadActualVelocity_0();
IF(MC_ReadActualVelocity_0.Valid = TRUE)THEN
        CamControl.Status.ActVelocity := MC_ReadActualVelocity_0.Velocity;
END_IF
```

**END_PROGRAM**

### 3. PROGRAM AXE VIRTUEL

**PROGRAM INIT**

```
(* get axis object *)
Axis1Obj := ADR(Vaxe1);

AxisStep := STATE_WAIT; (* start step *)
```

**END_PROGRAM**

**PROGRAM _CYCLIC**

```
(*************************************************************

      Control Sequence

*************************************************************)

(**************** CHECK FOR GENERAL AXIS ERROR ******************)

   IF ((MC_ReadAxisError_0.AxisErrorID <> 0) AND (MC_ReadAxisError_0.Valid = TRUE)) THEN

      AxisStep := STATE_ERROR_AXIS;

(***************** CHECK IF POWER SHOULD BE OFF *******************)

   ELSIF (BasicControl.Command.Power = FALSE) THEN

      IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE)) THEN

         AxisStep := STATE_ERROR_RESET;

      ELSE

         AxisStep := STATE_WAIT;

      END_IF

   END_IF

CASE AxisStep OF

(******************** WAIT *************************)

   STATE_WAIT:  (* STATE: Wait *)

      IF (BasicControl.Command.Power = TRUE) THEN

         AxisStep := STATE_POWER_ON;

      ELSE

         MC_Power_0.Enable := FALSE;

      END_IF

          (* reset all FB execute inputs we use *)


                           MC_Home_0.Execute := FALSE;

                           MC_Stop_0.Execute := FALSE;

//                    MC_MoveAbsolute_0.Execute := FALSE;

//                           MC_MoveAdditive_0.Execute := FALSE;

                           MC_MoveVelocity_0.Execute := FALSE;

                           MC_ReadAxisError_0.Acknowledge := FALSE;
```

```
                        MC_Reset_0.Execute := FALSE;
                        (* reset user commands *)
                        BasicControl.Command.Stop := FALSE;
                        BasicControl.Command.Home := FALSE;
                        BasicControl.Command.MoveVelocity := FALSE;
//                      BasicControl.Command.MoveAbsolute := FALSE;
        //              BasicControl.Command.MoveAdditive := FALSE;
                        BasicControl.Status.ErrorID := 0;


(******************** POWER ON **********************)
   STATE_POWER_ON:  (* STATE: Power on *)
      MC_Power_0.Enable := TRUE;
      IF (MC_Power_0.Status = TRUE) THEN
         AxisStep := STATE_READY;
      END_IF
      (* if a power error occured go to error state *)
      IF (MC_Power_0.Error = TRUE) THEN
         BasicControl.Status.ErrorID := MC_Power_0.ErrorID;
         AxisStep := STATE_ERROR;
      END_IF
(******************** READY **********************)
   STATE_READY:  (* STATE: Waiting for commands *)
      IF (BasicControl.Command.Home = TRUE)THEN
         BasicControl.Command.Home := FALSE;
         AxisStep := STATE_HOME;
      ELSIF (BasicControl.Command.Stop = TRUE) THEN
         AxisStep := STATE_STOP;



//     ELSIF (BasicControl.Command.MoveAbsolute = TRUE) THEN
```

```
//          BasicControl.Command.MoveAbsolute := FALSE;
//          AxisStep := STATE_MOVE_ABSOLUTE;
//      ELSIF (BasicControl.Command.MoveAdditive = TRUE) THEN
//          BasicControl.Command.MoveAdditive := FALSE;
//          AxisStep := STATE_MOVE_ADDITIVE;
        ELSIF (BasicControl.Command.MoveVelocity = TRUE) THEN
            BasicControl.Command.MoveVelocity := FALSE;
            AxisStep := STATE_MOVE_VELOCITY;
//        ELSIF (BasicControl.Command.Halt = TRUE) THEN
//          BasicControl.Command.Halt := FALSE;
//          AxisStep := STATE_HALT;
        END_IF
(******************** HOME **********************)
    STATE_HOME:  (* STATE: start homing process *)
        MC_Home_0.Position := BasicControl.Parameter.HomePosition;
        MC_Home_0.HomingMode := mcHOME_DIRECT;
        MC_Home_0.Execute := TRUE;
        IF (BasicControl.Command.Stop = TRUE) THEN
            MC_Home_0.Execute := FALSE;
            AxisStep := STATE_STOP;
        ELSIF (MC_Home_0.Done = TRUE) THEN
            MC_Home_0.Execute := FALSE;
            AxisStep := STATE_READY;
        END_IF
        (* if a homing error occured go to error state *)
        IF (MC_Home_0.Error = TRUE) THEN
            MC_Home_0.Execute := FALSE;
            BasicControl.Status.ErrorID := MC_Home_0.ErrorID;
            AxisStep := STATE_ERROR;
        END_IF
```

```
 (**********************HALT_MOVEMENT**************************)
//   STATE_HALT:   (* STATE: Halt movement *)
//      MC_Halt_0.Deceleration := BasicControl.Parameter.Deceleration;
//      MC_Halt_0.Execute := TRUE;
//      IF (MC_Halt_0.Done = TRUE) THEN
//         MC_Halt_0.Execute := FALSE;
//         AxisStep := STATE_READY;
//      END_IF
//      (* check if error occured *)
//      IF (MC_Halt_0.Error = TRUE) THEN
//         BasicControl.Status.ErrorID := MC_Halt_0.ErrorID;
//         MC_Halt_0.Execute := FALSE;
//         AxisStep := STATE_ERROR;
//      END_IF
(********************** STOP MOVEMENT ************************)
   STATE_STOP:  (* STATE: Stop movement *)
      MC_Stop_0.Deceleration := BasicControl.Parameter.Deceleration;
      MC_Stop_0.Execute := TRUE;
      (* if axis is stopped go to ready state *)
      IF ((MC_Stop_0.Done = TRUE) AND (BasicControl.Command.Stop = FALSE)) THEN
         MC_Stop_0.Execute := FALSE;
         AxisStep := STATE_READY;
      END_IF
      (* check if error occured *)
      IF (MC_Stop_0.Error = TRUE) THEN
         BasicControl.Status.ErrorID := MC_Stop_0.ErrorID;
         MC_Stop_0.Execute := FALSE;
         AxisStep := STATE_ERROR;
      END_IF
//   ****************** START ABSOLUTE MOVEMENT ********************
```

```
//    STATE_MOVE_ABSOLUTE:  (* STATE: Start absolute movement *)
//       MC_MoveAbsolute_0.Position     := BasicControl.Parameter.Position;
//       MC_MoveAbsolute_0.Velocity     := 360
//       MC_MoveAbsolute_0.Acceleration  := moteurL.status.Acceleration;
//       MC_MoveAbsolute_0.Deceleration  := moteurL.status.Deceleration;
//       MC_MoveAbsolute_0.Direction    := BasicControl.Parameter.Direction;
//       MC_MoveAbsolute_0.Execute := TRUE;
//       (* check if commanded position is reached *)
//       IF (BasicControl.Command.Stop = TRUE) THEN
//          MC_MoveAbsolute_0.Execute := FALSE;
//          AxisStep := STATE_STOP;
//       ELSIF (BasicControl.Command.Halt) THEN
//          BasicControl.Command.Halt := FALSE;
//          MC_MoveAbsolute_0.Execute := FALSE;
//          AxisStep := STATE_HALT;
//       ELSIF (MC_MoveAbsolute_0.Done = TRUE) THEN
//          MC_MoveAbsolute_0.Execute := FALSE;
//          AxisStep := STATE_READY;
//       END_IF
//       (* check if error occured *)
//       IF (MC_MoveAbsolute_0.Error = TRUE) THEN
//          BasicControl.Status.ErrorID := MC_MoveAbsolute_0.ErrorID;
//          MC_MoveAbsolute_0.Execute := TRUE;
//          AxisStep := STATE_ERROR;
//       END_IF
// (******************** START ADDITIVE MOVEMENT **********************)
//    STATE_MOVE_ADDITIVE:  (* STATE: Start additive movement *)
//       MC_MoveAdditive_0.Distance     := BasicControl.Parameter.Distance;
//       MC_MoveAdditive_0.Velocity     := 360;
//       MC_MoveAdditive_0.Acceleration  := moteurL.status.Acceleration;
```

```
//      MC_MoveAdditive_0.Deceleration  := moteurL.status.Deceleration;
//      MC_MoveAdditive_0.Execute := TRUE;
//      (* check if commanded distance is reached *)
//      IF (BasicControl.Command.Stop = TRUE) THEN
//          MC_MoveAdditive_0.Execute := FALSE;
//          AxisStep := STATE_STOP;
//      ELSIF (BasicControl.Command.Halt) THEN
//          BasicControl.Command.Halt := FALSE;
//          MC_MoveAdditive_0.Execute := FALSE;
//          AxisStep := STATE_HALT;
//      ELSIF (MC_MoveAdditive_0.Done = TRUE) THEN
//          MC_MoveAdditive_0.Execute := FALSE;
//          AxisStep := STATE_READY;
//      END_IF
//      (* check if error occured *)
//      IF (MC_MoveAdditive_0.Error = TRUE) THEN
//          BasicControl.Status.ErrorID := MC_MoveAdditive_0.ErrorID;
//          MC_MoveAdditive_0.Execute := FALSE;
//          AxisStep := STATE_ERROR;
//      END_IF
(******************** START VELOCITY MOVEMENT **********************)
   STATE_MOVE_VELOCITY:  (* STATE: Start velocity movement *)
      MC_MoveVelocity_0.Velocity     := 360;
      MC_MoveVelocity_0.Acceleration  := moteurL.status.Acceleration;
      MC_MoveVelocity_0.Deceleration  := moteurL.status.Deceleration;
      MC_MoveVelocity_0.Direction     := BasicControl.Parameter.Direction;
      MC_MoveVelocity_0.Execute := TRUE;
      (* check if commanded velocity is reached *)
      IF (BasicControl.Command.Stop = TRUE) THEN
         MC_MoveVelocity_0.Execute := FALSE;
```

```
        AxisStep := STATE_STOP;
    ELSIF (BasicControl.Command.Halt) THEN
        BasicControl.Command.Halt := FALSE;
        MC_MoveVelocity_0.Execute := FALSE;
        AxisStep := STATE_HALT;
    ELSIF (MC_MoveVelocity_0.InVelocity = TRUE) THEN
        MC_MoveVelocity_0.Execute := FALSE;
        AxisStep := STATE_READY;
    END_IF
    (* check if error occured *)
    IF (MC_MoveVelocity_0.Error = TRUE) THEN
        BasicControl.Status.ErrorID := MC_MoveVelocity_0.ErrorID;
        MC_MoveVelocity_0.Execute := FALSE;
        AxisStep := STATE_ERROR;
    END_IF
(******************** FB-ERROR OCCURED ************************)
  STATE_ERROR:  (* STATE: Error *)
    (* check if FB indicates an axis error *)
    IF (MC_ReadAxisError_0.AxisErrorCount<>0) THEN
        AxisStep := STATE_ERROR_AXIS;
    ELSE
        IF (BasicControl.Command.ErrorAcknowledge = TRUE) THEN
            BasicControl.Command.ErrorAcknowledge := FALSE;
            BasicControl.Status.ErrorID := 0;
            (* reset axis if it is in axis state ErrorStop *)
            IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE)) THEN
                AxisStep := STATE_ERROR_RESET;
            ELSE
                AxisStep := STATE_WAIT;
            END_IF
```

```
        END_IF
      END_IF
(******************** AXIS-ERROR OCCURED ************************)
   STATE_ERROR_AXIS:  (* STATE: Axis Error *)
      IF (MC_ReadAxisError_0.Valid = TRUE) THEN
        IF (MC_ReadAxisError_0.AxisErrorID <> 0) THEN
          BasicControl.Status.ErrorID := MC_ReadAxisError_0.AxisErrorID;
        END_IF
        MC_ReadAxisError_0.Acknowledge := FALSE;
        IF (BasicControl.Command.ErrorAcknowledge = TRUE) THEN
          BasicControl.Command.ErrorAcknowledge := FALSE;
          (* acknowledge axis error *)
          IF (MC_ReadAxisError_0.AxisErrorID <> 0) THEN
            MC_ReadAxisError_0.Acknowledge := TRUE;
          END_IF
        END_IF
        IF (MC_ReadAxisError_0.AxisErrorCount = 0) THEN
          (* reset axis if it is in axis state ErrorStop *)
          BasicControl.Status.ErrorID := 0;
          IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE)) THEN
            AxisStep := STATE_ERROR_RESET;
          ELSE
            AxisStep := STATE_WAIT;
          END_IF
        END_IF
      END_IF
(******************** RESET DONE ************************)
   STATE_ERROR_RESET:  (* STATE: Wait for reset done *)
      MC_Reset_0.Execute := TRUE;
      (* reset MC_Power.Enable if this FB is in Error*)
```

```
    IF (MC_Power_0.Error = TRUE) THEN
      MC_Power_0.Enable := FALSE;
    END_IF
    IF(MC_Reset_0.Done = TRUE)THEN
      MC_Reset_0.Execute := FALSE;
      AxisStep := STATE_WAIT;
    END_IF
(******************** SEQUENCE END ************************)
END_CASE
(*************************************************************
    Function Block Calls
*************************************************************)
(************************** MC_POWER ***************************)
MC_Power_0.Axis := Axis1Obj;  (* pointer to axis *)
MC_Power_0();
(************************** MC_HOME ****************************)
MC_Home_0.Axis := Axis1Obj;
MC_Home_0();
//(********************** MC_MOVEABSOLUTE *************************)
//MC_MoveAbsolute_0.Axis := Axis1Obj;
//MC_MoveAbsolute_0();
//********************** MC_MOVEADDITIVE ************************
//MC_MoveAdditive_0.Axis := Axis1Obj;
//MC_MoveAdditive_0();
(********************** MC_MOVEVELOCITY *************************)
MC_MoveVelocity_0.Axis := Axis1Obj;
MC_MoveVelocity_0();

(************************** MC_STOP ****************************)
MC_Stop_0.Axis := Axis1Obj;
```

```
MC_Stop_0();
(**************************MC_HALT*******************************)
MC_Halt_0.Axis := Axis1Obj;
MC_Halt_0();
(************************** MC_RESET ***************************)
MC_Reset_0.Axis := Axis1Obj;
MC_Reset_0();
(************************ MC_READAXISERROR ***********************)
MC_ReadAxisError_0.Enable := NOT(MC_ReadAxisError_0.Error);
MC_ReadAxisError_0.Axis := Axis1Obj;
MC_ReadAxisError_0.DataAddress := ADR(BasicControl.Status.ErrorText);
MC_ReadAxisError_0.DataLength := SIZEOF(BasicControl.Status.ErrorText);
MC_ReadAxisError_0.DataObjectName := 'acp10etxen';
MC_ReadAxisError_0();
(************************ MC_READSTATUS ************************)
MC_ReadStatus_0.Enable := NOT(MC_ReadStatus_0.Error);
MC_ReadStatus_0.Axis := Axis1Obj;
MC_ReadStatus_0();
BasicControl.AxisState.Disabled          := MC_ReadStatus_0.Disabled;
BasicControl.AxisState.StandStill        := MC_ReadStatus_0.StandStill;
BasicControl.AxisState.Stopping          := MC_ReadStatus_0.Stopping;
BasicControl.AxisState.Homing            := MC_ReadStatus_0.Homing;
BasicControl.AxisState.DiscreteMotion    := MC_ReadStatus_0.DiscreteMotion;
BasicControl.AxisState.ContinuousMotion  := MC_ReadStatus_0.ContinuousMotion;
BasicControl.AxisState.SynchronizedMotion := MC_ReadStatus_0.SynchronizedMotion;
BasicControl.AxisState.ErrorStop         := MC_ReadStatus_0.Errorstop;
(********************MC_BR_READDRIVESTATUS***********************)
MC_BR_ReadDriveStatus_0.Enable := NOT(MC_BR_ReadDriveStatus_0.Error);
MC_BR_ReadDriveStatus_0.Axis := Axis1Obj;
MC_BR_ReadDriveStatus_0.AdrDriveStatus := ADR(BasicControl.Status.DriveStatus);
```

```
MC_BR_ReadDriveStatus_0();
(******************** MC_READACTUALPOSITION **********************)
MC_ReadActualPosition_0.Enable := (NOT(MC_ReadActualPosition_0.Error));
MC_ReadActualPosition_0.Axis := Axis1Obj;
MC_ReadActualPosition_0();
IF(MC_ReadActualPosition_0.Valid = TRUE)THEN
   BasicControl.Status.ActPosition := MC_ReadActualPosition_0.Position;
END_IF
(******************** MC_READACTUALVELOCITY **********************)
MC_ReadActualVelocity_0.Enable := (NOT(MC_ReadActualVelocity_0.Error));
MC_ReadActualVelocity_0.Axis := Axis1Obj;
MC_ReadActualVelocity_0();
IF(MC_ReadActualVelocity_0.Valid = TRUE)THEN
   BasicControl.Status.ActVelocity := MC_ReadActualVelocity_0.Velocity;
END_IF
END_PROGRAM
```

### 4. PROGRAM ALIM

```
PROGRAM _INIT
        axis3Obj:=      ADR(alim);
        STEP:=STATE_WAIT;
(*timing*)


END_PROGRAM


PROGRAM _CYCLIC
        (*************** CHECK FOR GENERAL AXIS ERROR ON THE ALIM AXE *****************)
```

```
        IF ((MC_ReadAxisError_0.AxisErrorID <> 0) AND (MC_ReadAxisError_0.Valid = TRUE)) THEN
                STEP:= STATE_ERROR_AXIS;
                (***************** CHECK IF POWER SHOULD BE OFF ******************)
           ELSIF (ALIM.Command.Alim = FALSE ) THEN
                IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE)) THEN
                        STEP := STATE_ERROR_RESET;
                ELSE
                        STEP := STATE_WAIT;
                END_IF
        END_IF

CASE STEP OF
(***************** ALIM *****************)

STATE_WAIT :
                IF (ALIM.Command.Alim = TRUE) THEN
                        STEP := STATE_ALIM_ON;
                ELSE
                        MC_PowerAlim.Enable:=FALSE;
                END_IF
                        MC_BR_PowerMeter_Rolf.Enable:=FALSE;
                        MC_ReadAxisError_0.Acknowledge:=FALSE;
                        MC_Reset_0.Execute:= FALSE;
                        ALIM.Command.PowerMeter:=FALSE;
                        ALIM.Status.ErrorID:=0;

STATE_ALIM_ON:
                        MC_PowerAlim.Enable := TRUE;
                        IF (MC_PowerAlim.Status = TRUE) THEN
```

```
                    STEP := STATE_READY;
                END_IF
                (* if a power error occured go to error state *)
                IF (MC_PowerAlim.Error = TRUE) THEN
                    ALIM.Status.ErrorID := MC_PowerAlim.ErrorID;
                    STEP := STATE_ERROR;
                END_IF

STATE_READY:
                    IF (ALIM.Command.PowerMeter = TRUE) THEN
//                  ALIM.Command.PowerMeter := FALSE;
                  STEP := STATE_POWER_METER;
            END_IF

STATE_POWER_METER :
                ALIM.Status.MaximumActivePower   :=
MC_BR_PowerMeter_Rolf.PowerData.MaximumActivePower;
                ALIM.Status.ConsumedEnergy      := MC_BR_PowerMeter_Rolf.PowerData.ConsumedEnergy;
                ALIM.Status.AverageReactivePower :=
MC_BR_PowerMeter_Rolf.PowerData.AverageReactivePower;
                ALIM.Status.EnergyBalance       := MC_BR_PowerMeter_Rolf.PowerData.EnergyBalance;
                ALIM.Status.AverageActivePower   := -
(MC_BR_PowerMeter_Rolf.PowerData.AverageActivePower);
                ALIM.Status.MinimalActivePower   :=
MC_BR_PowerMeter_Rolf.PowerData.MinimalActivePower;
                ALIM.Status.RegeneratedEnergy    :=
MC_BR_PowerMeter_Rolf.PowerData.RegeneratedEnergy;

//// PROG TO DETERMINE IF THE MODE IS GENERATOR vOR MOTRO AND TO DISPLAY THEM ON THE POWER PANEL.

 // Test the mode (ABSORBE OR PRODUCE POWER)
                IF (ALIM.Status.MaximumActivePower - ALIM.Status.MinimalActivePower) >= 0
```

```
                    THEN ALIM.Status.InversePowerMin := ABS(ALIM.Status.MaximumActivePower);
                    ALIM.Status.InversePowerMax := ABS(ALIM.Status.MinimalActivePower);

                ELSE ALIM.Status.InversePowerMin := ALIM.Status.MaximumActivePower;
                    ALIM.Status.InversePowerMax := ALIM.Status.MinimalActivePower;
                END_IF
                // End of the test and display
                MC_BR_PowerMeter_Rolf.Enable :=TRUE;
                MC_BR_PowerMeter_Rolf.IntervalTime:=moteurL.status.TimeInterval;
                MC_BR_PowerMeter_Rolf.Mode:=mcONLY_PSM;
                MC_BR_PowerMeter_Rolf.RestartInterval:=0; (*temporaray set TO zero, no reset
possible...*)

            IF (MC_BR_PowerMeter_Rolf.Valid =TRUE) THEN
                    STEP := STATE_READY;
                    (*check IF error occured*)
            ELSIF (MC_BR_PowerMeter_Rolf.Error = TRUE) THEN
                    ALIM.Status.ErrorID := MC_BR_PowerMeter_Rolf.ErrorID;
                    MC_BR_PowerMeter_Rolf.Enable := FALSE;
                    STEP := STATE_ERROR;
             END_IF
            (*********  MANAGING THE  ERRORS  *********)
STATE_ERROR:
            (* check if FB indicates an axis error *)
            IF (MC_ReadAxisError_0.AxisErrorCount<>0) THEN
                STEP := STATE_ERROR_AXIS;
            ELSE
                IF (ALIM.Command.ErrorAcknowledge = TRUE) THEN
                    ALIM.Command.ErrorAcknowledge := FALSE;
```

```
                ALIM.Status.ErrorID := FALSE;
                (* reset axis if it is in axis state ErrorStop *)
            IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE)) THEN
                    STEP := STATE_ERROR_RESET;
                ELSE
                    STEP := STATE_WAIT;
                END_IF
            END_IF
        END_IF
    (******************* AXIS-ERROR OCCURED ************************)
    STATE_ERROR_AXIS: (* STATE: Axis Error *)
        IF (MC_ReadAxisError_0.Valid = TRUE) THEN
            IF (MC_ReadAxisError_0.AxisErrorID <> 0) THEN
                ALIM.Status.ErrorID := MC_ReadAxisError_0.AxisErrorID;
            END_IF
            MC_ReadAxisError_0.Acknowledge := FALSE;
            IF (ALIM.Command.ErrorAcknowledge = TRUE) THEN
                ALIM.Command.ErrorAcknowledge := FALSE;
                (* acknowledge axis error *)
                IF (MC_ReadAxisError_0.AxisErrorID <> 0) THEN
                    MC_ReadAxisError_0.Acknowledge := TRUE;
                END_IF
            END_IF
            IF (MC_ReadAxisError_0.AxisErrorCount = 0) THEN
                (* reset axis if it is in axis state ErrorStop *)
                ALIM.Status.ErrorID := 0;
                IF ((MC_ReadStatus_0.Errorstop = TRUE) AND (MC_ReadStatus_0.Valid = TRUE))
THEN
                    STEP := STATE_ERROR_RESET;
                ELSE
```

```
                                    STEP := STATE_WAIT;
                                END_IF
                            END_IF
                        END_IF
        (******************** RESET DONE ************************)
        STATE_ERROR_RESET: (* STATE: Wait for reset done *)
                MC_Reset_0.Execute := TRUE;
                (* reset MC_Power.Enable if this FB is in Error*)
                IF (MC_PowerAlim.Error = TRUE) THEN
                        MC_PowerAlim.Enable := FALSE;
                END_IF
                IF(MC_Reset_0.Done = TRUE)THEN
                        MC_Reset_0.Execute := FALSE;
                        STEP := STATE_WAIT;
                END_IF
END_CASE
        (************************* MC_PowerAlim ***********************)
        MC_PowerAlim.Axis := axis3Obj;
        MC_PowerAlim();
        (************************** MC_BR_PowerMeter ***********************)
        MC_BR_PowerMeter_Rolf.Axis := ADR(alim);
        MC_BR_PowerMeter_Rolf();
        (************************MC_READAXISERROR *********************)
        MC_ReadAxisError_0.Enable := NOT(MC_ReadAxisError_0.Error);
        MC_ReadAxisError_0.Axis := ADR(alim);
        MC_ReadAxisError_0.DataAddress := ADR(ALIM.Status.ErrorText);
        MC_ReadAxisError_0.DataLength := SIZEOF(ALIM.Status.ErrorText);
        MC_ReadAxisError_0.DataObjectName := 'acp10etxen';
        MC_ReadAxisError_0();
        (********************** MC_READSTATUS ************************)
```

```
MC_ReadStatus_0.Enable := NOT(MC_ReadStatus_0.Error);
MC_ReadStatus_0.Axis := ADR(alim);
MC_ReadStatus_0();
CamControl.AxisState.Disabled          := MC_ReadStatus_0.Disabled;
CamControl.AxisState.StandStill        := MC_ReadStatus_0.StandStill;
CamControl.AxisState.Stopping          := MC_ReadStatus_0.Stopping;
CamControl.AxisState.Homing            := MC_ReadStatus_0.Homing;
CamControl.AxisState.DiscreteMotion    := MC_ReadStatus_0.DiscreteMotion;
CamControl.AxisState.ContinuousMotion  := MC_ReadStatus_0.ContinuousMotion;
CamControl.AxisState.SynchronizedMotion := MC_ReadStatus_0.SynchronizedMotion;
CamControl.AxisState.ErrorStop         := MC_ReadStatus_0.Errorstop;
```

**END_PROGRAM**

## 5. PROGRAM POUR FAIRE VARIER LES PARAMETRES

```
(*********************************************************************
 * COPYRIGHT -- cnam
 *********************************************************************
 * Program: NewProgram
 * File: NewProgramCyclic.st
 * Author: Rolf Pérot
 * Created: November 29, 2012
 *********************************************************************
 * Implementation of program NewProgram
 *********************************************************************)
ACTION command:

//ACCELERATION WHEN THE MASTER AXIS IS SET ON
IF moteurL.command.DecreaseAcceleration THEN
```

```
IF moteurL.status.Acceleration - moteurL.parameter.AccelerationAccuracy >= moteurL.parameter.AccelerationMin
THEN

moteurL.status.Acceleration := moteurL.status.Acceleration - moteurL.parameter.AccelerationAccuracy;

                    END_IF

                END_IF

IF moteurL.command.IncreaseAcceleration THEN

IF moteurL.status.Acceleration + moteurL.parameter.AccelerationAccuracy <= moteurL.parameter.AccelerationMax
THEN

moteurL.status.Acceleration := moteurL.status.Acceleration + moteurL.parameter.AccelerationAccuracy;

                    END_IF

                END_IF


//DECELERATION WHEN THE MASTER AXIS IS SET OFF

IF moteurL.command.DecreaseDeceleration  THEN

IF moteurL.status.Deceleration - moteurL.parameter.DecelerationAccuracy >= moteurL.parameter.DecelerationMin
THEN

moteurL.status.Deceleration  := moteurL.status.Deceleration - moteurL.parameter.DecelerationAccuracy;

                    END_IF

                END_IF

IF moteurL.command.IncreaseDeceleration THEN

IF moteurL.status.Deceleration  + moteurL.parameter.DecelerationAccuracy <= moteurL.parameter.DecelerationMax
THEN

moteurL.status.Deceleration := moteurL.status.Deceleration + moteurL.parameter.DecelerationAccuracy;

                    END_IF

                END_IF

// Master,Frequency Scaling

// The way to monitor this parameter is to act on the MasterScale axis. (f= Speed set for the
virtual/MasterScaling=360/MasterScaling)


IF moteurL.command.IncreaseFrequency THEN

IF moteurL.status.Frequency  + moteurL.parameter.ScalingAccuracyFrequency <= moteurL.parameter.FrequencyMax
THEN
```

```
moteurL.status.Frequency := moteurL.status.Frequency + moteurL.parameter.ScalingAccuracyFrequency;

moteurL.status.MasterScaling := (360/moteurL.status.Frequency);

                END_IF

        END_IF

IF moteurL.command.DecreaseFrequency THEN

IF moteurL.status.Frequency - moteurL.parameter.ScalingAccuracyFrequency >= moteurL.parameter.FrequencyMin
THEN

moteurL.status.Frequency := moteurL.status.Frequency - moteurL.parameter.ScalingAccuracyFrequency;

moteurL.status.MasterScaling := (360/moteurL.status.Frequency);

                END_IF

        END_IF

        // Slave,Distance scaling

        //// The way to monitor this parameter is to act on the SlaveScale axis. 1 mm is equal to 196 units

        IF moteurL.command.IncreaseDistance THEN

                IF moteurL.status.Distance + moteurL.parameter.ScalingAccuracyDistance <=
moteurL.parameter.DistanceMax THEN

                        moteurL.status.Distance  := moteurL.status.Distance +
moteurL.parameter.ScalingAccuracyDistance;

                        moteurL.status.SlaveScaling := (moteurL.status.Distance*19.6);

                END_IF

        END_IF

        IF moteurL.command.DecreaseDistance THEN

                IF moteurL.status.Distance - moteurL.parameter.ScalingAccuracyDistance >=
moteurL.parameter.DistanceMin THEN

                        moteurL.status.Distance := moteurL.status.Distance -
moteurL.parameter.ScalingAccuracyDistance;

                        moteurL.status.SlaveScaling := (moteurL.status.Distance*19.6);

                END_IF

        END_IF


// TIME INTERVAL FOR THE MEASURE OF THE DIFFERENT POWERS

IF moteurL.command.IncreaseTimeInterval THEN
```

```
IF moteurL.status.TimeInterval  + moteurL.parameter.ScalingInterval <= moteurL.parameter.ScalingIntervalMax
THEN

moteurL.status.TimeInterval  := moteurL.status.TimeInterval + moteurL.parameter.ScalingInterval;

                END_IF

        END_IF

IF moteurL.command.DecreaseTimeInterval THEN

IF moteurL.status.TimeInterval  - moteurL.parameter.ScalingInterval >= moteurL.parameter.ScalingIntervalMin THEN

moteurL.status.TimeInterval  := moteurL.status.TimeInterval - moteurL.parameter.ScalingInterval;

                END_IF

        END_IF

//CHOOSING THE CAMTABLE

IF (moteurL.command.profile) AND NOT (moteurL.command.profileBis)  THEN

moteurL.status.profile:= 'profile';

ELSIF (moteurL.command.profileBis) AND NOT (moteurL.command.profile) THEN

moteurL.status.profile:= 'profileBis';

        END_IF


                END_ACTION
```

## 6. VARIABLE LOCAL DES PROGRAMMES

**PROGRAMME ALIM**

```
VAR

        MC_BR_PowerMeter_Rolf : MC_BR_PowerMeter;

        MC_ReadStatus_0 : MC_ReadStatus;

        MC_Reset_0 : MC_Reset;

        MC_ReadAxisError_0 : MC_ReadAxisError;

        MC_PowerAlim : MC_Power;

        axis3Obj : UDINT;

END_VAR
```

```
VAR CONSTANT
	STATE_READY : USINT := 9;
END_VAR
VAR
	STEP : UINT; (*current step in the step sequence*)
END_VAR
VAR CONSTANT
	STATE_POWER_METER : USINT := 78;
	STATE_WAIT : USINT := 77;
	STATE_ALIM_ON : USINT := 10; (*constant for a waiting step*)
	STATE_ERROR_RESET : USINT := 102; (*constant for the step in which the errors are reset*)
	STATE_ERROR_AXIS : USINT := 100; (*constant for the axis error step*)
	STATE_ERROR : USINT := 101; (*constant for the error step*)
END_VAR
```

**AXE VIRTUEL**

```
VAR
	MC_Home_0 : MC_Home; (*Function Block*)
	MC_MoveAbsolute_0 : MC_MoveAbsolute; (*Function Block*)
	MC_MoveAdditive_0 : MC_MoveAdditive; (*Function Block*)
	MC_MoveVelocity_0 : MC_MoveVelocity; (*Function Block*)
	MC_Power_0 : MC_Power; (*Function Block*)
	MC_ReadActualPosition_0 : MC_ReadActualPosition; (*Function Block *)
	MC_ReadActualVelocity_0 : MC_ReadActualVelocity; (*Function Block*)
	MC_ReadAxisError_0 : MC_ReadAxisError; (*Function Block*)
	MC_ReadStatus_0 : MC_ReadStatus; (*Function Block*)
	MC_Reset_0 : MC_Reset; (*Function Block*)
	MC_Stop_0 : MC_Stop; (*Function Block*)
	MC_BR_ReadDriveStatus_0 : MC_BR_ReadDriveStatus; (*Function Block*)
```

```
    MC_Halt_0 : MC_Halt; (*Function Block*)

    Axis1Obj : UDINT; (*axis reference*)

    AxisStep : UINT; (*current step in the step sequence*)

END_VAR

VAR CONSTANT

    STATE_READY : USINT := 10; (*constant for a waiting step*)

    STATE_POWER_ON : USINT := 11; (*constant for the step in which the controller is switched on*)

    STATE_WAIT : USINT := 12; (*constant for a waiting step*)

    STATE_HOME : USINT := 13; (*constant for the step in which the axis is homed*)

    STATE_MOVE_VELOCITY : USINT := 14; (*constant for the step in which a movement with defiened
velocity is started*)

    STATE_MOVE_ABSOLUTE : USINT := 15; (*constant for the step in which a movement to defiened
position is started*)

    STATE_MOVE_ADDITIVE : USINT := 16; (*constant for the step in which a movement with defiened
distance is started*)

    STATE_STOP : USINT := 17; (*constant for the step in which movements are stopped*)

    STATE_ERROR_RESET : USINT := 18; (*constant for the step in which the errors are reset*)

    STATE_ERROR_AXIS : USINT := 19; (*constant for the axis error step*)

    STATE_ERROR : USINT := 20; (*constant for the error step*)

    STATE_HALT : USINT := 21; (*constant for the step in which movements are stopped*)

END_VAR
```

**AXE REEL**

```
VAR

    MC_CamIn_0 : MC_CamIn; (*Function Block*)

    MC_CamOut_0 : MC_CamOut; (*Function Block*)

    MC_CamTableSelect_0 : MC_CamTableSelect; (*Function Block*)

    MC_Home_0 : MC_Home; (*Function Block*)

    MC_Power_0 : MC_Power; (*Function Block*)
```

```
        MC_ReadActualPosition_0 : MC_ReadActualPosition; (*Function Block*)
        MC_ReadActualTorque_0 : MC_ReadActualTorque;
        MC_ReadActualVelocity_0 : MC_ReadActualVelocity; (*Function Block*)
        MC_ReadAxisError_0 : MC_ReadAxisError; (*Function Block*)
        MC_ReadStatus_0 : MC_ReadStatus; (*Function Block*)
        MC_Reset_0 : MC_Reset; (*Function Block*)
        MC_Stop_0 : MC_Stop; (*Function Block*)
        MC_BR_ReadDriveStatus_0 : MC_BR_ReadDriveStatus; (*Function Block*)
        MC_Halt_0 : MC_Halt; (*Function Block*)
        Axis2Obj : UDINT; (*axis reference*)
        Axis1Obj : UDINT; (*axis reference*)
        AxisStep : UINT; (*current step in the step sequence*)
        CamTableID : USINT; (*cam table ID*)
END_VAR
VAR CONSTANT
        STATE_WAIT : USINT := 0; (*constant for a waiting step*)
        STATE_POWER_ON : USINT := 1; (*constant for the step in which the controller is switched on*)
        STATE_HOME : USINT := 2; (*constant for the step in which the axis is homed*)
        STATE_READY : USINT := 3; (*constant for a waiting step*)
        STATE_CAM_SELECT : USINT := 4; (*constant for the step in which the cam table is selected*)
        STATE_CAM_START : USINT := 5; (*constant for the step in which the coupling is started*)
        STATE_CAM_STOP : USINT := 6; (*constant for the step in which the coupling is stopped*)
        STATE_ERROR : USINT := 7; (*constant for the error step*)
        STATE_ERROR_AXIS : USINT := 8; (*constant for the axis error step*)
        STATE_ERROR_RESET : USINT := 9; (*constant for the step in which the errors are reset*)
END_VAR
```

### 7. VARIABLE GLOBAL.VAR

**VARIABLE GLOBAL VAR**

```
(*********************************************************************
 * COPYRIGHT -- cnam
 *********************************************************************
 * File: Global.typ
 * Author: Rolf Pérot
 * Created: November 13, 2012
 *********************************************************************
 * Global data types of project ROTATION1
 *********************************************************************)

TYPE
	moteurparameter_typ :	STRUCT
		ScalingInterval : UINT := 100; (*0.1 s*)
		ScalingIntervalMax : UINT := 60000; (*60 s max *)
		ScalingIntervalMin : UINT := 0; (*0 s*)
		ScalingAccuracyDistance : REAL := 1.0;
		ScalingAccuracyFrequency : REAL := 0.5; (*Relative to the master axis (speed of the master implies frequency of the slave axix)*)
		FrequencyMax : REAL := 50.0; (*Relative to the slave axis (see NewProgramCyclic.st) // Master,Frequency Scaling*)
		FrequencyMin : REAL := 0.0; (*Relative to the slave axis (see NewProgramCyclic.st) // Master,Frequency Scaling*)
		Direction : BOOL;
		DecelerationAccuracy : REAL := 20.0;
		AccelerationAccuracy : REAL := 20.0;
		AccelerationMin : REAL := 20.0;
		AccelerationMax : REAL := 360.0;
		DecelerationMin : REAL := 20.0;
		DecelerationMax : REAL := 360.0;
```

```
        DistanceMin : REAL := 0.0;
        DistanceAccuracy : REAL := 196.0;
        DistanceMax : REAL := 80.0;
        PositionAccuracy : REAL := 1.0;
        VelocityMin : REAL := 1.0;
        PositionMax : REAL := 360.0;
        PositionMin : REAL := 1.0;
        VelocityAccuracy : REAL := 1.0;
        VelocityMax : REAL := 360.0;
    END_STRUCT;
    moteurstatus_typ :    STRUCT
        profile : STRING[50];
        TimeInterval : UINT := 500; (*starting value, displayed value, prog starting with this value ( in
ms)*)
        Distancemm : REAL; (*Desired Distance to do for the slave axis in mm*)
        SlaveScaling : REAL := 1.0;
        MasterScaling : REAL := 1.0;
        Frequency : REAL := 1.0; (*Frequency of the slave axis*)
        Distance : REAL := 40.0;
        Direction : USINT := 1;
        Position_in_mm : REAL := 0.0; (*Desired Position to reach*)
        Position : REAL := 0.0;
        Deceleration : REAL := 20.0;
        Acceleration : REAL := 20.0;
        Velocity : REAL := 0.0;
    END_STRUCT;
    moteurcmd_typ :      STRUCT
        profile : BOOL;
        profileBis : BOOL;
```

```
        DecreaseTimeInterval : BOOL;
        IncreaseTimeInterval : BOOL;
        DecreaseFrequency : BOOL; (*refer to the master scaling*)
        IncreaseFrequency : BOOL; (*refer to the master scaling*)
        DecreaseSlaveScaling : BOOL;
        IncreaseSlaveScaling : BOOL;
        DirectionNeg : BOOL;
        DirectionPos : BOOL;
        DecreaseDistance : BOOL;
        IncreaseDeceleration : BOOL;
        DecreaseDeceleration : BOOL;
        IncreaseDistance : BOOL;
        DecreasePosition : BOOL;
        IncreasePosition : BOOL;
        DecreaseAcceleration : BOOL;
        IncreaseAcceleration : BOOL;
        DecreaseVelocity : BOOL;
        IncreaseVelocity : BOOL;
END_STRUCT;
moteur_typ :  STRUCT
        command : moteurcmd_typ;
        status : moteurstatus_typ;
        parameter : moteurparameter_typ;
END_STRUCT;
basiccommand_typ :  STRUCT
        MoveAbsolute : BOOL;
        MoveAdditive : BOOL;
        MoveVelocity : BOOL;
```

```
        Halt : BOOL;
        ErrorAcknowledge : BOOL;
        Position : BOOL;
        Distance : BOOL;
        Stop : BOOL;
        Home : BOOL;
        Power : BOOL;
END_STRUCT;
basicparameter_typ :        STRUCT
        Direction : USINT; (*Used only by Absolute Mvt*)
        Deceleration : REAL; (*Used both by Additive Mvt & Absolute Mvt & Master Axis*)
        HomePosition : USINT := 0; (*Value in unit when the homing is done*)
        Acceleration : REAL; (*Used both by Additive Mvt & Absolute Mvt & Master Axis*)
        Distance : REAL; (*Used only by Additive Mvt*)
        Position : REAL; (*Used by the  Absolute Mvt*)
END_STRUCT;
basicaxisstate_typ :    STRUCT
        Disabled : USINT;
        ErrorStop : USINT;
        StandStill : USINT;
        Homing : USINT;
        Stopping : USINT;
        SynchronizedMotion : USINT;
        ContinuousMotion : USINT;
        DiscreteMotion : USINT;
        Standstill : USINT;
END_STRUCT;
basicstatus_typ :       STRUCT
```

```
        ErrorID : UINT;
        ErrorText : ARRAY[0..3]OF STRING[79];
        DriveStatus : MC_DRIVESTATUS_TYP;
        ActPosition : REAL;
        ActVelocity : REAL;
END_STRUCT;
Basic_typ :     STRUCT
        AxisState : basicaxisstate_typ;
        Status : basicstatus_typ;
        Parameter : basicparameter_typ;
        Command : basiccommand_typ;
END_STRUCT;
camstatus_typ :         STRUCT
        ActTorque : REAL;
        DriveStatus : USINT;
        ActVelocity : REAL;
        ActPosition : REAL;
        ErrorText : ARRAY[0..3]OF STRING[79];
        ErrorID : UINT;
END_STRUCT;
camparameter_typ :  STRUCT
        StartMode : USINT;
        SlaveScaling : REAL;
        MasterScaling : REAL;
        SlaveOffset : REAL;
        MasterOffset : REAL;
        HomePosition : USINT := 0;
        Deceleration : REAL;
```

```
        Direction : BOOL;
        Acceleration : REAL;
        Velocity : REAL;
        Distance : REAL;
        Position : REAL;
END_STRUCT;
camaxisstate_typ :    STRUCT
        ContinuousMotion : USINT;
        StandStill : USINT;
        SynchronizedMotion : USINT;
        ErrorStop : USINT;
        DiscreteMotion : USINT;
        Homing : USINT;
        Stopping : USINT;
        Disabled : USINT;
END_STRUCT;
AxisStateRolf_Typ :    STRUCT
        Disabled : USINT;
END_STRUCT;
ParameterRolf_typ :   STRUCT
END_STRUCT;
CommandRolf_typ :   STRUCT
        ErrorAcknowledge : BOOL;
        PowerMeter : BOOL; (*switch ON the bloc modules for the desired measures*)
        Alim : BOOL; (*switch ON the Alim filter*)
END_STRUCT;
StatusRolf_typ :        STRUCT
        InversePowerMin : REAL;
```

```
        InversePowerMax : REAL;
        AverageReactivePower : REAL;
        EnergyBalance : REAL;
        MaximumActivePower : REAL;
        ConsumedEnergy : REAL;
        MinimalActivePower : REAL;
        AverageActivePower : REAL;
        RegeneratedEnergy : REAL;
        ErrorID : UINT;
        ErrorText : ARRAY[0..3]OF STRING[79];
END_STRUCT;
camcommand_typ :   STRUCT
        sens : USINT;
        DisengageSlave : BOOL;
        ErrorAcknowledge : BOOL;
        Halt : BOOL;
        StartSlave : BOOL;
        MoveVelocity : BOOL;
        MoveAdditive : BOOL; (*deplacement dune distance*)
        MoveAbsolute : BOOL; (*deplacement a une position *)
        Position : BOOL;
        Distance : BOOL;
        Home : BOOL;
        Power : BOOL;
        Stop : BOOL;
END_STRUCT;
ALIMROLF_typ :        STRUCT
        Status : StatusRolf_typ;
```

```
        Parameter : ParameterRolf_typ;
        AxisState : AxisStateRolf_Typ;
        Command : CommandRolf_typ;
    END_STRUCT;
    meca_typ :    STRUCT
        MechanicalPower : REAL;
    END_STRUCT;
    Cam_typ :     STRUCT
        meca : meca_typ;
        AxisState : camaxisstate_typ;
        Command : camcommand_typ;
        Parameter : camparameter_typ;
        Status : camstatus_typ;
    END_STRUCT;
END_TYPE
```

### 8. VARIABLE GLOBAL TYP

```
(*********************************************************************
 * COPYRIGHT -- cnam
 *********************************************************************
 * File: Global.typ
 * Author: Rolf Pérot
 * Created: November 13, 2012
 *********************************************************************
 * Global data types of project ROTATION1
 *********************************************************************)

TYPE
```

```
moteurparameter_typ :        STRUCT
        ScalingInterval : UINT := 100; (*0.1 s*)
        ScalingIntervalMax : UINT := 60000; (*60 s max *)
        ScalingIntervalMin : UINT := 0; (*0 s*)
        ScalingAccuracyDistance : REAL := 1.0;
        ScalingAccuracyFrequency : REAL := 0.5; (*Relative to the master axis (speed of the master implies frequency of the slave axix)*)
        FrequencyMax : REAL := 50.0; (*Relative to the slave axis (see NewProgramCyclic.st) // Master,Frequency Scaling*)
        FrequencyMin : REAL := 0.0; (*Relative to the slave axis (see NewProgramCyclic.st) // Master,Frequency Scaling*)
        Direction : BOOL;
        DecelerationAccuracy : REAL := 20.0;
        AccelerationAccuracy : REAL := 20.0;
        AccelerationMin : REAL := 20.0;
        AccelerationMax : REAL := 360.0;
        DecelerationMin : REAL := 20.0;
        DecelerationMax : REAL := 360.0;
        DistanceMin : REAL := 0.0;
        DistanceAccuracy : REAL := 196.0;
        DistanceMax : REAL := 80.0;
        PositionAccuracy : REAL := 1.0;
        VelocityMin : REAL := 1.0;
        PositionMax : REAL := 360.0;
        PositionMin : REAL := 1.0;
        VelocityAccuracy : REAL := 1.0;
        VelocityMax : REAL := 360.0;
END_STRUCT;
moteurstatus_typ :    STRUCT
```

```
        profile : STRING[50];
        TimeInterval : UINT := 500; (*starting value, displayed value, prog starting with this value ( in
ms)*)
        Distancemm : REAL; (*Desired Distance to do for the slave axis in mm*)
        SlaveScaling : REAL := 1.0;
        MasterScaling : REAL := 1.0;
        Frequency : REAL := 1.0; (*Frequency of the slave axis*)
        Distance : REAL := 40.0;
        Direction : USINT := 1;
        Position_in_mm : REAL := 0.0; (*Desired Position to reach*)
        Position : REAL := 0.0;
        Deceleration : REAL := 20.0;
        Acceleration : REAL := 20.0;
        Velocity : REAL := 0.0;
    END_STRUCT;
    moteurcmd_typ :     STRUCT
        profile : BOOL;
        profileBis : BOOL;
        DecreaseTimeInterval : BOOL;
        IncreaseTimeInterval : BOOL;
        DecreaseFrequency : BOOL; (*refer to the master scaling*)
        IncreaseFrequency : BOOL; (*refer to the master scaling*)
        DecreaseSlaveScaling : BOOL;
        IncreaseSlaveScaling : BOOL;
        DirectionNeg : BOOL;
        DirectionPos : BOOL;
        DecreaseDistance : BOOL;
        IncreaseDeceleration : BOOL;
        DecreaseDeceleration : BOOL;
```

```
        IncreaseDistance : BOOL;
        DecreasePosition : BOOL;
        IncreasePosition : BOOL;
        DecreaseAcceleration : BOOL;
        IncreaseAcceleration : BOOL;
        DecreaseVelocity : BOOL;
        IncreaseVelocity : BOOL;
END_STRUCT;
moteur_typ :  STRUCT
        command : moteurcmd_typ;
        status : moteurstatus_typ;
        parameter : moteurparameter_typ;
END_STRUCT;
basiccommand_typ :  STRUCT
        MoveAbsolute : BOOL;
        MoveAdditive : BOOL;
        MoveVelocity : BOOL;
        Halt : BOOL;
        ErrorAcknowledge : BOOL;
        Position : BOOL;
        Distance : BOOL;
        Stop : BOOL;
        Home : BOOL;
        Power : BOOL;
END_STRUCT;
basicparameter_typ :          STRUCT
        Direction : USINT; (*Used only by Absolute Mvt*)
        Deceleration : REAL; (*Used both by Additive Mvt & Absolute Mvt & Master Axis*)
```

```
        HomePosition : USINT := 0; (*Value in unit when the homing is done*)
        Acceleration : REAL; (*Used both by Additive Mvt & Absolute Mvt & Master Axis*)
        Distance : REAL; (*Used only by Additive Mvt*)
        Position : REAL; (*Used by the  Absolute Mvt*)
END_STRUCT;
basicaxisstate_typ :    STRUCT
        Disabled : USINT;
        ErrorStop : USINT;
        StandStill : USINT;
        Homing : USINT;
        Stopping : USINT;
        SynchronizedMotion : USINT;
        ContinuousMotion : USINT;
        DiscreteMotion : USINT;
        Standstill : USINT;
END_STRUCT;
basicstatus_typ :       STRUCT
        ErrorID : UINT;
        ErrorText : ARRAY[0..3]OF STRING[79];
        DriveStatus : MC_DRIVESTATUS_TYP;
        ActPosition : REAL;
        ActVelocity : REAL;
END_STRUCT;
Basic_typ :     STRUCT
        AxisState : basicaxisstate_typ;
        Status : basicstatus_typ;
        Parameter : basicparameter_typ;
        Command : basiccommand_typ;
```

```
END_STRUCT;
camstatus_typ :         STRUCT
        ActTorque : REAL;
        DriveStatus : USINT;
        ActVelocity : REAL;
        ActPosition : REAL;
        ErrorText : ARRAY[0..3]OF STRING[79];
        ErrorID : UINT;
END_STRUCT;
camparameter_typ :  STRUCT
        StartMode : USINT;
        SlaveScaling : REAL;
        MasterScaling : REAL;
        SlaveOffset : REAL;
        MasterOffset : REAL;
        HomePosition : USINT := 0;
        Deceleration : REAL;
        Direction : BOOL;
        Acceleration : REAL;
        Velocity : REAL;
        Distance : REAL;
        Position : REAL;
END_STRUCT;
camaxisstate_typ :     STRUCT
        ContinuousMotion : USINT;
        StandStill : USINT;
        SynchronizedMotion : USINT;
        ErrorStop : USINT;
```

```
        DiscreteMotion : USINT;
        Homing : USINT;
        Stopping : USINT;
        Disabled : USINT;
END_STRUCT;
AxisStateRolf_Typ :    STRUCT
        Disabled : USINT;
END_STRUCT;
ParameterRolf_typ :   STRUCT
END_STRUCT;
CommandRolf_typ :   STRUCT
        ErrorAcknowledge : BOOL;
        PowerMeter : BOOL; (*switch ON the bloc modules for the desired measures*)
        Alim : BOOL; (*switch ON the Alim filter*)
END_STRUCT;
StatusRolf_typ :        STRUCT
        InversePowerMin : REAL;
        InversePowerMax : REAL;
        AverageReactivePower : REAL;
        EnergyBalance : REAL;
        MaximumActivePower : REAL;
        ConsumedEnergy : REAL;
        MinimalActivePower : REAL;
        AverageActivePower : REAL;
        RegeneratedEnergy : REAL;
        ErrorID : UINT;
        ErrorText : ARRAY[0..3]OF STRING[79];
END_STRUCT;
```

```
camcommand_typ :   STRUCT
        sens : USINT;
        DisengageSlave : BOOL;
        ErrorAcknowledge : BOOL;
        Halt : BOOL;
        StartSlave : BOOL;
        MoveVelocity : BOOL;
        MoveAdditive : BOOL; (*deplacement dune distance*)
        MoveAbsolute : BOOL; (*deplacement a une position *)
        Position : BOOL;
        Distance : BOOL;
        Home : BOOL;
        Power : BOOL;
        Stop : BOOL;
END_STRUCT;
ALIMROLF_typ :        STRUCT
        Status : StatusRolf_typ;
        Parameter : ParameterRolf_typ;
        AxisState : AxisStateRolf_Typ;
        Command : CommandRolf_typ;
END_STRUCT;
meca_typ :     STRUCT
        MechanicalPower : REAL;
END_STRUCT;
Cam_typ :      STRUCT
        meca : meca_typ;
        AxisState : camaxisstate_typ;
        Command : camcommand_typ;
```

Parameter : camparameter_typ;

Status : camstatus_typ;

END_STRUCT;

END_TYPE

## C. B CODEUR GC-MK5

Vérification des signaux de sorties à l'oscilloscope :

**Output Signals**

**Fig A** : Allure théorique des courbes de sorties du codeur magnétique suivant la doc technique

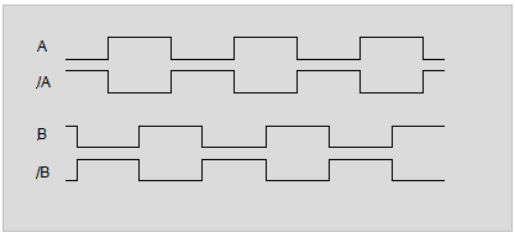
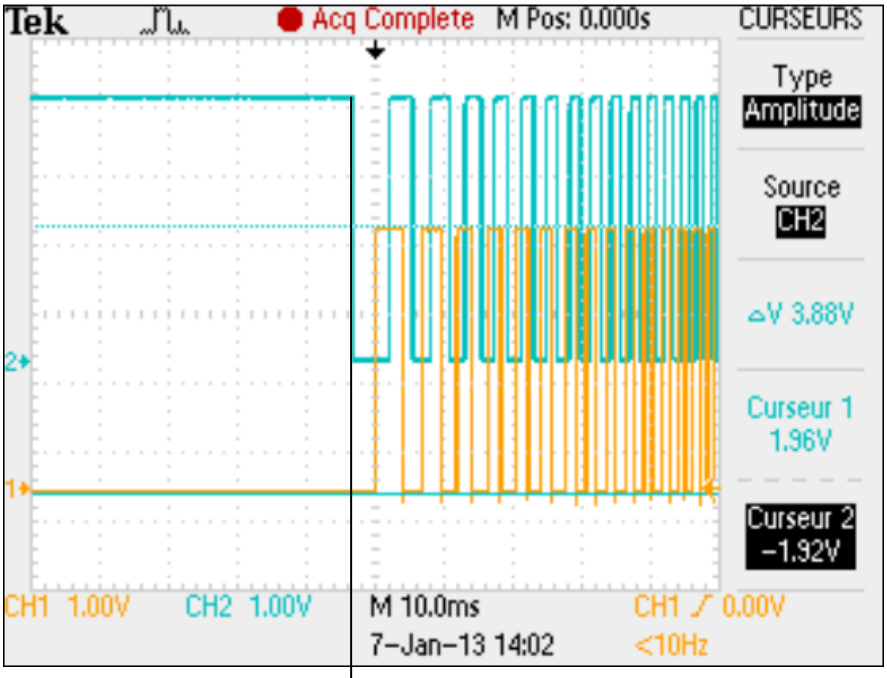



**Fig B** : A et /B mesuré à l'oscilloscope




0 t

Arrêt du mover

Déplacement manuel du mover avec une accélération très légère…

On constate que le signal est bien déphasé d'un quart de période en avance par rapport à A.

Crête à crête : 3.88 V

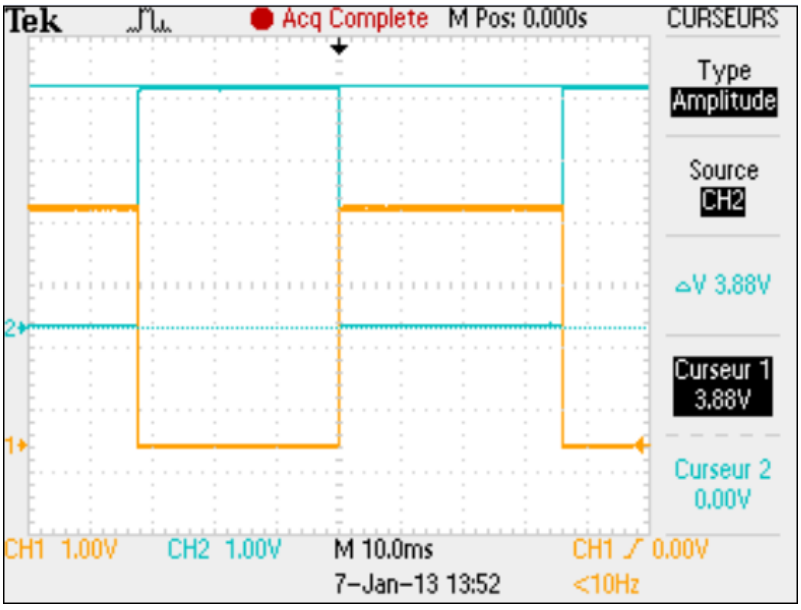


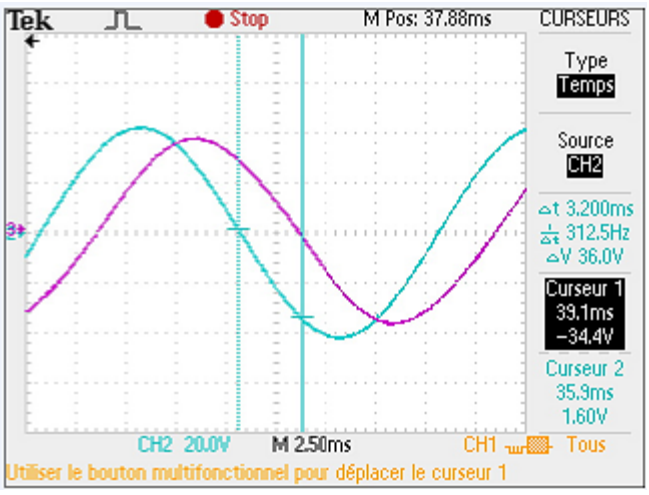

Figure XIX-4: signaux A et /A

## D. Mesure des paramètres du stator de la machine

Essai à 50 Hz

Calcul de l'impédance de la phase blanche

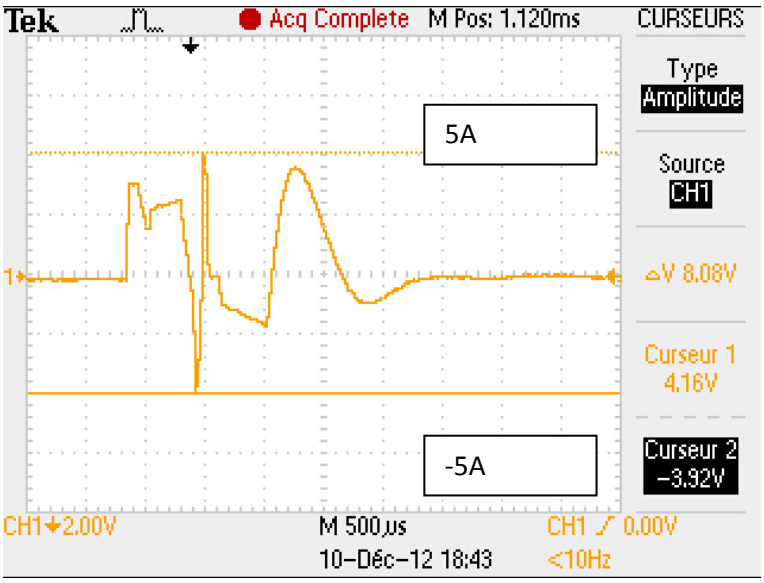
Mesure du déphasage( Ieff, Ueff)



Déduction de R :

Déduction de ls

## E. Mesure des courants de terre lors de la mise sous tension de l'installation



# XX. Liste des figures