7e année

La Revue 3E.I

"London Eye", la plus haute roue observatoire du monde.

Son automatisme, conçu par la société Semer, s'appuie sur une architecture redondante mettant en œuvre 2 systèmes de supervision Monitor Pro, 66 automates programmables Premium et 64 variateurs de vitesse Altivar 58

Les automatismes industriels

Photo : Schneider Electric

Publication trimestrielle du Cercle Thématique 13.01 de la SEE

ENSEIGNER L'ELECTROTECHNIQUE ET L'ÉLECTRONIQUE INDUSTRIELLE

Société de l'Electricité, de l'Electronique et des Technologies de l'Information et de la Communication

N° 23 - Décembre 2000

JOURNÉES ÉLECTROTECHNIQUE

"Arc Electrique, la foudre en bouteille au service de l'électrotechnique"

15 et 16 Mars 2001
Université Paul Sabatier - Toulouse

Organisateurs : Département EEA (UPS), CPAT et LGET (UPS)
Georges Zissis e-mail zissis@cpat.ups-tlse.fr
Manifestation commune Club EEA et Club Electrotechnologies et Enseignement (CEE)

SOCIETE de l'ELECTRICITE, de l'ELECTRONIQUE et des TECHNOLOGIES de l'INFORMATION et de la COMMUNICATION.
48, rue de la Procession 75 724 PARIS CEDEX 15
Tel : 01 44 49 60 00 fax : 01 44 49 60 49

SEE, association reconnue d'utilité publique par le décret du 7 décembre 1886
Siret 785 393 232 00026, APE 731 Z, n° d'identification FR 44 785 393 232

La REVUE 3EI *publication trimestrielle du* **Cercle Thématique 13-01 de la SEE :**
Enseignement de l'Electrotechnique et de l'Electronique Industrielle.

Edition SEE, *48 Rue de la Procession 75 724 PARIS CEDEX 15*
Directeur de la publication
François AILLERET
Président de la SEE

Rédacteur en Chef
François BOUCHER

Equipe de Rédaction
Animateurs :
Marie-Michèle LE BIHAN
Gérard DELAVIER

Gilles FELD
Brigitte GRELAUD
Jean-Philippe ILARY
Pascal LOOS
Philippe NEVEU
Michel THION

Site WEB *(lesite3EI.com)*
Responsable : *Philippe LE BRUN*

Fabrice BAUDOIN
Jean-Pierre TAREL
Jean-Philippe ILARY

Comité de lecture :
Jean-Pierre CARON
Jean FAUCHER (ENSEEIHT)
François FOREST (Univ. de MontpelierII)
Michel LAVABRE (ENS Cachan)
Michel MEUNIER (SUPELEC)
Bernard MULTON (ENS Cachan ,Antenne de Bretagne)
Jean-Marie PETER (SEE)

Abonnement annuel *(4 numéros)*
Année scolaire 2000 / 2001
Individuel *: 195 F TTC (France et CEE)*
260 F TTC pays hors CEE

Collectivité *:250 F TTC (France et CEE)*
320 F TTC pays hors CEE

Réalisation et impression *(1000 ex)*
Repro-Systèmes
23, rue de Verdun 77 181 Le Pin

Routage et Expédition
Départ Presse
ZI les Richardets
93 966 Noisy le Grand

Dépôt Légal : décembre 2000
Commission Paritaire 1202 G 78028
ISSN 1252-770X

Sommaire du n°23
Thème : les automatismes industriels

Depuis six ans déjà, l'éditorial de décembre de ***La Revue 3EI*** *est consacré à vous informer sur le fonctionnement de la revue et sur l'action du cercle thématique 13-01 de la SEE ; nous ne modifierons pas cette habitude.*

Précisons d'abord que le nombre de nos abonnés continue sa lente progression ; vous étiez ***830*** *pour cette sixième année. Ceci est un encouragement pour toute l'équipe de rédaction qui ne ménage pas ses efforts pour réaliser une publication répondant à l'attente de ses lecteurs. Il faut néanmoins que ce nombre de lecteurs, souscrivant un abonnement, continue de croître, c'est le seul moyen d'assurer l'équilibre financier de la revue et par voie de conséquence son existence.*

*Cette sixième année fut pour nous une année transitoire : équipe de rédaction renouvelée, présentation de la Revue 3EI modifiée et lancement du site web (****lesite3EI.com****).*

L'équipe de rédaction *est désormais plus nombreuse ; les collègues l'ayant rejointe ont de suite fait preuve de leur détermination, de leur dynamisme et de leur efficacité. Cette équipe accueillera avec plaisir tous ceux qui souhaitent participer à ce travail, prenant mais au combien passionnant. Nous manquons d'informations sur les évènements qui se déroulent en province et ils sont nombreux. (conférences, journées d'étude, mise en service de matériels nouveaux ...) des correspondants en régions pourraient nous faire part des divers évènements ; ce fut le cas pour les journées* ***GEVIQ'2000*** *à Marseille.*

La présentation de La Revue 3EI *a été modifiée entraînant une contrainte pour les auteurs d'articles, mais en revanche cette transformation a permis d'aérer la présentation de notre publication. Nous devons continuer de travailler sur la forme en introduisant bientôt, nous l'espérons, quelques pages en couleur.*

*Désormais, le format pour l'envoi d'un article est disponible sur le Web (****lesite3EI.com****).*

Le site 3EI*, constitue un moyen de communication et d'échanges d'informations entre les enseignants et aussi un excellent complément à la Revue (c'est le cas encore pour ce numéro 23 où des exemples de réalisations d'automatismes industriels y seront présentés en complément aux articles édités).*

Au cours de cette année 2001 il n'y aura pas de ***Journées 3EI****. Nous en avons expliqué la raison, dans le numéro précédent ; c'est avant tout un problème de financement . Cela ne signifie nullement l'arrêt définitif de ce type de manifestation ; elles renaîtront peut-être en 2003 si nous avons des soutiens financiers suffisants pour offrir des journées d'études à des prix suffisamment raisonnables pour être accessibles au plus grand nombre de nos collègues.*

La Revue 3EI, débute sa septième année de parution, par ce numéro 23 dont le thème est consacré aux automatismes industriels. L'équipe de rédaction espère que ce numéro comme les suivants continueront de vous apporter des informations utiles dans la réalisation du travail d'enseignant et compte aussi sur votre aide pour continuer la promotion de notre travail en faisant connaître autour de vous.

Bien amicalement, l'Equipe de rédaction de la Revue 3EI.

TRAITEMENT DU SIGNAL Série Aide-mémoire.

Francis COTTET, Editions DUNOD (237 pages, 148 F TTC)

Cet aide mémoire s'adresse principalement aux étudiants d'IUT, de 1er et 2e cycles, ainsi qu'aux élèves ingénieurs. Il intéressera également les ingénieurs et techniciens en exercice.
L'ouvrage, qui offre une approche pragmatique, est composé de deux grandes parties : le traitement des signaux analogiques et le traitement des signaux numériques. L'aspect "théorie du signal" est limité au strict nécessaire pour la compréhension des modèles utilisés, les bases mathématiques sont rappelées en annexe.

HISTOIRE DE L'ELECTRICITE

LA FEE ET LA SERVANTE

La société française face à l'électricité XIXème - XXème siècle ;
Alain BELTRAN et Patrice CARRE, Editions BELIN 1991 (348 pages, 170 FTTC).

Le monde moderne est né de l'électricité. Fée à l'orée du siècle, servante docile à la veille du second conflit mondial, l'électricité a tour à tour émerveillé, conquis ou inquiété par ses usages multiples. A l'aide de documents peu connus et à l'écoute des témoignages les plus divers, les auteurs ont su retrouver les sensibilités et les réactions des consommateurs. Au delà de son intérêt historique, cette réflexion sur l'impact social et culturel de l'innovation déclenchée par la "révolution électrique" nous invite à méditer sur les mutations à venir. L'exemple de l'électricité est riche de leçons : toute société, sous peine de dépérir, doit intégrer la modernisation technique.
Alain Beltran, agrégé d'histoire et chargé de cours à l'université Paris Sorbonne, est également l'auteur de la "**FEE ELECTRICITE**", Découvertes Gallimard 1991.

LOUIS FIGUIER, LES MERVEILLES DE L'ELECTRICITE.

Textes choisis et présentés par Fabienne CARDOT; (522 pages, très nombreuses gravures).
Ouvrage édité (1985) par l'Association pour l'Histoire de l'Electricité en France -
48, rue de la Procession, 75015 PARIS (120F + 29 F de frais d'envoi TTC).

En patronnant la réédition de ces textes aujourd'hui quasi introuvables, en les faisant précéder d'une présentation critique qui comble une lacune de la bibliographie de l'histoire des techniques, l'A.H.E.F. a voulu rendre hommage à un grand vulgarisateur Louis Figuier (1819-1894) et mettre à la disposition de tous une source historique qui - ce n'est pas si fréquent- n'a rien d'austère. Explications techniques, biographies, récits anecdotiques, accompagnés de nombreuses gravures, font revivre l'esprit d'invention et l'enthousiasme inaugural qui amenèrent les hommes des années 1870-1900 à maîtriser et utiliser, pour la première fois, l'électricité.

ARCHITECTURES D'AUTOMATISMES INDUSTRIELS : évolutions et grandes tendances

Alain LETOURMY
SCHNEIDER ELECTRIC SA
92 566 RUEIL MALAISON CEDEX

Ces vingt dernières années, les architectures d'automatismes ont très fortement évolué. Ce phénomène s'amplifie avec l'arrivée des nouvelles technologies de l'information et de la communication (NITC).

Ces changements successifs sont dus, d'une part, à l'évolution des besoins des utilisateurs et d'autre part, aux développements des technologies . Par utilisateurs, il faut entendre tous les intervenants : exploitants, intégrateurs, metteurs en œuvre, équipes de maintenance, etc.

Architectures d'automatismes, des évolutions progressives.

Depuis que les automatismes sont réalisés sur la base d'unités de traitement (machines informatiques ou automates programmables), les architectures ont fortement évolué et sont passées par différents stades pour arriver aux architectures d'aujourd'hui, basées sur l'adoption des grands standards de communication et sur l'arrivée des NTIC.

Les automatismes centralisés

Jusque dans les années 80, les automatismes s'appuyant sur des automates programmables industriels (API), ne traitaient que des fonctions séquentielles.
En simplifiant, les API

- Géraient des demandes d'exécution d'opérateur, l'état de l'automatisme (image des Entrées)
- Elaboraient des demandes d'exécution d'actions (positionnement des sorties)
- Contrôlaient le bon déroulement des actions (gestion des entrées).

A la demande des utilisateurs, les API ont du rapidement gérer de nombreuses fonctions complémentaires comme des fonctions métier, des fonctions de diagnostic (système mais aussi application), etc .

Les automatismes centralisés géraient tout un ensemb le de fonctions d'automatismes qui n'avaient pas forcement d'interactions entre elles. Lorsqu'il y avait déjà un automate dans l'usine, les automaticiens qui devaient automatiser une fonction supplémentaire, se posaient simplement la question : l'automate ou le système d'automatismes en place peut il gérer les E/S supplémentaires et quelle est la capacité de mémoire disponible ?
Bien souvent, l'automatisation supplémentaire était réalisée avec cet automate existant, même si cette nouvelle automatisation n'avait aucun rapport avec l'automatisme résident.

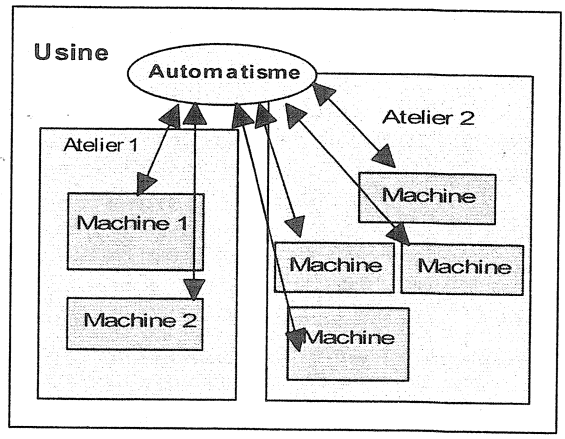


Figure 1 :Les automatismes centralisés

Ces automatismes centralisés amenaient des nombreuses contraintes :

-aucune autonomie des différents sous-ensembles;
-des mises en service et une maintenance lourdes et difficiles à effectuer du fait de la quantité d'E/S gérées;
-arrêt de l'ensemble des fonctions gérées par l'API, en cas de défaut système de cet API ou d'arrêt pour la maintenance du moindre élément de l'outil de production.

Les automatismes décentralisés

Du fait des contraintes imposées par les systèmes centralisés, les utilisateurs ont très rapidement demandé de segmenter les automatismes.

Les premières segmentations ont été faites en découpant l'automatisme en entités fonctionnelles. Cette segmentation permet de simplifier les automatismes en réduisant le nombre d'E/S gérées et présente donc les avantages de faciliter la mise en service et la maintenance .

Adoptée par la plupart des utilisateurs dès les années 85/87, cette segmentation a généré le besoin d'une nouvelle fonction transverse d'automatisme. Cette dernière, **la fonction de communication,** est devenue la clef de voûte de la conception des architectures d'automatisme .

Les constructeurs d'API ont donc créé des offres de réseaux locaux industriels (RLI) afin d'assurer une communication efficace entre les différents API .

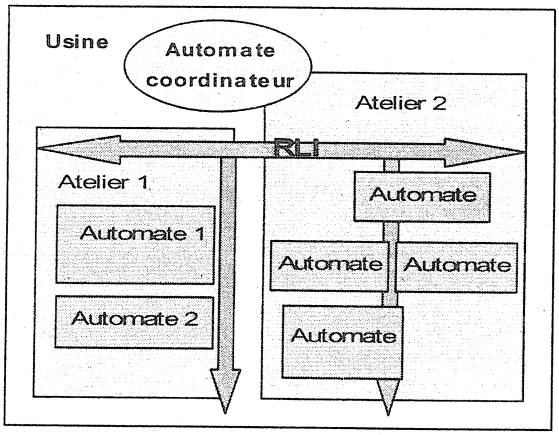


Figure 2 : les automatismes décentralisés

La périphérie d'automatisme distribuée

A la demande des utilisateurs finaux, pour faire baisser les coûts de câblage notamment, il a été nécessaire de prendre en compte la topologie des automatismes.

En complément à cette découpe fonctionnelle, les constructeurs d'API ont répondu en offrant la possibilité de déporter des racks automate. Ceux-ci peuvent recevoir des entrées/sorties et des coupleurs métier.

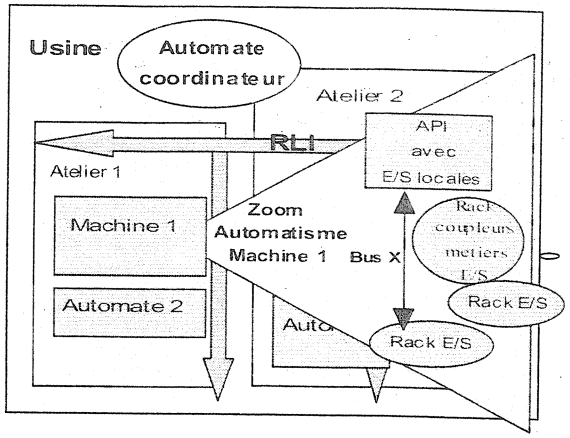


Figure 3 : la périphérie d'automatisme distribuée.

Cette offre est particulièrement adaptée dans le cas d'automatisation de machines et permet de placer les coupleurs métier et les groupes d'entrées/sorties au plus près des capteurs et actionneurs.

La décentralisation des entrées/sorties et de la périphérie d'automatisme

Sur des sites plus étendus, il est souvent nécessaire de gérer un nombre de points diffus importants. Pour réaliser encore des gains de câblage, il était nécessaire de proposer un système économique pour relier tous ces points.

La réponse des constructeurs de produits d'automatismes est arrivée avec les réseaux et bus de terrain. Ceux-ci ont permis de gérer dans un premier temps des E/S décentralisées puis de la périphérie d'automatisme. Ces réseaux de terrain permettent de réaliser des gains de câblage importants, mais surtout ils permettent de rendre accessibles des services (diagnostic, programmation,...) sur tout le site.

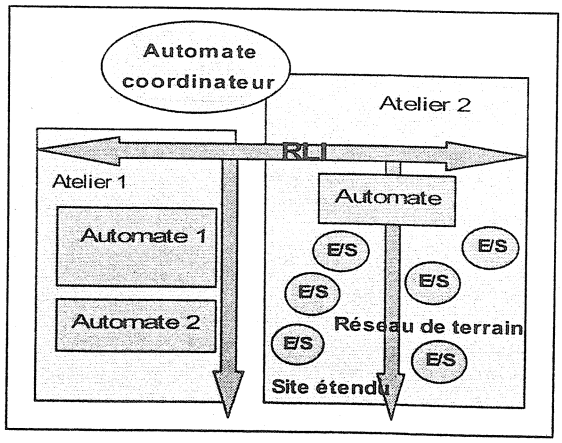


Figure 4 : la décentralisation des entrées /sorties

Le rôle de l'informatique dans les automatismes.

De nombreux automatismes centralisés ont été réalisés avec des machines informatiques jusque dans les années 85/87. Par la suite la plupart des automatismes industriels ont été conçus avec des API. Dans une certaine mesure, l'informatique a été prise en compte par les automaticiens au travers des offres de supervision des constructeurs de produits d'automatismes.

Automatisme et Informatique, deux mondes qui se sont longtemps ignorés

Le besoin d'assurer une communication entre les deux mondes de l'informatique et des automatismes, qui se sont longtemps ignorés, est devenu indispensable avec la nécessité d'augmenter la productivité des usines de fabrication. Pour les automaticiens, assurer cette communication n'a pas toujours été chose facile. Pendant longtemps, les réseaux informatiques étaient, standardisés mais gardaient quelques caractéristiques propriétaires. Les réseaux locaux industriels d'automatismes, quant à eux, étaient tous propriétaires.

La communication entre ces deux mondes a d'abord été assurée par des liaisons série (RS), puis par des produits issus de partenariats entre les constructeurs d'automates programmables et les grands de l'informatique tels que IBM, HP, DEC. Ces derniers ont proposé, dans leur catalogue, une offre de coupleurs RLI comme Ethway ou Uni-Telway.

Avec la standardisation progressive d'Ethernet, dans les deux mondes, et une tentative de standardisation d'un protocole MMS (*manufacturing message spécification*) comportant des services communs aux deux mondes, la frontière entre ces deux mondes devait tomber !

Quelques applications basées sur « cet espéranto automatique/informatique » ont été réalisées, mais très vite, ce standard « MMS » n'a plus été utilisé.

Le CIM créé une segmentation des réseaux et bus.

La demande d'une automatisation beaucoup plus intégrée a amené les grandes théories du CIM (*Computer Integrated Manufacturing*).

Le CIM n'a pas fait tomber la frontière entre automatisme et informatique, mais a segmenté l'automatisme en niveaux: capteurs/actionneurs, automatismes, supervision, informatique.

Les constructeurs d'automates programmables ont créé des réseaux et des bus adaptés à ces différents niveaux. Ainsi à chaque niveau, correspond un bus ou un réseau :

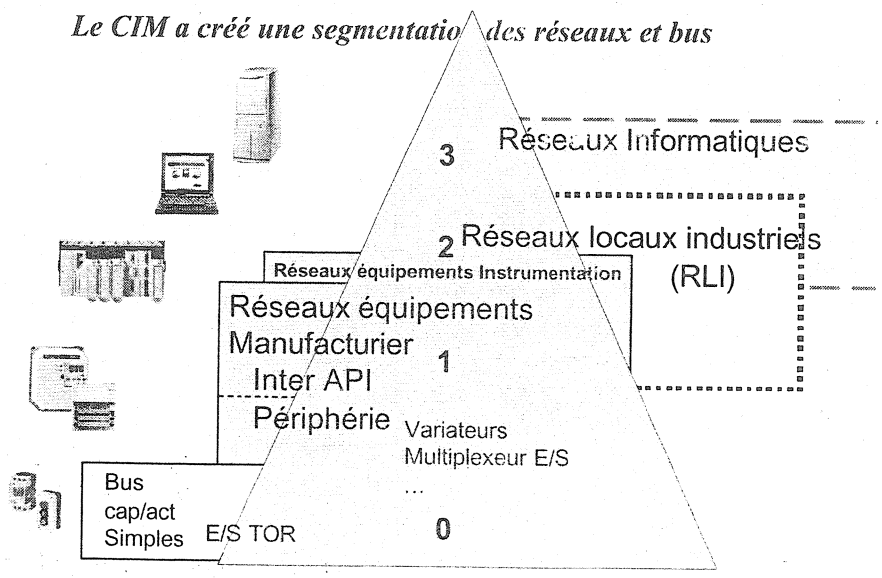
les « *sensor bus* », bus capteurs et actionneurs unitaires simples;

les « *device bus* », bus et réseaux pour la périphérie d'automatisme : variateurs, robots, axes...

les «*field bus* », réseaux de communication entre unités de traitement : automates programmables, superviseur, commandes numériques...

les réseaux locaux industriels qui établissent la communication entre l'automatisme et le monde informatique.

Le CIM a créé une segmentation des réseaux et bus



Evolution de la communication, les grandes tendances.

Avec l'adoption progressive de standards communs par les deux mondes, la frontière entre automatique et informatique tombe enfin.

La communication entre ces deux mondes converge grâce à l'adoption par Schneider Electric de protocoles standards mondiaux Ethernet et TCP-IP, mais aussi grâce à la prise en compte des standards d'ouverture logiciels comme OPC et des techniques Web. Ces nouvelles technologies, associées aux Extranet, Intranet et Internet, permettent de donner un accès aux données de l'automatisme, en tout lieu, à toute personne autorisée.

Cette convergence est renforcée par l'arrivée de nombreux constituants d'automatismes basés sur cette technologie.

Le concept « Transparent Factory » s'appuie sur toutes ces nouvelles technologies de l'information et de la communication (NTIC).

Transparent Factory et le CIM

Nous devons nous interroger sur le devenir de la structuration du CIM. Compte tenu de la tendance à inclure, de plus en plus de traitements au niveau des capteurs et actionneurs, ceux-ci deviennent de véritables unités de traitement autonomes et doivent donc se connecter sur Ethernet TCP-IP afin de rendre accessibles les informations qu'ils gèrent.

Ethernet devient ainsi un support répandu et efficace en matière de souplesse et de flexibilité pour les liaisons entre les automates programmables, les terminaux de dialogue homme/machine, les entrées/sorties, etc. De plus, cette technologie, véritablement ouverte, se connecte à tout environnement d'automatisme sur des systèmes MES (*Manufacturing Execution System*) et ERP (*Enterprise Resource Planning*).

Elle utilise des outils tels que les logiciels de navigation du Web pour les diagnostics, le contrôle local et à distance et toutes une série de services, disponibles grâce à Ethernet TCP-IP et à la croissance accélérée des technologies de World Wide Web .

Transparent Factory est la structure qui permet de mettre à disposition la bonne information : au bon endroit, au bon moment, à toute personne autorisée.

L'impact des NTIC sur le modèle CIM

La dernière décennie a vu l'apparition des nouveaux langages de programmation et mécanismes d'échanges de données dans le monde informatique. L'évènement majeur a été l'implémentation des protocoles TCP et IP dans les architectures et par extension le développement fulgurant d'Internet et son accès dans l'entreprise.

La problématique de gestion des flux s'est vue considérablement réduite par l'utilisation d'IP.

Depuis 10 ans, Ethernet TCP/IP s'est développé sur les architectures d'automatisme et est devenu un réseau fédérateur à partir du niveau 2 du CIM. La généralisation de ce réseau a permis une prise en compte simple de nouveaux outils de conduite et d'exploitation des données (MES / ERP). Parallèlement à l'apparition du MES, les postes de supervision ont tendance à se repositionner au niveau de la conduite.

Aujourd'hui, nous pouvons envisager la descente d'Ethernet TCP/IP sur le bas des architectures d'automatisme (au niveau 1 dans un premier temps) afin de répondre aux nouveaux besoins fonctionnels des composants d'automatisme. Au-delà du médium Ethernet, ce sont surtout les protocoles TCP et IP qui sont visés.

Dans la terminologie NTIC, il ne faut pas comprendre seulement l'utilisation d'Ethernet TCP/IP mais également les technologies et mécanismes éprouvés et usités dans le monde informatique tels que les langages JAVA, CORBA, HTML, XML mais également de nouvelles technologies issues du monde des télécommunications telles que WAP, Bluetooth, WML ...

Ainsi, les NTIC permettent :

-de repenser l'approche dans la conception des architectures d'automatisme;

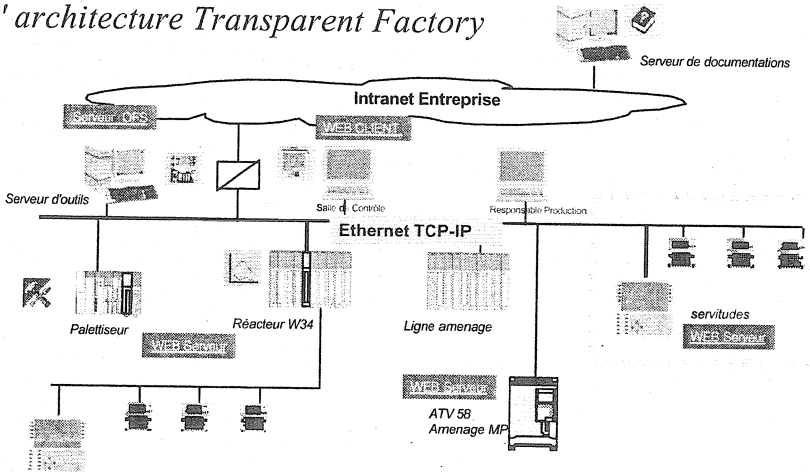
-d'augmenter les capacités et les performances sur les réseaux de communication par le développement de nouveaux modèles de communication;

-de développer des services associés aux métiers des différents acteurs du procédé.

Globalement, les NTIC ont tendance à aplanir la pyramide du CIM, c'est à dire à fusionner certains niveaux car il n'est plus utile de décrire les niveaux sous une approche quantitative mais fonctionnelle.

ETHERNET / INTERNET

L' architecture Transparent Factory



LES AUTOMATES PROGRAMMABLES INDUSTRIELS

GUIGNABEL Alain
SCHNEIDER-ELECTRIC SA
5, rue Nadar – F-92566 RUEIL-MALMAISON CEDEX

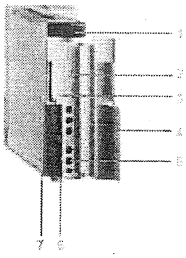
Résumé : En 30 ans, les automates programmables industriels (ou API) ont pris une place incontournable dans tous les domaines de l'automatisation. En plus de leurs fonctions de base, à savoir le traitement d'automatismes issus des relais électromécaniques, ils traitent des fonctions de plus en plus nombreuses et de plus en plus complexes. Ils s'intègrent naturellement dans des architectures d'automatismes performantes, assurant une transparence totale de la communication aux utilisateurs. Aujourd'hui ils s'ouvrent sans problème sur INTERNET.

Depuis 1969, date où MODICON lançait le premier "Programmable Logic Controler", les utilisateurs et les constructeurs ont évolué pour développer de nouvelles unités de traitement, formés de plusieurs processeurs coopérant pour réaliser différentes fonctions d'automatisme : fonctions logiques combinatoire et séquentielle, fonctions analogiques (acquisition, régulation), fonction commande de mouvement, fonction pesage, ...

Les automates programmables industriels (ou API), sont les systèmes les plus utilisés pour résoudre tous les problèmes d'automatismes. les contraintes d'environnement sont des critères essentiels de choix d'un API (les essais d'environnement sont décrits dans la Norme IEC 1131-2).

I. LE MATERIEL

L'architecture matérielle d'un API, qui décrit comment sont implantées les différentes fonctions, comprend 2 modèles : l'automate compact (un produit intègre en un seul bloc toutes les fonctions) et l'automate modulaire (l'API se compose suivant le besoin en implantant des cartes sur un socle). Quel qu'il soit le modèle, l'API comprend toujours les éléments suivants :



L'alimentation qui fournit les tensions nécessaires au fonctionnement interne de l'API, à partir soit du secteur, soit d'une source continue :

1 Un bloc de visualisation,

2 Un bouton-poussoir RESET,

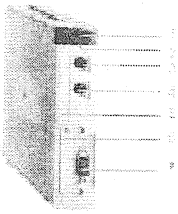
3 Un emplacement recevant une pile assurant la sauve-garde de la mémoire RAM interne du processeur,

4 Un volet assurant la protection de la face avant du module,

5 Un bornier à vis permettant le raccordement,

6 Un passage pour collier de serrage des câbles,

7 Un fusible assurant la protection.



Le processeur, qui définit la puissance du système, suivant le nombre d'entrées/sorties qu'il peut gérer, la vitesse de traitement, ...

1 Un bloc de visualisation comprenant des voyants signalisant l'état du processeur,

2 Un bouton RESET provoquant un démarrage à froid de l'automate,

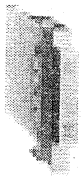
3 Une prise terminal TER : raccordement d'un terminal de programmation ou de réglage,

4 Une prise terminal AUX : raccordement d'un périphérique,

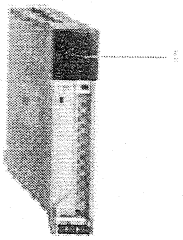
5 Un emplacement pour une carte d'extension mémoire,

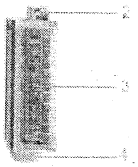
6 Un emplacement pour une carte de communication,

7 Un connecteur SUB-D 9 contacts pour communication sur bus de terrain.



La mémoire, définie en kilo-octets (ko) ou kilo-mots de 16 bits, qui stocke le programme de l'application.





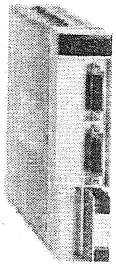
Les entrées/sorties TOR (tout ou rien), le nombre d'entrées/sorties positionne le niveau de l'automate programmable dans la gamme. Les entrées/sorties peuvent être locales ou déportées à distance.

1 Un bloc de visualisation des voies et de diagnostic du module,

2 Un bornier à vis débrochable pour raccordement direct des entrées/sorties aux capteurs et pré-actionneurs,

3 Une porte pivotante permettant l'accès aux vis du bornier servant également de support à l'étiquette de repérage,

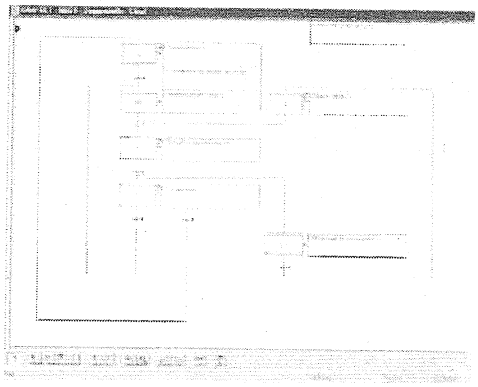
4 Un support rotatif comprenant le dispositif de détrompage.



Les coupleurs métiers où l'on retrouve toutes les fonctions particulières d'automatismes, par exemple :

- les traitement des fonctions de communication, les bus et réseaux,
- les fonctions analogiques et la régulation,
- les fonctions de pesage et de dosage,
- les fonctions de comptage décomptage rapide,
- les fonctions commande de mouvement,
- etc,...

II. LE LOGICIEL



Les langages de programmation des automates programmables industriels sont définies dans la norme IEC 1131-3 :

II.1. Langage à contacts (LD)

Un programme en langage à contacts est composé d'une suite de réseaux de contacts exécutée de façon séquentielle par l'automate. Chaque réseau de contacts peut être :

- récupéré par une étiquette,
- complété par un commentaire de 222 caractères.

Editeur de programmes : langage à contacts

L'éditeur du langage à contacts offre de nombreux outils assurant la construction des réseaux de contacts de façon conviviale : une palette d'éléments graphiques permet d'accéder directement, par la souris ou l'aide du clavier, aux différents symboles graphiques du langage : contacts, fil booléen, bobines, blocs opérations, blocs fonctions prédéfinis....

II.2. Langage littéral structuré (ST)

Le langage littéral, structuré est un langage évolué de type algorithmique particulièrement adapté à la programmation de fonctions arithmétiques complexes, manipulations de tableaux, gestion de messages ... Le langage littéral permettant la transcription directe d'une analyse à base d'organigramme est organisé en phrases. Chaque phrase est composée d'une étiquette (1000 étiquettes maxi), de commentaires (256 caractères maxi) et d'instructions.

Quatre structures de contrôle de phrase sont proposées :

- Action conditionnelle IF,
- Action itérative conditionnelle WHILE (action répétitive tant qu'une condition est vérifiée),
- Action itérative conditionnelle REPEAT (action répétitive jusqu'à ce qu'une condition soit vérifiée),
- Action répétitive FOR (action répétitive un certain nombre de fois).

II.3. Langage Grafcet (SFC)

Le langage Grafcet permet de décrire de manière simple et graphique la partie séquentielle d'automatismes. Il correspond au langage "Diagramme fonctionnel et séquence" SFC décrit dans la Norme IEC 1131-3. Le langage Grafcet SFC s'utilise uniquement dans une section de la tâche maître.

Les programmes écrits en langage Grafcet SFC se composent :

- Des macro-étapes qui sont la représentation unique d'un ensemble d'étapes et de transition,
- Des étapes auxquelles sont associées les actions à effectuer,
- Des transitions auxquelles sont associées les conditions (réceptivités),
- Des liaisons orientées reliant les étapes et les transitions.

Les actions (continues, impulsionnelles à l'activation ou la désactivation) et les réceptivités sont programmables langage soit à contacts, soit en littéral, soit en liste d'instructions.

II.4. Langage liste d'instructions (IL)

Le langage liste d'instructions est un langage représentant, sous forme textuelle, l'équivalent d'un schéma à relais permet d'écrire des équations booléennes et d'utiliser l'ensemble des fonctions disponibles dans le langage.

Un programme en langage liste d'instructions comprend une suite d'instructions des différentes familles suivantes :

- Instructions sur bit, par exemple lire l'entrée n° 3 : LD %I1.3,
- Instructions sur bloc fonction, par exemple lancer le temporisateur n° 0 : IN%TM0,
- Instructions numériques sur entier simple, double format et flottant, par exemple faire une addition : (%MW10 + %MW50 + 100),
- Instructions sur tableau de mots, chaîne de caractères, par exemple faire une affectation : (%MW10:10=%KW50:10),
- Instructions sur programme, par exemple appeler le sous-programme n° 10 : SR10.

Chaque instruction se compose d'un code instruction et d'un opérande de type bit ou mot.

II.5. Bloc fonctions utilisateur DFB

Les logiciels offrent à l'utilisateur la possibilité de créer ses propres blocs fonctions répondant aux spécificités de ses applications. Une fois créés en bibliothèque, ces blocs fonctions pourront être utilisés avec les logiciels de programmation.

Ces blocs fonctions utilisateur permettent de structurer une application, ils seront utilisés dès qu'une séquence de programme se trouve répétée à plusieurs reprises dans l'application ou pour figer une programmation standard.

L'utilisation d'un bloc fonction DFB dans une ou plusieurs applications permet :

- De simplifier la conception et la saisie du programme,
- D'accroître la lisibilité du programme,
- De faciliter sa mise au point (toutes les variables manipulées par le bloc fonction DFB sont identifiées sur son interface),
- D'utiliser des variables internes propres aux DFB donc indépendantes de l'application.

La mise en œuvred'un bloc fonction DFB s'effectue en trois phases :

- La conception du DFB qui se compose d'un nom, de paramètres (entrées/sorties) de variables et du code en langage littéral ou ladder,
- La création d'une instance DFB dans l'éditeur de variables ou lors de lappel de la fonction dans l'éditeur programme,
- L'utilisation de cette instance dans le programme de façon identique à un bloc fonction standard.

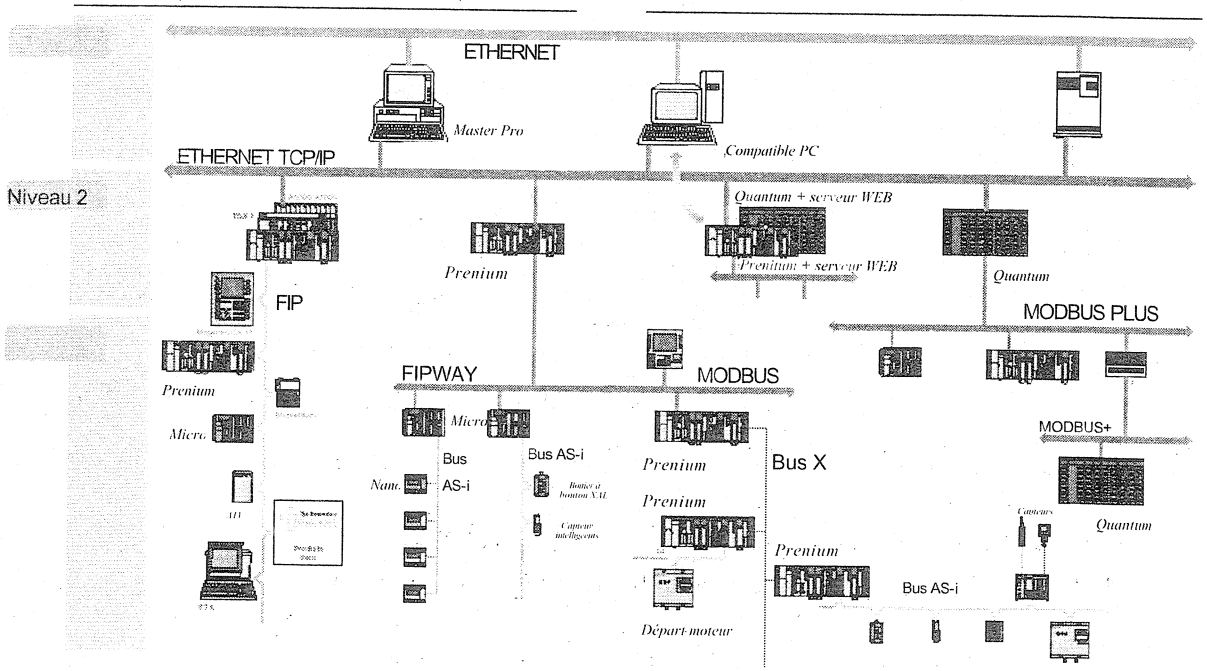
III. LES APPLICATIONS

L'automate programmable industriel est utilisé dans tous les secteurs de l'automatisation, que ce soit l'industrie, le bâtiment, la distribution d'énergie, les infrastructures,...et dans tous les types de procédé, manufacturier, semi-continu ou batch, continu.

L'API s'inscrit dans toutes les architectures d'automatisme modernes, intégrant toutes les techniques de communication d'aujourd'hui : ethernet, internet, intranet, bus de terrain,.....

Il intègre toutes les solutions d'automatismes telles que régulation des procédés, commande de mouvement, pesage et dosage,.....Nous vous proposons dans la suite quelques exemples de communication et d'applications d'automatismes tels la régulation des procédés industriels et le pesage / dosage.

IV. LA COMMUNICATION



La communication entre les différents systèmes permet la réalisation d'architecture d'automatismes sur les trois niveaux du CIM (Computer Integrated Manufacturing). Les réseaux se déclinent en réseau d'atelier (ETHERNET), réseau d'automatismes (FIPWAY, MODBUS) et bus de terrain (FIP, AS-i,...). Ils offrent différents services.

Le mécanisme d'adressage de la couche réseau (couche 3 du modèle OSI), permet l'interconnexion de plusieurs réseaux, ou segments, Ethernet, Ethernet TCP/IP et/ou Fipway. L'architecture de communication est conçue pour couvrir les applications multi-réseaux répondant aux problèmes de concentration, de redondance et de communication inter-réseaux

Les services fournis dépendent du type d'équipement (automate programmable, commande numérique, terminal de programmation, poste de supervision...), chacun pouvant, suivant sa fonction, être Client et/ou Serveur.

Un équipement Client peut accéder aux fonctions système d'un automate programmable (Serveur), même dépourvu de programme application, pour : lecture/écriture des objets langage (bits, mots...), chargement/déchargement de programmes, arrêt ou mise en marche...

Un automate Client peut par son programme application accéder aux autres équipements de l'architecture : lecture/écriture d'objets sur un autre automate ou une commande numérique, sélection de programme sur une commande numérique...

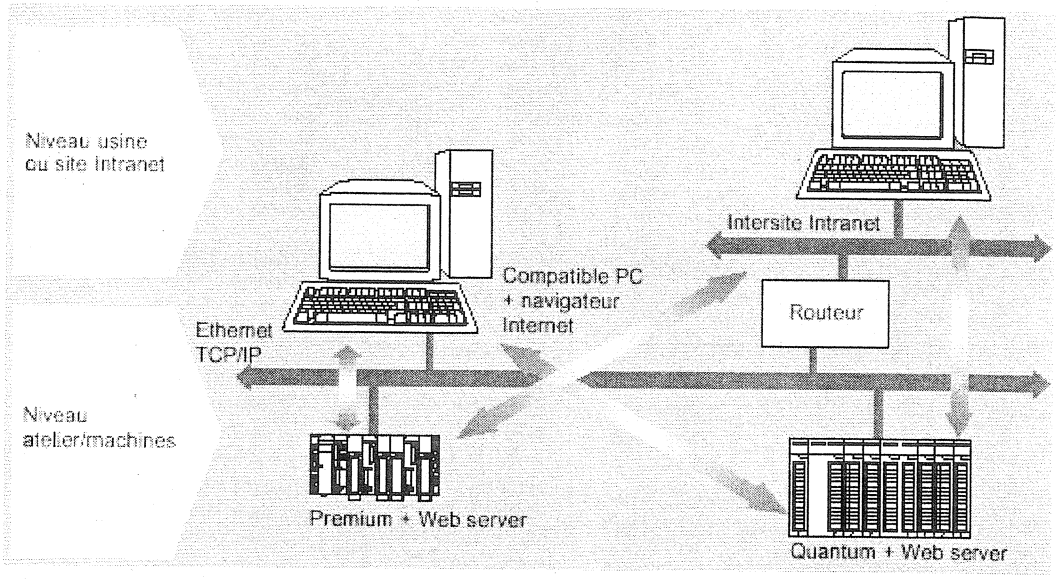
L'envoi de requêtes se fait à l'aide de :

- Bibliothèque de fonction de communication,
- Blocs fonction texte ou blocs fonction optionnels pour automates.

Les terminaux de programmation ou les superviseurs sont des clients. Un terminal connecté sur une quelconque station du réseau ou connecté directement au réseau Fipway/Ethernet TCP/IP, communique avec n'importe quelle autre station du réseau (tous les échanges sont transparents pour l'utilisateur) comme si le terminal était physiquement connecté sur l'automate avec lequel il établit le dialogue.

La transparence terminal est utilisable entre des stations connectées sur des segments différents Ethernet TCP/IP, Fipway, Fip d'une même architecture multi-réseaux.

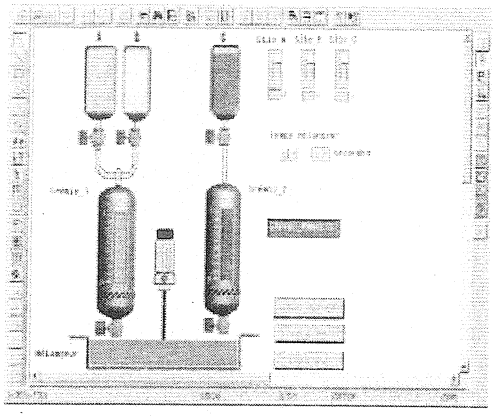
IV.1. Serveur Web embarqué



Le coupleur Ethernet comprend en plus la fonction Serveur Web embarqué. Ce serveur est un serveur de données temps réel automate. Toutes les données du processeur supportant le coupleur Serveur Web sont présentées sous forme de pages standard Web au format Html et sont ainsi accessibles par tout navigateur Internet du marché capable d'exécuter du code Java embarqué.

Toutes les fonctions apportées par le serveur Web ne nécessitent aucune configuration ou programmation ni au niveau de l'automate, ni au niveau du compatible PC supportant le navigateur Internet. De plus, ce coupleur peut s'utiliser dans une configuration existante sans aucune modification du programme résidant.

IV.2. Visualisation des pages Web prédéfinies



Le coupleur Ethernet avec Serveur Web embarqué TSX ETY 110 WS dispose d'un espace mémoire du type Flash EPROM (1), accessible, comme un disque dur et permettant l'accueil (hébergement) de pages Web définies par l'utilisateur.

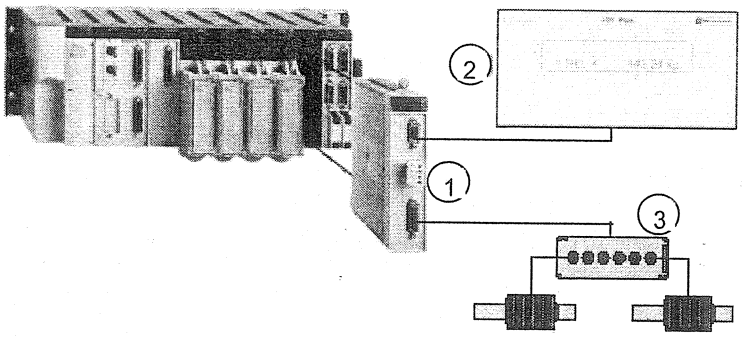
Ces pages Web peuvent être créées avec tout outil standard permettant la création et l'édition au format Html (FrontPage, Word, PowerPoint). Ces pages Web ainsi créées permettent entre autres de :

- Visualiser toutes variables automate en temps réel,
- Réaliser des hyper liens vers des serveurs externes (documentation fournisseurs...).

Cette fonction convient particulièrement à la création de graphismes et d'images destinés à des fins de :

- Visualisation, contrôle, diagnostic,
- Elaboration de rapports de production temps réel,
- Aide à la maintenance,
- Guides opérateur.
- ...

V. SYSTEME DE PESAGE INTEGRE



① *Module de pesage*

Le module de pesage, de format standard est l'élément central de la chaîne de pesage. Il dispose :

- D'une entrée de mesure pouvant recevoir jusqu'à 8 capteurs,
- D'une liaison plombable pour afficheur,
- De deux sorties réflexes "Tout ou rien" pour les applications de dosage pondéral.

Le module de pesage peut être livré étalonné en usine.

② *Indicateur de poids*

L'afficheur déporté affiche le poids mesuré sans aucune configuration préalable. Lorsque la liaison vers le module de pesage est plombée, l'afficheur devient alors afficheur principal.

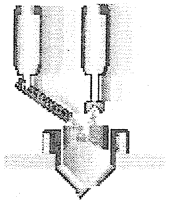
Pour transactions commerciales.

L'ensemble module/afficheur est conforme aux recommandations CIML et homologué CE pour bascules de classe III (6000 échelons) et pour bascules de classe IIII (1000 échelons).

③ *Capteurs*

Destinés à des conditions d'environnement sévère, les capteurs à jauges de contraintes constituent le 3ème élément de la chaîne de pesage. Ces capteurs peuvent être fournis étalonnés en usine ou à la presse. Capteurs, accessoires, (boîtiers, câbles, simulateurs...) plate-forme de pesage.

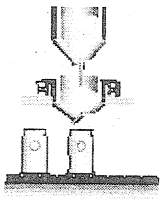
V.1. Solutions pour une mise en œuvre d'applications de pesage automatique



Dosage multi-produits

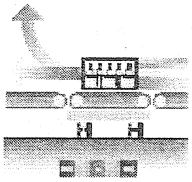
Dosage de 1 à 12 produits parmi 32,

stockés dans 16 silos.



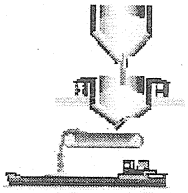
Dosage pondérale

Dosage de quantités déterminées dans un emballage ou une trémie intermédiaire.



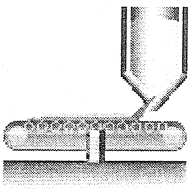
Trieuse pondérale

Contrôle et tri d'objets par pesée.



Totalisateur discontinu

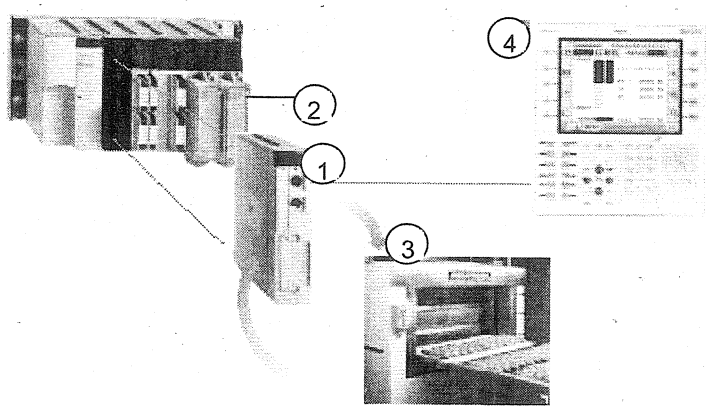
Totalisation de pesées successives pour réception ou expédition de matière en vrac.



Régulation de débit sur bande transporteuse et totalisateur continu T

Régulation de débit sur bande transporteuse par la masse ou la vitesse.

VI. REGULATION DE PROCEDES



① *Processeurs de régulation*

Les processeurs de régulation ont des caractéristiques identiques aux autres processeurs avec en plus la possibilité de gérer 10 voies de régulation (de 10 à 30 boucles). Ces voies peuvent être configurées afin d'exécuter des algorithmes de procédés industriels.

- Boucle cascade
- Boucle auto-adaptative
- Programmateur de consigne
- ...

② *Entrées/Sorties*

Comme tout processeur automate, les processeurs de régulation gèrent l'ensemble d'une station constituée de racks connectés sur le bus. Les interfaces d'entrées/sorties nécessaires aux traitements de régulation sont des voies de modules analogues ou "tout ou rien" insérées dans les racks ou déportées.

③ *Boucles de régulation*

La mise en œuvre logicielle des boucles de régulation se fait par paramétrage (technologie Plug and Play) lors de la configuration du processeur automate. L'utilisateur renseigne des schémas de boucles prédéfinies qui intègrent également la gestion des modes de marche et le lien avec les entrées/sorties.

④ *Dialogue opérateur et conduite*

Les terminaux de dialogue opérateur proposent des écrans pré-configurés dédiés à la régulation facilitant l'exploitation et la conduite des boucles. Ces écrans proposent des faces avant de régulateur ainsi que des vues de tendance et des vues de surveillance.

♦ Présentation

Les fonctions de régulation proposées par ces processeurs sont particulièrement adaptées pour :

- Les procédés séquentiels nécessitant des fonctions auxiliaires de régulation telles que les machines d'emballage, les machines de traitement de surface, les presses...
- Les process simples tels que les fours de traitement des métaux, les fours à céramiques, les groupes frigorifiques...
- Les asservissements ou les régulations mécaniques dont le temps d'échantillonnage est critique telles que la régulation de couple, la régulation de vitesse...

Les processeurs de régulation possèdent entre autres les caractéristiques suivantes :

- Les 10 voies de régulation configurables permettent de gérer de 10 à 30 boucles suivant le type de boucle choisie.
- Les processeurs utilisent l'ensemble de la gamme des entrées/sorties des automates.
- Les process de régulation peuvent s'insérer dans l'architecture globale d'un site, grâce à l'intégration de l'automate dans différents réseaux de communication.
- Les calculs liés à la régulation sont effectués en flottant, exprimés en unités physiques.

♦ Généralités

Les processeurs de régulation permettent la mise en œuvre de 10 voies de régulation adoptant chacune l'un des 5 profils de boucle suivant :

- La boucle de type **process** : boucle à un seul correcteur,
- Le régulateur à **3 boucles simples** : régulateur permettant d'augmenter la capacité du nombre de boucles,
- La boucle **auto-sélective** appelée aussi **sous contrainte** : composée de 2 boucles en parallèle avec un algorithme de sélection de la sortie,
- La boucle **cascade** : composée de 2 boucles dépendantes (la sortie de la boucle maître est la consigne de la boucle esclave),
- Le **programmateur de consigne** : comprenant 6 profils maxi composés, au total de 48 segments.

Les 10 voies sont indépendantes ce qui permet par exemple d'obtenir :

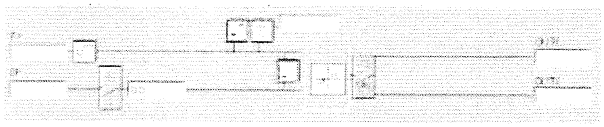
- 30 boucles simples,
- 5 programmateurs de consigne, associés chacun à 5 boucles de régulation,
- 2 programmateurs de consigne et 8 boucles process.

Les différentes boucles se caractérisent par :

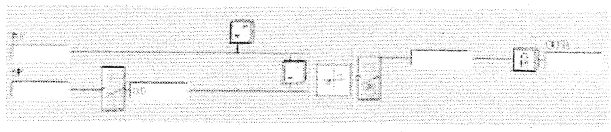
- Leurs algorithmes différents,
- 5 branches de traitement (mesure, consigne, Feed Forward, correcteur et traitement de la sortie),
- Des fonctions de calculs (gain, filtrage, racine carrée...) définies à l'aide de paramètres.

- **Types de boucles**

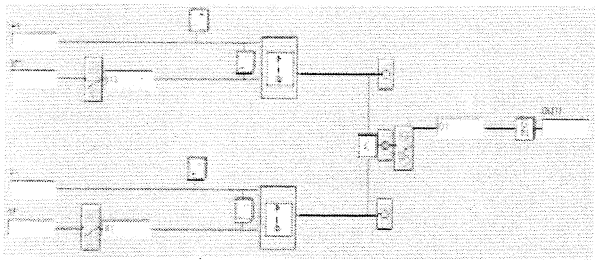
Boucle process



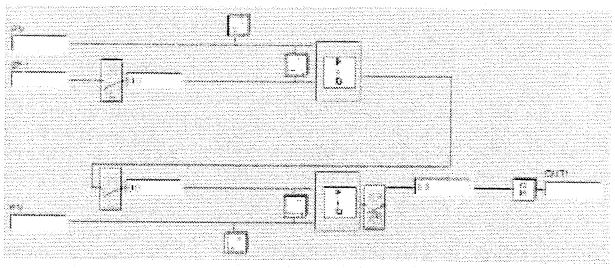
Boucle simple



Boucle auto-sélective



Boucle cascade



- **Branches de traitement**

Le paramétrage (choix des fonctions à utiliser) des profils de boucles de régulation permet d'adapter l'algorithme au process à commander.

Traitement de la mesure

Le traitement de la mesure peut se faire soit de façon standard soit de façon externe.

Traitement standard, l'utilisateur dispose des fonctions suivantes : filtrage, limitation de la mesure entre bornes, générateur de fonctions avec mise à l'échelle, gestion d'alarmes sur dépassement de seuils, totalisateur et simulation de la valeur mesurée.

Traitement externe, il permet d'avoir en entrée de correcteur une valeur de mesure PV (Process Value) dont le traitement a été effectué en dehors de la boucle de régulation. Cette solution est utile, si le calcul de la mesure nécessite des fonctions spécifiques ou personnalisées.

- **Traitement de la consigne**

Selon le type de boucle choisi, il est possible d'opter pour l'un des 4 types de consignes suivantes : consigne de ratio, consigne de sélection, consigne simple ("remote" avec mise à l'échelle) ou programmateur de consigne. Dans le cas du régulateur à 3 boucles simples ou de la boucle de contrainte (dans une boucle auto-sélective), seuls la consigne simple et le programmateur de consignes sont utilisables.

- **Traitement Feed Forward**

Le traitement Feed Forward permet de compenser une perturbation mesurable dès qu'elle apparaît. Ce traitement en boucle ouverte, anticipe l'effet de perturbation. Il dispose de la fonction "Leading" (avance/retard de phase).

- **Correcteur et traitement de la commande**

Le correcteur peut être choisi parmi les 6 types suivants : PID auto réglant, correcteur en mode "Tout ou Rien" à 2 ou 3 états, correcteur chaud/froid (PID ou à modèle auto réglant) ou correcteur Split Range (PID ou à modèle auto réglant).

- **Traitement de la sortie**

Le traitement de la sortie peut se faire selon 3 types : sortie analogique, sortie servomoteur ou sortie PWM. Quel que soit le type de sortie, la commande calculée par le correcteur traverse un limiteur de gradient et un limiteur dont les bornes inférieures et supérieures permettent de définir la plage de variation de la sortie.

- **Programmateur de consigne**

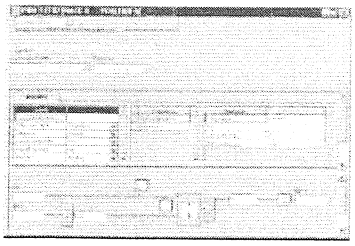
Le programmateur de consigne propose 6 profils maxi composés au total de 48 segments. Il est ainsi possible de réaliser un programmateur de 48 segments, six programmateurs de 8 segments ou un programmateur de 24 segments avec un programmateur de 16 segments et un programmateur de 8 segments...Chaque segment est configuré en tant que rampe ou palier. Il est caractérisé par :

  - La consigne à atteindre,
  - La durée du segment ou pente du segment (s'il s'agit d'une rampe).

Un profil peut être exécuté une fois, un certain nombre de fois ou rebouclé en continu. De plus, la notion de palier garanti permet de ne décompter le temps que si la mesure est bien dans la plage spécifiée.

- **Configuration des voies de régulation**

  Des écrans spécifiques, accessibles à partir des logiciels de programmation, permettent la configuration des boucles de régulation. La configuration des boucles process, simples, auto-sélectives et cascades propose un paramétrage par défaut. Les différentes fonctions intégrées dans les algorithmes (racine carrée, générateur de fonction...) et la valeur initiale de chaque paramètre sont prédéfinies.



Exemple : configuration d'une boucle process

Le type de boucle ayant été choisi, le paramétrage de celle-ci est réalisé en sélectionnant ou désélectionnant des options dans les branches de traitements. Aucune programmation n'est donc nécessaire, les schémas des boucles s'enrichissent ou se simplifient au fur et à mesure de la validation des paramètres.

Ci-contre, la sélection du correcteur PID permet de visualiser les différents paramètres valides pour ce type de correcteur (KP, TI, TD...).

- **Exécution des voies de régulation**

  La période d'échantillonnage des boucles est prédéfinie à 300 ms. Celle-ci définit la période de traitement du correcteur en mode automatique. Il est possible de modifier cette période dans l'écran de configuration de la boucle.

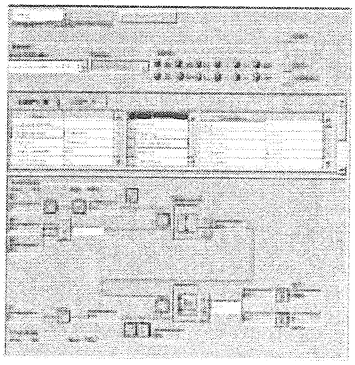
  L'ensemble des entrées/sorties et les paramètres des différentes voies de régulation configurées sont accessibles à l'utilisateur au niveau du programme ou via les différents outils des logiciels de programmation (en particulier les éditeurs langages et les tables d'animation).

- **Fonctions de mise au point**

  Le réglage et la mise au point des boucles de régulation s'effectuent de façon simple et conviviale entre autre à l'aide de l'écran métier de configuration des boucles qui, en mode connecté, donne accès aux fonctions suivantes :

  - Visualisation et animation du schéma de l'algorithme de la boucle,
  - Visualisation des alarmes liées au process et des défauts de la voie,
  - Simulation des valeurs des interfaces d'entrées : par exemple dans le cas où celles-ci sont non connectées (mesure, Feed Forward),
  - Ajout, suppression ou remplacement de fonctions de calcul en mode connecté,
  - Modification des paramètres de réglage de chacune des fonctions,
  - Modification des modes de marche du régulateur et pilotage en manuel.

  Avec les correcteurs intégrés aux boucles de régulation, il est possible d'utiliser la fonction autoréglage qui calcule un jeu de paramètres de réglage (Kp, Ti, Td ou Ks, T1, T-delay) sur demande. Une fois la mise au point de la boucle effectuée, il est possible de sauvegarder les valeurs courantes issues des tests, dans les valeurs initiales des paramètres de la boucle. De cette façon au redémarrage de la boucle, celle-ci repartira avec des valeurs correctes.



Mise au point d'une boucle

L'écran de mise au point permet de :

- Visualiser en dynamique les valeurs des variables liées à la boucle,
- Connaître les paramètres choisis (voire de les modifier),
- Visualiser les alarmes.

Les menus vont permettre de piloter en manuel la boucle, de réaliser un autoréglage, de sauvegarder les paramètres...

Mise au point du programmateur de consigne

Les voies programmateur de consigne disposent de leur propre écran de mise au point permettant de visualiser :

- Le numéro du segment en cours et de l'itération,
- Le temps d'exécution du segment en cours,
- Le temps d'exécution global.

QUELLES COMMANDES D'AXES POUR LES AUTOMATES PROGRAMMABLES INDUSTRIELS ?

Philippe VESSE
OMRON-ELECTRONICS

Introduction

Les nouveaux développements dans le domaine de la commande d'axes proposent aujourd'hui des systèmes plus rapides et plus puissants alors qu'il était déjà plus simple à programmer et plus facile à intégrer.

Cela ne signifie pas que construire des machines autour des systèmes de contrôle électroniques les rendent implicitement plus rapides que celles qui emploient des systèmes de contrôle mécaniques mais indéniablement l'amélioration des performances se fait surtout en terme de flexibilité. Ainsi le changement de produit ou de format peut maintenant s'effectuer par une simple action de l'opérateur quand les systèmes mécaniques nécessitent une mise en place et des réglages.

La commande d'axes électronique n'est pas le simple remplacement de la mécanique. C'est une approche complètement différente pour le design des machines. En effet le savoir-faire des ingénieurs mécaniciens est plus que jamais requis mais maintenant ils doivent se concentrer sur le développement de solutions élégantes pour réduire les masses et les inerties plutôt que sur l'étude des liens mécaniques puisque les charges seront le plus souvent en lien direct avec les actionneurs qu'ils soient moteurs, servomoteurs, etc. ...

La commande d'axe électronique propose des avantages spécifiques sur les systèmes mécaniques en offrant des possibilités de profiles de mouvements plus adaptés sollicitant moins la mécanique, de synchronisation complexes de mouvements permettant ainsi des cadences plus élevées tout en diminuant l'usure et donc la maintenance.

La commande d'axe électronique s'est également développée en proposant aujourd'hui des solutions pour remplacer des systèmes mécaniques simples tel que vérins hydrauliques ou pneumatiques, embrayages, embrayages frein, butée mobile en somme tous ces systèmes qui subissant une sollicitation importante sont à la fois difficiles à maintenir en état et à modifier en fonction des différentes productions.

La flexibilité recherchée dans tous les secteurs industriels, les nécessaires gains de productivité, passant par une diminution des arrêts machines et d'une maintenance facilité, ont grandement poussé au développement des solutions de commandes d'axes électroniques comme remplacement des solutions mécaniques utilisées par le passé; il est clair également que ce changement n'aurait pu s'opérer dans les mêmes proportions sans les avancées technologiques réalisées dans le domaine des actionneurs et particulièrement en ce qui concerne ce qui est devenu l'une des solutions les plus utilisées à savoir le moteur asynchrone avec des variateurs de fréquence de plus en plus performants et les servomoteurs qui, réservés par le passé uniquement aux applications haut de gamme, sont aujourd'hui utilisés couramment du fait de leur rapport performance / coût de plus en plus attractif. Ceci étant dû à la fois aux avancées de l'électronique de puissance et des techniques de fabrications de moteurs eux-mêmes.

Mais il faut bien le dire, la mise en place de solution commande d'axes électroniques ne va pas sans problème. Ainsi la facilité d'intégration est le principal souci des utilisateurs quand il s'agit de savoir comment réaliser le câblage de tous les éléments, de faire communiquer ces différents éléments et enfin de mettre en œuvre l'ensemble de l'application.

Dans ce contexte, s'il existe sur le marché une vaste palette de solutions pour réaliser une commande d'axes électronique, l'utilisation déjà très répandue de l'automate programmable industriel (API) a

largement contribué à ce que la commande d'axes intégrée dans les API soit aujourd'hui une solution pour bons nombres d'applications des plus simples au plus complexes.

De plus si l'on considère qu'un système typique comprendrait un automate programmable industriel, un système de commande d'axes, une interface Homme Machine et des actionneurs, intégrer tous ces produits relève parfois du challenge particulièrement pour la communication entre l'API et le système de commande d'axes. Là où la vitesse et la fiabilité des échanges sont primordiale pour atteindre les performances souhaitées en terme de gestion de mouvements comme de flexibilité de la machine, la solution d'intégrer la commande d'axes dans l'API s'impose alors souvent aux utilisateurs.

Le but ici n'est pas de présenter une description exhaustive de toutes les solutions commandes d'axes électroniques utilisées dans le domaine industriel tant le champ d'applications est large et les solutions nombreuses mais de décrire les solutions les plus diffusées et intégrées dans les API.

Quelles commandes d'axes pour les automates programmables industriels ?

Tout système de commande d'axes peut se définir par rapport à :

- Son système de contrôle
- Son contrôle de mouvement
- Son type d'asservissement et de la commande à l'actionneur.

1 – Son système de contrôle

Quel que soit l'actionneur utilisé et le mouvement généré on définit trois types de systèmes de contrôle comme illustré sur la figure 1 :

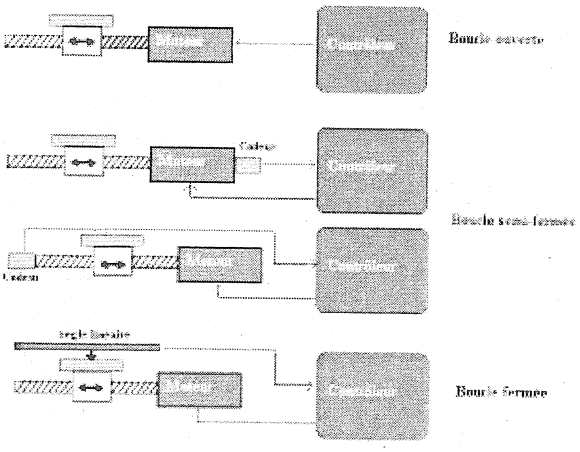


Figure 1

Le choix du système de contrôle retenu dépendra des caractéristiques comme des performances attendues. Ainsi si la précision de positionnement du mobile est prépondérante, on choisira un système boucle fermée puisque dans ce cas la précision de la position lue sera indépendante de la précision du système mécanique mais dépendra uniquement de la précision du capteur. Le système boucle ouverte ne permettant de s'assurer de la précision de la position atteinte, il faudra soit pouvoir être en mesure de négliger celle ci soit de faire en sorte en limitant les variations de charge ou la réponse dynamique demandée que le mobile réalise des cycles de déplacements où les erreurs de positionnement n'iront pas en s'accumulant, des cycles où l'actionneur sera dimensionné de manière à ce que ces déplacements soient « sécurisés ».

Outre la précision de positionnement requise, la vitesse de déplacement et donc le temps de positionnement est également déterminant dans le choix du système de contrôle. Comme nous venons de le voir pour ne pas entamer la qualité du positionnement, un système boucle ouverte limitera les performances en vitesse alors que celles ci pourront être identiques en boucle fermée et en boucle semi-fermée.

A l'inverse des deux critères précédents que sont la vitesse et la précision du positionnement, le système boucle ouverte propose l'avantage de la simplicité facilitant la mise en œuvre de cette solution. Simplicité allant de paire avec le coût limité du point de vue de la commande mais également des actionneurs et de la mécanique.

Par ailleurs la nature même de certains actionneurs les destine à s'intégrer dans un système boucle ouverte (les moteurs pas à pas par exemple) alors que d'autres comme les servomoteurs peuvent tout à la fois être utilisés aussi bien en boucle semi-fermée qu'en boucle fermée si la précision requise l'impose.

Ainsi on peut résumer par des critères simples présentés dans le tableau 1:

	Boucle ouverte	Boucle semi-fermée	Boucle fermée
Précision	-	+	+++
Vitesse / Temps de position[1]	-	++	+++
Mise en œuvre	+++	++	+
Actionneurs typiques	Moteur pas à pas	servomoteurs	Moteurs linéaires

Tableau 1

Les solutions commande d'axes disponibles aujourd'hui sur les API permettent de mettre en place les trois types de systèmes de contrôle même s'il faut remarquer qu'à l'heure actuelle la grande majorité des applications industrielles gérées par des API sont soit en boucle ouverte soit en boucle semi-fermée.

2 – Contrôle de mouvement

Quel que soit le système de contrôle retenu en fonction des performances attendues, le choix de la commande d'axes doit se faire en fonction du type de

mouvement qu'elle peut générer et contrôler. On peut alors distinguer trois types de mouvements ou d'applications :

- Contrôle Point à Point
- Contrôle de trajectoire
- Contrôle de synchronisation

- **Contrôle Point à Point**

Dans ce cas le positionnement est contrôlé indépendamment sur chaque axe. La trajectoire du mobile comme le temps de positionnement dépendra ici des vitesses de déplacements fixées pour chacun des axes.

Ce mode de contrôle permet de s'adresser aux applications les plus simples où le nombre d'axes commandés peut être important mais où chaque axe opère indépendamment. Finalement, ce mode de contrôle est plus souvent utilisé en complément du contrôle de trajectoire par exemple pour réaliser toutes les opérations de reprises de cycles, dégagements ou les opérations semi-automatiques.

Contrôle de trajectoire

Contrairement au contrôle point à point, ce n'est plus seulement le point de départ et la position cible qui sont pris en compte mais également la trajectoire décrite entre ces points (figure 2)

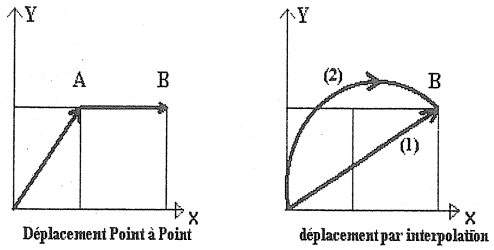


Figure 2

On notera ici les formes les plus courantes pour décrire une trajectoire, les déplacements par interpolation linéaire (1) ou par interpolation circulaire (2). Sachant qu'il existe également dans les API des cartes de commandes d'axes pouvant gérer des interpolations circulaires hélicoïdales (3axes). Les différents systèmes existants se distinguent ensuite par les nombres d'axes pouvant être gérés par interpolation.

Il est important de préciser les limites d'applications généralement à prendre en compte. C'est ainsi que ces systèmes permettant de réaliser interpolation linéaire et circulaire s'ils sont conçus pour la commande d'applications telles que : table X/Y, palettiseur, système de pose, suivi ou découpe de profiles ou des systèmes d'assemblages comme de manipulateurs, des robots simples comme les robots orthogonaux ou bien encore des systèmes automatiques comme des opérations de bobinage, ces systèmes ne sont pas conçus pour réaliser des interpolations linéaires ou circulaires avec des robots articulés ou cylindriques dans la mesure où les conversions de coordonnées d'un plan à l'autre ne sont pas possibles

Il apparaît clairement que le contrôle point à point et le contrôle de trajectoire sont devenus les domaines de prédilection des API dans les applications industrielles fixant également en général les limites de leurs utilisations.

Contrôle de synchronisation

Le contrôle point à point et le contrôle de trajectoire ont en commun le fait que la position à atteindre à un moment donné soit connue et fixée à l'avance. Dans le cas des applications de synchronisation, la position à atteindre à un instant t n'est pas connue à priori mais sera fixée en fonction du mouvement décrit par un ou plusieurs axes.

Même s'il faut distinguer les applications nécessitant une synchronisation en vitesse, de celles nécessitant une synchronisation en vitesse et position, dans les deux cas, ces applications plus complexes ne sont pas le plus souvent traitées par des systèmes API même si des développements récents font apparaître des solutions dans ce domaine.

3 – Type d'asservissement et de commande à l'actionneur

On pourrait envisager de construire un système de commande d'axe avec n'importe quelles combinaisons entre l'asservissement et la commande à l'actionneur. Pour des raisons d'habitudes et des réalités techniques, la commande la plus répandue est la commande analogique (le plus souvent +/- 10 V) et la commande par train d'impulsions moins répandue est réservée aux applications « boucle ouverte ».

Asservissement boucle ouverte et commande par train d'impulsions.

En boucle ouverte et commandé par train d'impulsions, la position à atteindre est fixée par le nombre d'impulsions que l'actionneur reçoit. Aucune information en retour n'est requise et la vitesse de déplacement sera elle proportionnelle à la fréquence des impulsions. L'actionneur le plus utilisé est le moteur pas à pas.

Il existe aujourd'hui des variateurs pour servomoteurs qui traitent les trains d'impulsions. L'asservissement reste boucle ouverte du point de vue de la commande d'axes puisque ni l'information vitesse ni l'information position n'est ramenée à la carte de commande mais dans ce cas le variateur pour servomoteur reçoit une consigne vitesse et position par le train d'impulsions et réalise l'asservissement vitesse et position. (figure 3)

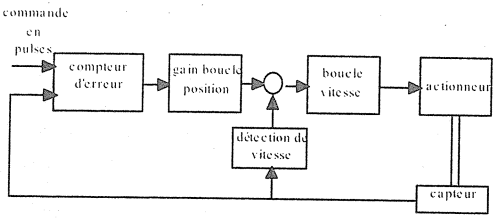


Figure 3

Dans les deux cas, moteur pas à pas ou servomoteurs, la carte de commande d'axes pourra traiter une information tout ou rien « position atteinte » en provenance soit du variateur soit d'un capteur pour l'enchaînement des différents déplacements.

Si l'utilisation des servomoteurs est un bon compromis entre la simplicité et la dynamique de servomoteurs, l'utilisation de cette solution est limitée par les raisons suivantes :

- Pour le système de commande la position courante réelle du mobile est inconnue pendant le déplacement ce qui interdit de déclencher une action en fonction de cette position.
- La fréquence du train d'impulsions étant réalisée à partir d'une fréquence d'horloge fixe divisée pour obtenir la fréquence voulue, la précision de la commande en vitesse dépend de la fréquence d'horloge utilisée.

(Il faut noter que cette erreur sur la vitesse n'influe en rien sur la précision du positionnement)

- Du fait de cette imprécision en vitesse, si les cartes de commandes d'axes par train d'impulsions gèrent des mouvements par interpolations linéaires, elles ne peuvent en aucun cas gérer des interpolations circulaires ou des applications de synchronisations puisque certaines valeurs pour la consigne en vitesse sont impossibles.

En conclusion, offrant un bon compromis simplicité / performance, la solution commande par train d'impulsions est maintenant couramment utilisée et présente dans les systèmes API mais reste cependant dédiée aux applications simples.

Asservissement boucle fermée ou semi-fermée et commande analogique.

C'est de loin la solution la plus utilisée dans le domaine de la commande d'axes que ce soit dans le cadre des API ou dans tous les autres. Tout d'abord parce que la commande analogique ne connaît pas les limites de la commande par train d'impulsions mais également car la commande analogique permet de travailler avec tous les types d'actionneurs que ce soient les moteurs électriques couramment utilisés ou tous autres types pneumatiques, hydrauliques, etc. ...

Nous l'avons vu le système boucle semi-fermée est le plus utilisé avec lui l'actionneur le plus courant est le servomoteur. Nous utiliserons ce système pour illustrer la commande analogique. Le schéma de principe du point de vue de la commande est représenté figure 4 :

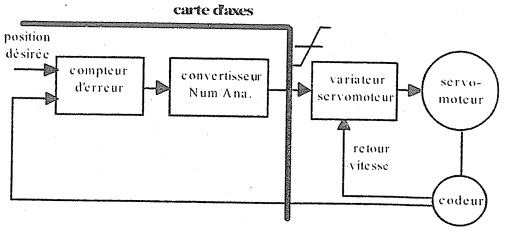


Figure 4

- Le compteur d'erreur la position cible en unités impulsions codeur.
- Le compteur d'erreur est directement relié au convertisseur numérique analogique où les pulses reçus par le compteur d'erreur sont converties en tension analogique.
- Cette tension devient la consigne au variateur servomoteur.
- Le variateur commande en rotation le moteur à une vitesse de rotation proportionnelle à la consigne analogique.
- Le codeur moteur en synchronisme avec la rotation du moteur renvoie des impulsions en retour.
- Le contenu du compteur d'erreur diminue d'autant des impulsions en retour jusqu'à ce que le compteur d'erreur soit nul. Ainsi la tension analogique devient nulle et le moteur s'arrête.

Dans les asservissements en vitesse et position les correcteurs P.I et P.D sont les plus couramment utilisés.

Action P.D (proportionnelle. Dérivée) :

Cette action correspond à une compensation par avance de phase. En effet, l'action dérivée entraîne une réponse en « avance » par rapport à la consigne.

Le terme P permet d'augmenter la précision, le terme D au risque de l'instabilité diminue l'erreur de traînage. Dans la pratique, l'utilisation du terme dérivée est souvent remplacée par un correcteur « Feed-Forward » correspondant à une correction en avance exprimée en pourcentage de la commande.

Action P.I (proportionnelle. Intégrale) :

Cette action correspond à une compensation en retard de phase. En effet, l'action intégrale entraîne une réponse en « retard » par rapport à la consigne. En compensant le risque d'instabilité du système, le terme intégrale agit au détriment du temps de réponse.

Ainsi dans la pratique, les boucles d'asservissements en vitesse sont le plus souvent du type P.I avec éventuellement une compensation de gain et les boucles d'asservissements en position sont du type P avec Feed-Forward. Dans la plupart de cas, les

cartes de commandes d'axes boucle fermée ou semi-fermée avec commande analogique assure une boucle en position du type P avec Feed-Forward. Certaines cartes d'axes cependant disposent de correcteurs plus complets de type PID avec Feed-Forward qui peuvent être rendus nécessaires pour assurer des applications à dynamiques élevées ou de synchronisation exigeantes.

En conclusion, les cartes de commandes d'axes train d'impulsions et analogiques se partagent les applications industrielles proposant chacune des avantages en fonction des performances attendues et de la nécessité de limiter les coûts.

Les cartes de commandes d'axes dans les API sont donc à même de répondre à un large champ d'applications également parce que les fonctions qu'elles proposent ne se limitent pas aux différents types d'asservissement. En effet les cartes de commandes d'axes API proposent un grand nombre de fonctions leur permettant de prendre en charge la totalité de la gestion de l'axes ou des axes. Citons par exemple :

- Recherche et retour à l'origine (gérant ainsi les fins de course et les capteurs Origine et proximité d'origine).
- Apprentissage (permettant d'enregistrer les positions)
- Correction du jeu mécanique, gestion du frein pour les axes verticaux.
- Gestion des arrêts d'urgence et des séquences de reprise
- Etc. ... .

Chaque carte d'axes se distinguant ensuite par le moyen de programmer ou de paramétrer les positions et les séquences de déplacement. Certaines disposant de leur mémoire propre, c'est dans cette mémoire que seront stockées positions et séquences. Il peut alors s'agir de tables de données ou de programme en langage évolué (programmation ISO par exemple) ou bien encore d'une combinaison des deux.

Mais si les cartes de commandes d'axes dans les API ont pu ainsi s'imposer dans de nombreuses applications, certains domaines sont encore fermés à ces solutions tout comme ils le sont aux API eux-mêmes comme la robotique industrielle ou la machine outils où d'une manière générale des fonctions spécifiques sont nécessaires comme la possibilité de gérer un grand nombre de points pour décrire des trajectoires précises ou nombreux paramètres de corrections des positions, trajectoires et des vitesses.

La commande d'axe dans les API avait commencé à partir de fonctions de comptage simple et d'actions Tout ou Rien, nous en sommes d'ores et déjà à des cartes de commandes d'axes offrant des plus en plus de fonctions et s'ouvrant sur des applications de plus en plus complexes suivant ainsi l'évolution même des API s'adressant à un champ de plus en plus large d'applications. L'apparition de réseaux dédiés et de supports adaptés pour la transmission d'informations pour la commande d'axes ouvrent de nouvelles possibilités à leurs utilisations dans les applications multi-axes et synchronisation évoluée.

GRAFCET ET AUTOMATES PROGRAMMABLES :
NORMES ET REALITES

Marcel GRANDPIERRE
Professeur - ENSEEIHT
(Ecole Nationale Supérieure d'Electrotechnique, d'Electronique, d'Informatique et d'Hydraulique de Toulouse)
Département de formation Génie Electrique-Automatique
2, Rue Camichel 31071 Toulouse Cedex
Email: marcel.grandpierre@leei.enseeiht.fr

Résumé : ***Le GRAFCET (graphe de commande étapes/transitions) est aujourd'hui un outil connu et reconnu dans le monde des automaticiens pour formaliser et mettre en œuvre les parties commandes des automatismes.***

Les normes nationales et internationales dont il fait l'objet pourraient laisser croire qu'il est aujourd'hui parfaitement abouti et donc sans problème ni interrogation.

Mais attention ! Ses principes, syntaxes et règles ne sont pas aussi simples que leur seul énoncé peut le laisser croire. Des commentaires, des analyses, des illustrations s'avèrent indispensables notamment pour ceux qui ont en charge sa présentation pédagogique.

Les automates programmables (A.P.I.), outils privilégiés de la mise en œuvre du GRAFCET nécessitent encore une approche quelque peu prudente. S'agissant de systèmes relativement "fermés" dans leurs aspects informatiques internes, l'automaticien n'aura souvent d'autres moyens d'évaluation que de les soumettre à des tests spécifiques. Il pourra ainsi caractériser de façon claire leur comportement dans telle ou telle situation et notamment par rapport à des configurations parfaitement admises par les normes GRAFCET mais dont la prise en compte s'avère parfois sujette à interprétation. Dès lors averti et rassuré il pourra développer ses applications en se concentrant sur son métier : les automatismes et l'informatique industrielle

Introduction

" Il faut rendre hommage aux logiciens qui ont fait preuve d'un certain courage en brûlant ce qu'ils ont adoré. La méthode d'Huffman, en particulier, a été ramenée à sa juste place, celle d'un moyen efficace pour l'enseignement de la logique et la formation du logicien..., mais en aucun cas une méthode pratique universelle pour aborder la conception des grands systèmes. Souhaitons que cette attitude courageuse ouvre de nouveaux horizons à la logique de demain".

Voilà la conclusion de M. Blanchard dans un ouvrage qu'il a consacré au GRAFCET [1]. Les années ont passé et force est de constater que le GRAFCET est devenu un outil particulièrement prisé par les automaticiens et les informaticiens industriels. Sa genèse quelque peu atypique l'a pourtant longtemps exposé à un certain nombre de critiques. La simplicité apparente de son graphisme ne sous-entendait-elle pas un manque de formalisme théorique ? Dans le même temps en effet, le formalisme et le calcul matriciel aisément associables aux Réseaux de Pétri, se voulaient beaucoup plus rigoureux et par la même beaucoup plus rassurants.

Pourtant, depuis sa première normalisation en 1982, le GRAFCET à fait son chemin et il s'est petit à petit imposé comme outil convivial et efficace pour modéliser une grande partie des automatismes industriels mais aussi aujourd'hui pour générer les logiciels de commande adéquats. A cela sans doute trois raisons essentielles :

- ✓ L'introduction de l'enseignement du GRAFCET à tous les niveaux de formation dans le domaine des automatismes. Au programme, du CAP aux diplômes d'ingénieurs, il constitue aujourd'hui une sorte de "culture" commune à tous les automaticiens [2].
- ✓ Le développement et la diffusion d'automates programmables industriels (A. P. I.) permettant une mise en œuvre aisée et de plus en plus transparente.

- ✓ L'offre d'ateliers de développement logiciel utilisant les possibilités les plus avancées de l'informatique actuelle.

Le GRAFCET s'appuyant sur des normes tant dans son formalisme [3] [4] que dans sa mise en œuvre au moyen d'automates programmables [5] il semblerait naturel de penser que son utilisation ne se prête guère à interprétation et que les problèmes soulevés ne peuvent qu'être secondaires et exceptionnels.

Il en va effectivement ainsi dans la majorité des applications industrielles qui se particularisent plus souvent par leur taille que par leur complexité. L'intégration du GRAFCET dans des enseignements de bas ou moyens niveaux peut donc légitimement se contenter d'une "approche outil". Une bonne connaissance des règles de syntaxe et d'évolution et une manipulation correcte des outils logiciels de développement, de codage et de mise au point suffisent à rendre un technicien opérationnel.

Il existe pourtant des cas qui, s'ils ne sont pas les plus nombreux sont évidemment les plus délicats, où le GRAFCET se heurte à des contraintes structurelles ou temporelles qui obligent à se poser la question du bien fondé de son utilisation. C'est alors la question du choix de l'outil qui se pose, de la confrontation de ses possibilités et de ses limitations avec les contraintes du problème traité. Il est dès lors essentiel de remonter au niveau des concepts, des différents postulats et hypothèses qui doivent être clairement énoncés. De même la distance modèle théorique / réalisation pratique doit-elle être évaluée au plus juste.

Nous allons ci-après essayer d'illustrer un certain nombre de difficultés que nous avons ressenties à travers 20 ans de pratique et d'enseignement du GRAFCET et de l'informatique industrielle. Pour chacun des points abordés nous essayerons de proposer un exemple simple qui le met en évidence et qui permet notamment de caractériser le comportement de tel ou tel automate programmable par rapport au problème évoqué.

I. LOGIQUE PROGRAMMEE, ORGANIGRAMMES ET GRAFCET

L'énoncé des règles de syntaxe et des règles d'évolution du GRAFCET ne plonge généralement pas l'auditeur dans un abîme de perplexité. Les exposés en sont clairs et concis et leur compréhension semble relever de raisonnements relativement élémentaires. Les choses se compliquent pourtant singulièrement lorsqu'il est question de réalisations programmées. Il est alors assez classique de voir des amalgames entre organigramme et GRAFCET ou entre programme bouclé et GRAFCET connexe par exemple. Précisons par des exemples quelques-uns uns de ces problèmes:

I.1. Logique câblée / logique programmée

Il est essentiel pour les automatismes que la partie commande (PC) soit en liaison permanente avec son environnement (PO) au travers d'entrées (capteurs) et de sorties (actionneurs). Ce sont les événements (changement d'état des entrées) qui provoquent d'éventuels changements d'état des sorties sans que les instants ou se produisent ces changements ne soient maîtrisables par la partie commande. Le parallèle entre réalisation câblée et réalisation programmée peut être illustré par la figure 1 :

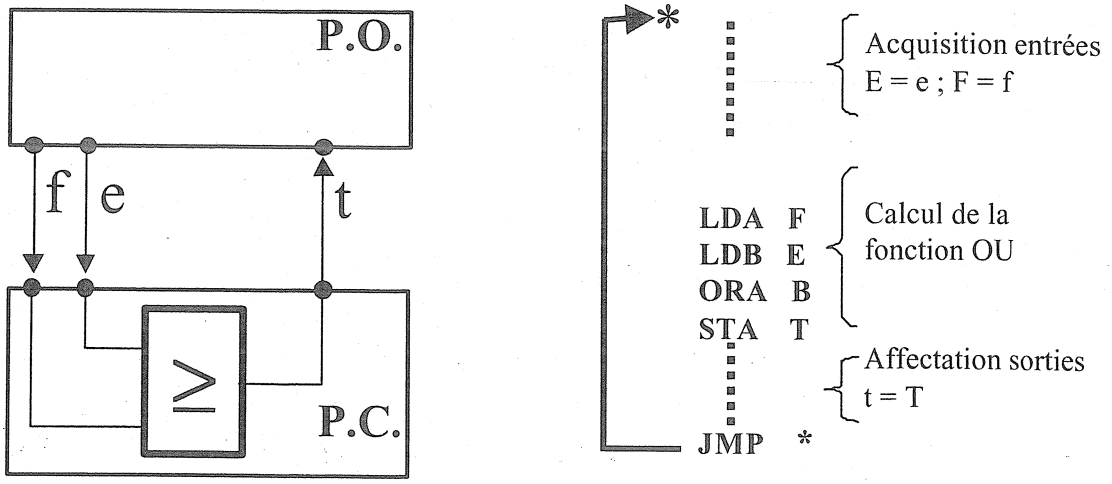


Fig.1 : Schéma de réalisation câblée et programmée d'un opérateur OU

De part sa liaison physique permanente avec le système, la solution câblée est toujours **attentive** aux éventuels changements d'état des entrées. Pour assurer cette même propriété au niveau programmé il est indispensable de boucler le programme sur lui-même de façon à ce que les calculs qu'il met en œuvre soient réactualisés en permanence. Nous nous contentons ici de donner les principes étant entendu que dans la pratique des procédures spécifiques à l'informatique industrielle (attente d'événements, interruptions...) peuvent être mises en œuvre pour optimiser certains critères et souvent en particulier pour minimiser les temps de calcul.

I.2. Quelques confusions et erreurs possibles

Ce nécessaire bouclage du programme chargé du contrôle d'un automatisme peut conduire à un certain nombre de confusions et d'erreurs dont les principales sont résumées ci-après :

✓ **Boucles internes**

La nécessité d'une scrutation permanente des entrées pour être réceptif aux changements d'état des entrées exclut formellement toute boucle arrière dans le programme de réalisation. Il est en effet évident que de telles boucles peuvent "piéger" le processeur pendant des temps incompatibles avec la nécessité de réceptivité et réactivité. Dans les approches sommaires du GRAFCET et en particulier lorsqu'il est fait référence à des structures basiques en informatique, les risques de boucle interne naissent essentiellement de deux types de raisonnements :

- ***L'assimilation de la notion de réceptivité du GRAFCET à celle de tests des organigrammes (Figure 2)***

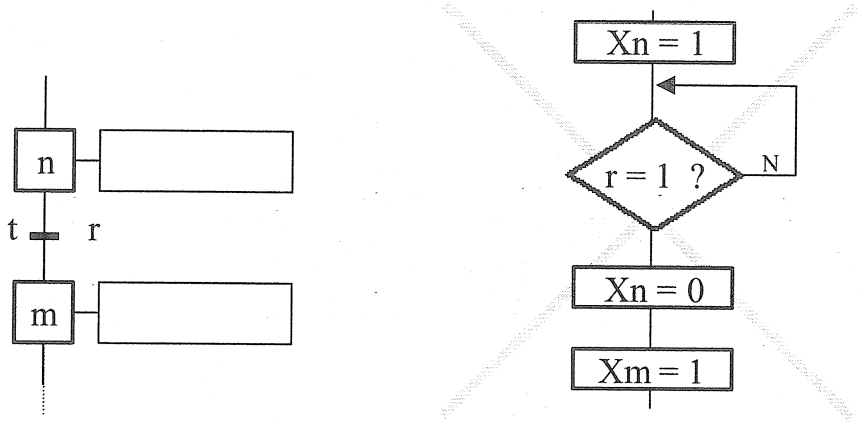


Fig.2 : Confusion possible entre GRAFCET et organigramme

- ***La conception de temporisations aux moyens de boucles d'attente (figure 3)***

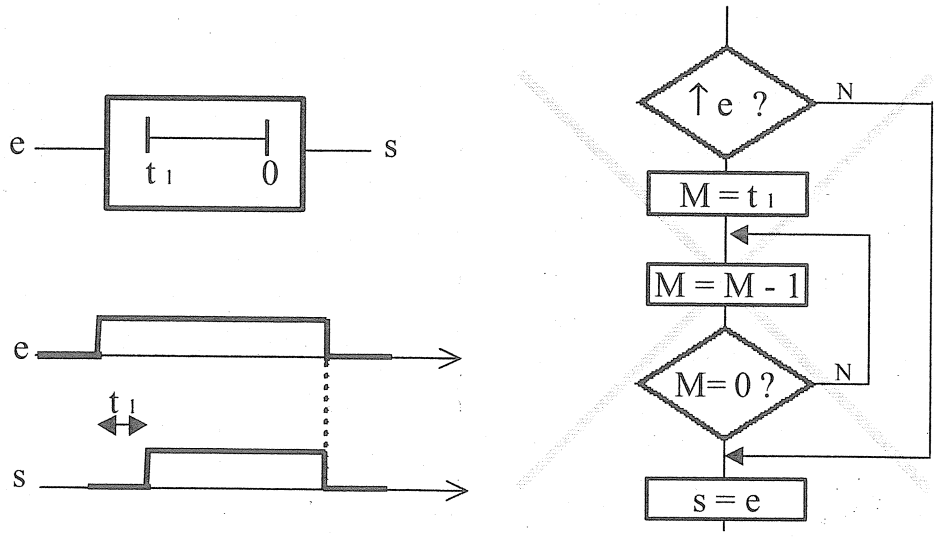


Fig. 3 : Programmation d'une temporisation incompatible avec le GRAFCET

✓ **GRAFCETs connexes et boucles de programmes**

Il n'est pas rare non plus de voir une confusion entre la boucle de retour d'un GRAFCET et le bouclage nécessaire du programme de mise en œuvre

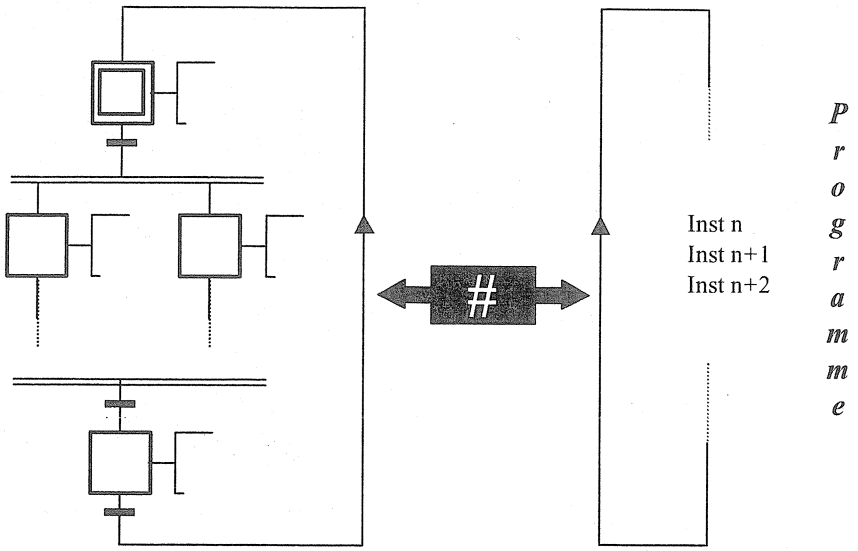


Fig. 4 : Confusion entre bouclage d'un GRAFCET et bouclage d'un programme

Les problèmes évoqués ci-avant relèvent plus du domaine de la compréhension des principes que de la mise en œuvre proprement dite. Cela signifie que dès lors qu'ils sont posés correctement les solutions sont évidentes à partir des techniques fondamentales de la microinformatique. En terme d'enseignement il semble cependant essentiel de les mettre en évidence car ils risquent de passer totalement inaperçus dans la mesure où tous les automates mettent évidemment en œuvre des solutions adéquates.

II. REGLES D'EVOLUTION ET MISE EN OEUVRE

Le GRAFCET, modèle comportemental de la partie commande d'automatismes logiques est régi par cinq règles [3] :

- ✓ ***Initialisation.***
- ✓ ***Franchissement d'une transition.***
- ✓ ***Evolution de la situation.***
- ✓ ***Transitions simultanément franchissables.***
- ✓ ***Activation et désactivation simultanées***

Les trois premières de ces règles sont à la base même du concept GRAFCET. Il ne fait donc guère de doute que leur mise en œuvre soit correctement assurée par une très grande majorité des automates programmables actuels. Il n'en va pas toujours de même pour les deux dernières. Même si leur occurrence effective dans les automatismes s'avère relativement exceptionnelle, il n'en demeure pas moins que les structures correspondantes sont parfaitement dans les normes GRAFCET et qu'elles devraient donc être supportées par les outils destinés à leur mise en œuvre.

A titre d'exemple nous allons examiner ci-après quelques problèmes liés à ces deux dernières règles :

II.1. Règle n°4 et parallélisme interprété

La règle n° 4 du GRAFCET stipule que *"plusieurs transitions simultanément franchissables sont simultanément franchies"*. Sa mise en œuvre programmée exige une structuration en deux temps :

✓ *Test des conditions de franchissement*

Pour chacune des transitions et ce, dans un ordre quelconque puisque leur numérotation est arbitraire, il faut réaliser un test de validation (activation de toutes les étapes précédentes) et un test de la réceptivité associée (proposition logique)

✓ *Franchissement effectif des transitions franchissables*

Activation et désactivation des étapes en fonction des résultats des tests précédents et en conformité avec les règles 2 et 3 d'évolution du GRAFCET

La mise en œuvre de cette règle paraît simple. Il n'est pourtant pas sûr que tous les automates programmables en assurent une mise en œuvre effective et efficace. Pour avoir une évaluation correcte du comportement de tel ou tel matériel par rapport à ce problème nous suggérons de lui soumettre le GRAFCET élémentaire de la figure 5 qui permet en même temps de vérifier une des options proposées par la norme GRAFCET qui est de prendre en compte dans les expressions booléennes des réceptivités les variables logiques (%Xi) représentatives de l'état de certaines étapes du GRAFCET.

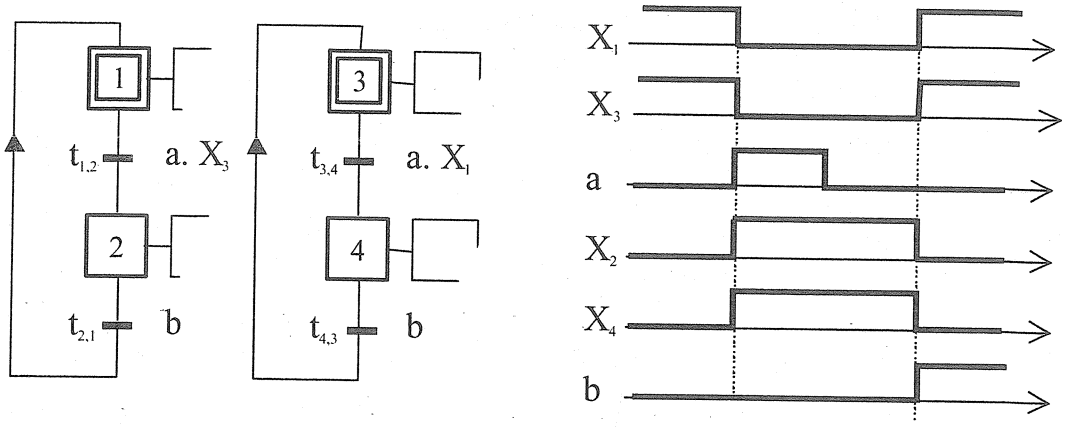


Fig. 5 : Configuration test pour évaluer le comportement d'un automate vis à vis de la règle n°4

Selon la norme, le GRAFCET ci-avant soumis à la séquence a puis b doit conduire au chronogramme donné. Si tel n'est pas le cas, c'est que l'automate ne repose pas sur une hypothèse synchrone. Le GRAFCET peut encore être utilisé mais au pris d'une solution structurelle (figure 6) :

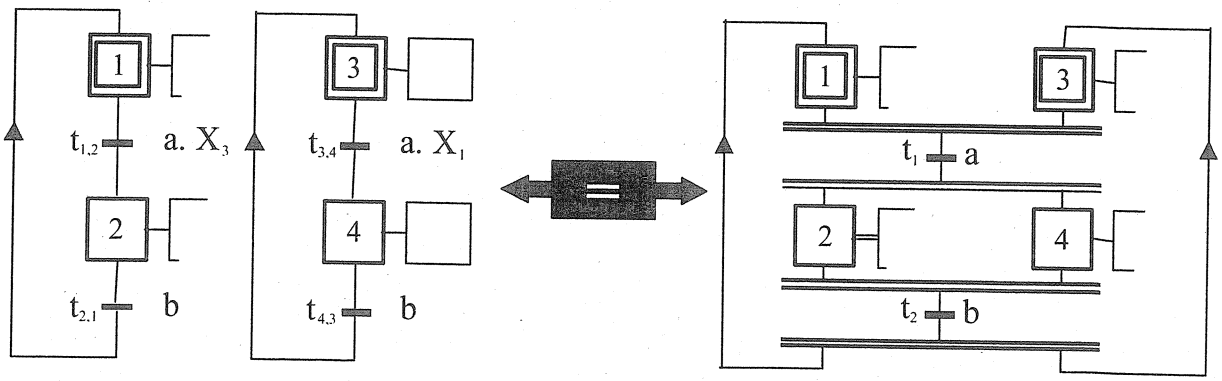


Fig. 6 : Solution structurelle à d'éventuels problèmes de franchissements synchrones

A contrario, si l'automate respecte scrupuleusement la règle de franchissement simultané, tous les problèmes ne sont pas résolus pour autant. Il reste en effet le risque de parallélisme interprété. Le GRAFCET de la figure 7 illustre ce cas quelque peu délicat. Mais attention, il ne s'agit pas ici non plus d'un problème de mise en œuvre. La difficulté est du domaine de la modélisation et du cahier des charges. La solution passe le plus souvent par l'utilisation de réceptivités exclusives.

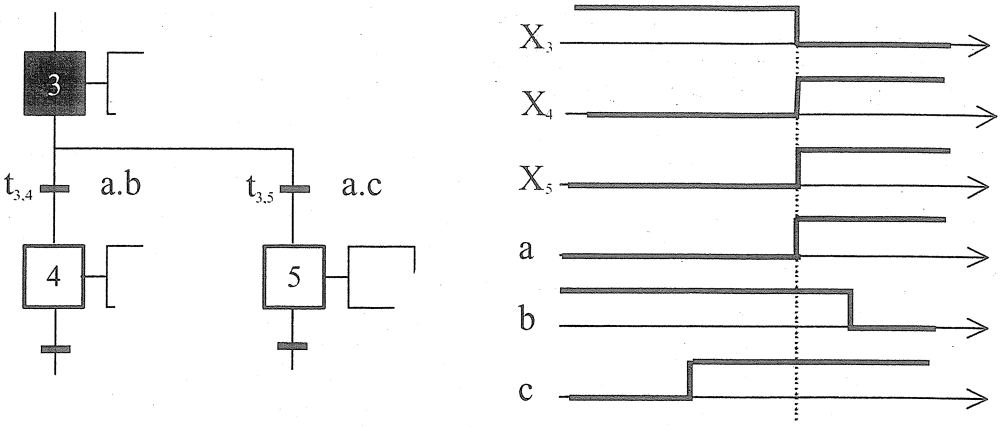


Fig. 7 : Illustration du risque de parallélisme interprété

II.2. Règle n°5 et calcul des actions

La règle n°5 (si *au cours d'une évolution une étape doit à la fois être activée et désactivée, elle reste active*) mérite-t-elle aussi une attention toute particulière. Elle exige que la mise en œuvre du franchissement évoquée ci-avant soit traitée de façon à donner la priorité à l'activation. C'est à dire que les conséquences de tous les franchissements soient calculées et traitées avant que les variables d'activation des étapes ne soient effectivement actualisées.

Pour tester le comportement d'un automate vis à vis de cette règle il peut être utile de lui soumettre la structure GRAFCET ci-après :

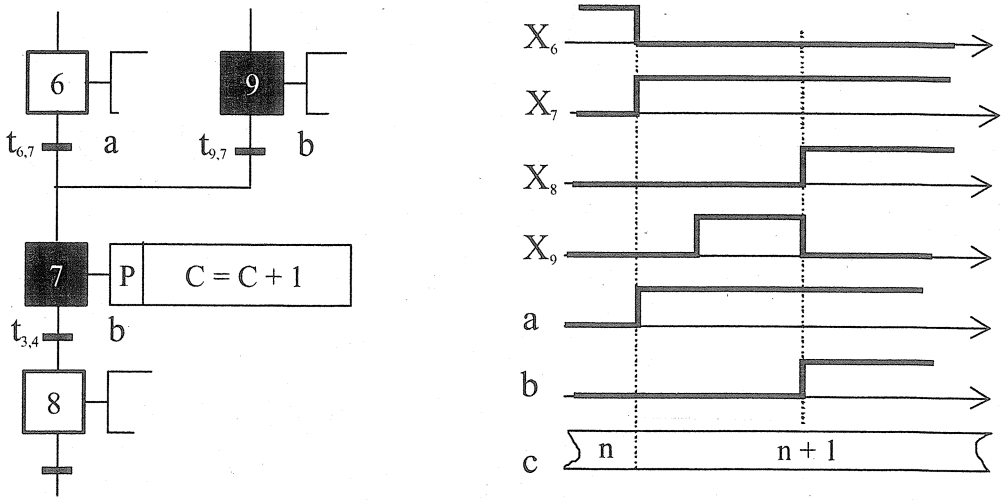
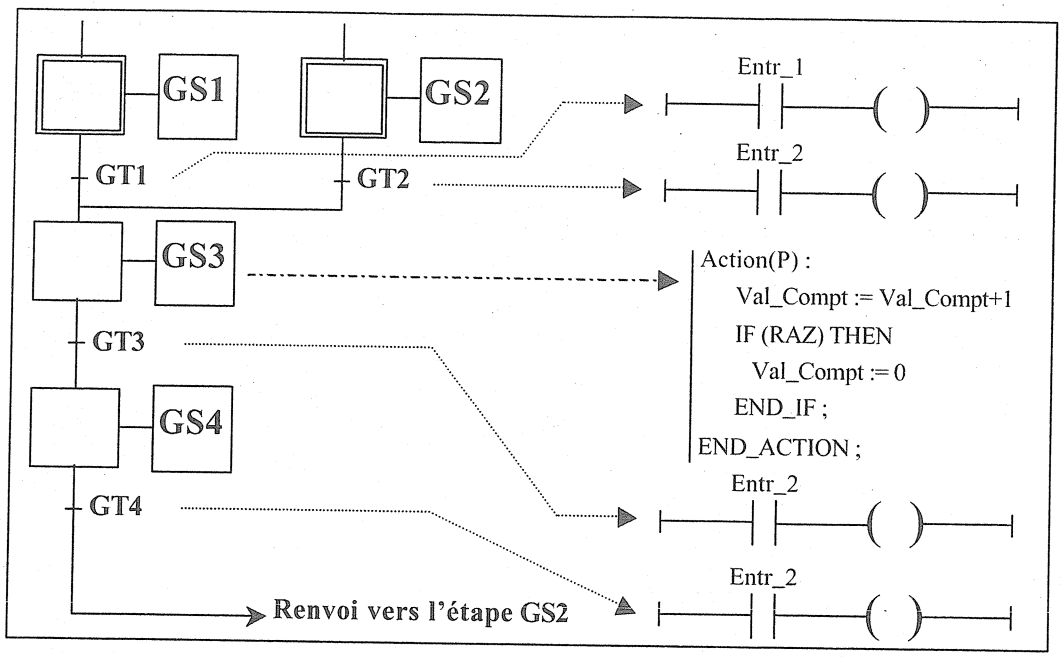


Fig. 8 : Test de comportement vis à vis de la règle n°5

L'association d'une action de type impulsionnel (P) à l'étape n°7 est nécessaire pour s'assurer qu'au cours de l'évolution, l'étape n°7 n'est pas désactivée de façon fugitive puis réactivée. Si tel était le cas, l'examen des situations de façon externe ne permettrait de visualiser que la situation de départ $\{X_7, X_9\}$ et la situation d'arrivée $\{X_7, X_8\}$ laissant croire au respect de la règle n°5 alors qu'il n'en est rien.

Un exemple concret de test est donné ci-après. Il s'agit d'un GRAFCET implanté sur un automate LT100 (Leroy Automation) développé et mis au point au moyen d'un atelier logiciel ISaGRAF (CJ International). Après initialisation, l'entrée Entr_1 permet d'activer l'étape GS2. Lorsque l'entrée Entr_2 passe à 1 les deux transitions GT2 et GT3 sont franchissables. Conformément à la règle n°5 l'étape GS3 reste active et l'action Val_Compt qui lui est associée n'est pas incrémentée



Exemple 1 : Test de comportement vis à vis de la règle 5 (Automate LT100, Atelier ISaGRAF[1])

III. INTERPRETATIONS DU GRAFCET

L'objectif du GRAFCET est de déterminer le comportement (séquence de sortie) d'une partie commande d'automatismes soumise à un stimulus (séquence d'entrée). Le caractère normé du GRAFCET pourrait laisser croire que la solution est forcément unique. En fait plusieurs interprétations sont possibles suivant les hypothèses retenues. Les mises en œuvre pratiques du GRAFCET sont étroitement dépendantes de ces interprétations et peuvent donc conduire à des comportements sensiblement différents suivant les options choisies [6].

Deux grandes familles d'interprétations sont généralement proposées. Nous allons les rappeler ci-après et nous donnerons ensuite des tests simples pour évaluer le comportement des automates programmables par rapport aux options choisies qui ne sont pas toujours explicites.

III.1. Interprétation sans recherche de stabilité (S.R.S.)

C'est évidemment la plus simple car elle se contente de l'algorithme[2] ci-après (figure 9):

Cette interprétation attrayante par sa simplicité peut cependant conduire à des difficultés [7]. Comme illustration, le cas de la figure 10 où, partant de la situation $\{X_1, X_4\}$, la séquence [a puis b] peut conduire soit à la situation $\{X_3\}$soit à la situation $\{X_3, X_5\}$suivant l'instant où se situe le changement de la variable b par rapport aux différentes lectures des entrées. Nous avons donc un comportement qui n'est pas déterministe par rapport à la séquence d'entrée. Le problème vient du fait que toutes les conséquences d'un événement d'entrée ne sont pas calculées avant qu'un autre événement ne soit pris en compte.

[1] Pour des raisons de taille, le dossier établi par le logiciel de conception n'a pas été utilisé in extenso, le graphisme a été compacté

[2] Par soucis de clarté de l'exposé les algorithmes évoqués dans cet article ne prennent pas en compte les problèmes d'initialisation, de forçages ou d'actions conditionnelles

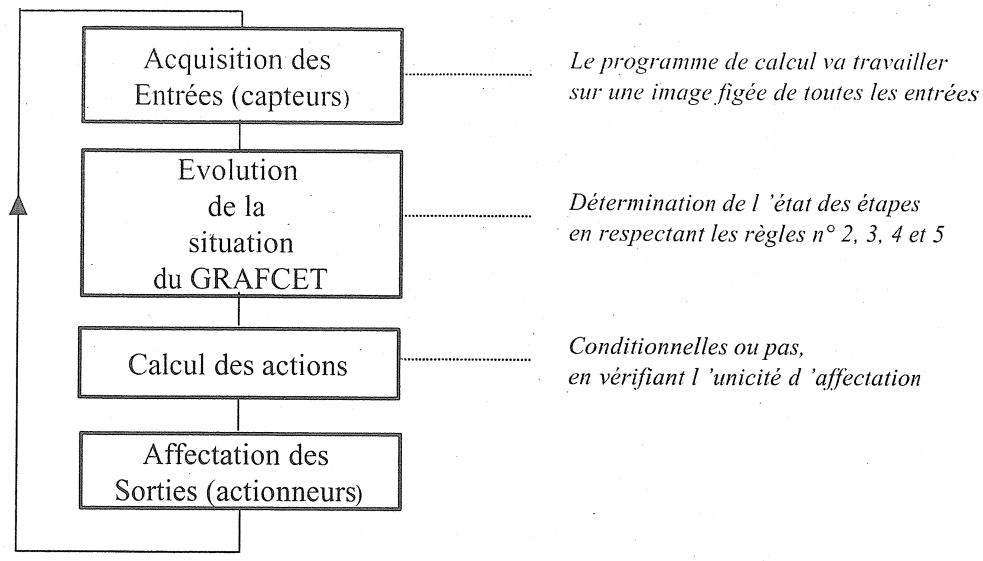


Fig. 9 : Algorithme d'interprétation GRAFCET sans recherche de stabilité(srs)

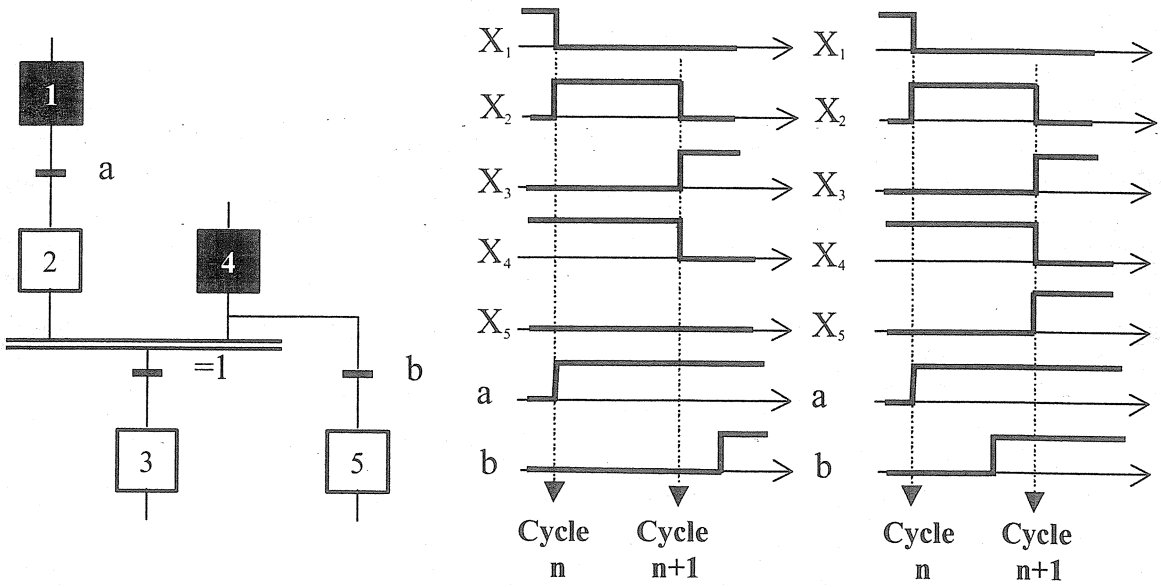


Fig.10 : Exemple de difficultés dues à un algorithme d'interprétation SRS

III.2. Interprétation avec recherche de stabilité (A.R.S.)

Cette approche de type synchrone (réactivité, causalité à temps nul ...) consiste justement à ne prendre en compte de nouveaux événements que lorsque le GRAFCET a atteint une situation stable. Cette solution proche des hypothèses de mode fondamental des machines séquentielles exige pourtant quelques précautions notamment vis à vis des sorties. Ainsi, les postulats temporels du modèle GRAFCET stipulent que "*la durée d'activité d'une étape peut être aussi petite que l'on veut mais non nulle*". Ce postulat semble à priori contradictoire avec la notion de synchronisme évoquée ci-avant. Il doit être interprété comme le fait que, à l'échelle de temps interne, l'activité fugitive d'une étape est parfaitement détectable et que les actions associées de type impulsionnel doivent s'en trouver affectées. Par contre les actions de type continu n'ont aucun sens si leur durée d'activité est extrêmement faible par rapport aux temps mis en jeux dans la partie opérative.

Cela conduit donc à l'algorithme de la figure 11 ci-après.

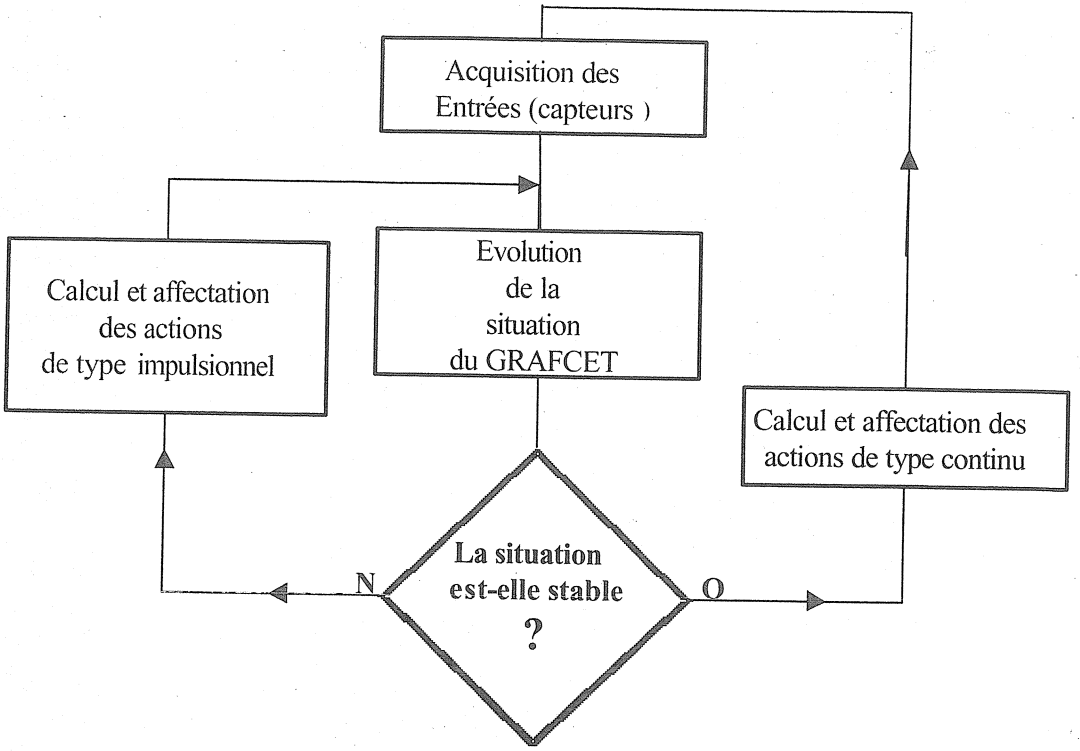


Fig. 11 : Algorithme d'interprétation GRAFCET avec recherche de stabilité(SRS)

Cette interprétation ARS du GRAFCET semble tout à fait cohérente avec les approches synchrones des systèmes temps réel complexes (SIGNAL, LUSTRE, ESTEREL...) . Elle peut pourtant poser elle aussi des problèmes dont les principaux sont :

✓ **Le test de stabilité de la situation.**

Il existe en effet deux façons de déterminer si la situation est stable:

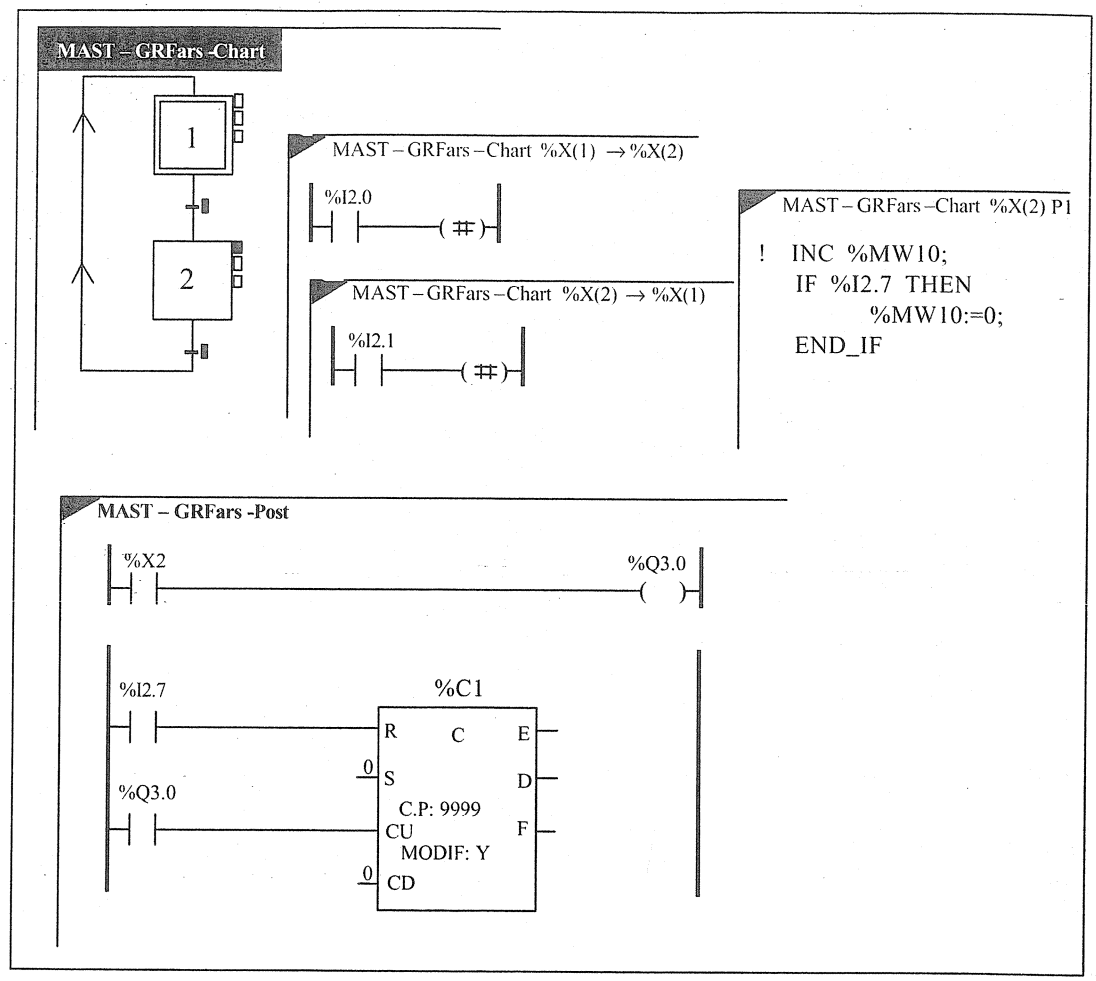
- soit il n'y a plus de transitions franchissables,
- soit deux situations successives (cycles de calcul) sont identiques.

Or ces deux test ne sont pas rigoureusement équivalents [8].

✓ **L'apparition de boucles infinies**

Dans ce cas, aucune situation stable n'étant atteinte, le contrôle du processus n'est plus effectif puisque les entrées ne sont plus scrutées.

Pour résumer les problèmes évoqués ci-avant nous proposons le GRAFCET de l'exemple 2. Si les entrées se succèdent suivant la séquence %I2.0 puis %I2.1, le GRAFCET prend successivement la situation {%X1} puis {%X2}, le mot interne %MW10 ainsi que le compteur %C1 sont incrémentés à chaque activation de l'étape 2. Par contre, si l'entrée %I2.1 est à 1 avant que l'entrée %I2.0 ne passe elle-même à 1, nous sommes conduits à un cycle instable. Est-ce à dire que l'automate testé met en œuvre une recherche de stabilité du GRAFCET ? Pas du tout ! En effet, le passage à 0 de l'une des variables d'entrée permet de stopper les oscillations preuve que les entrées sont scrutées à chaque cycle et non pas après confirmation d'une situation stable du GRAFCET. Par ailleurs, l'action de type continues insérée dans le traitement postérieur du GRAFCET (%Q3.0 = %X2) est calculée à chaque cycle car le compteur %C1 est incrémenté alors qu'un algorithme ARS ne l'affecterait pas tant que la situation évolue.



Exemple 2 : Test de comportement vis à vis de l'interprétation (Automate Premium, Atelier PL7Pro)

Conclusion

Le GRAFCET s'impose aujourd'hui comme un des outils importants de l'automaticien. Il s'avère en effet particulièrement efficace pour modéliser la patrie commande des automatismes logiques. De plus, de nombreux outils ont été développés pour en assurer la saisie, l'analyse et la mise en œuvre proprement dite (ateliers logiciels, automates programmables industriels…).

Mais la normalisation et l'efficacité des outils associés n'est pas sans danger. Elles peuvent en effet conduire à une sorte de banalisation et peuvent laisser croire que des efforts de compréhension et de validation ne sont plus nécessaires. Ceci s'avère très sensible lorsqu'il s'agit d'enseignement et donc de pédagogie. L'expérience montre en effet que des amalgames peuvent notamment se créer avec des concepts ou des méthodes propres à l'informatique. Il est donc encore aujourd'hui fondamental de poser les principes du GRAFCET, de bien illustrer ses règles de syntaxes et de comportement. Par ailleurs, les outils qu'ils soient matériels ou logiciels associés au GRAFCET cherchent avant tout à rendre les procédures d'automatisation les plus efficaces possibles. Pour ce faire des options ont souvent été prises. Mais cela n'est pas forcément explicite et peut conduire à des difficultés si des méthodes d'évaluation adéquates ne sont pas mises en œuvre.

En fait pour conclure et résumer nous pourrions reprendre le titre de l'article de D.Gendreau :

" L'enseignement du GRAFCET : pas si facile ! "

BIBLIOGRAPHIE

[1] M.BLANCHARD – Comprendre, maîtriser et appliquer le GRAFCET – Cepadues Editions

[2] D.GENDREAU – L'enseignement du GRAFCET : pas si facile ! – Revue Technologies – n° 94, p 11/18 - 1998

[3] NFC-82. Norme NF C 03-190 : Schémas, diagrammes, tableaux: Diagramme fonctionnel GRAFCET pour la description des systèmes logiques de commande, 82.

[4] IEC International Electronical Commission. IEC 848-Preparation of function charts for control systems, 90.

[5] IEC International Electronical Commission. IEC 1131-Standard for programmables controllers part : programming languages, 93

[6] GREPA - LE GRAFCET: de nouveaux concepts - Cepadues Editions

[7] R. DAVID, H. ALLA - Du GRAFCET aux réseaux de Petri - Editions HERMES

[8] P. LE PARC - Apports de la méthodologie synchrone pour la définition et l'utilisation du GRAFCET-Thèse de l'Université de Rennes 1, 94

.....

Serveur thématique GRAFCET: http://japura.lurpa.ens-cachan.fr/grafcet/

MACHINES D'ETATS ET GRAFCET

Jean Louis BIANCHI
Lycée Jules Ferry - VERSAILLES

Résumé : Après avoir présenté les structures de machines d'états et évoqué leur champ d'application, l'article présente un exemple simple d'application à la commande d'un moteur pas à pas.

Dans une deuxième partie, les rapports avec le GRAFCET sont abordés, et notamment le passage du GRAFCET vers un graphe d'états représentant une machine et ses possibilités d'implantation dans les composants de type ©PLD avec l'utilisation d'outils de synthèse logique (VHDL).

1 Domaine d'étude des machines d'états:

On utilise la représentation machine d'états lorsqu'un système séquentiel se caractérise par un nombre fini d'états, et par le fait qu'il se trouve à tout instant dans un et un seul état.

Les changements d'états sont provoqués par un événement extérieur faisant partie d'une liste d'événements possibles également finie.

Un tel système est dit déterministe, il est entièrement représentable et défini par :

- La liste finie des états représentatifs du système : $\{S_1, S_2, S_3, \ldots, S_m\}$.
- La liste finie des événements qui peuvent induire des changements d'états : $\{E_1, E_2, E_3, \ldots, E_m\}$.
- La liste finie des règles de transition d'état déclenchée par les événements qui se caractérise par : un événement, un état de départ, et un état de destination.
- Par ailleurs, un tel système se doit de provoquer des actions sur le monde extérieur. Ces actions sont associées aux changements d'états et peuvent prendre en compte les conditions d'entrées (événements extérieurs) selon le type de machine.

Il s'agit d'un outil simple et puissant de description connue outre Atlantique sous le sigle F.S.M. (Finite State Machine) encore qualifié de "représentation Automate".

Cette représentation concerne des domaines aussi variés que l'Electronique, les Automatismes, et de plus en plus le développement de logiciels embarqués. Les principaux intérêts outre l'objectif de performance, sont la fiabilité, la maintenance et la portabilité.

De nombreuses applications sont, au moins pour leur logique de décision, assimilables à des machines d'états. Citons, pour notre intérêt particulier ici, les contrôles et automatismes séquentiels mais également les gestions d'interfaces, les systèmes d'acquisition, etc. .

2 Les supports logiciels :

L'approche "machine d'états" se généralisant, de nombreux outils logiciels intègrent ce type de description à des niveaux de complexité variables. Certains génèrent à partir d'un outil graphique de description (Graphes d'états) du code C qui, lorsqu'il s'exécute respecte l'automate conçu; c'est le cas des logiciels tels que VISUALSTATE qui peuvent appréhender des systèmes très complexes.

Plus spécifiquement, certains outils permettent la description Machine d'états par représentation graphique en vue de la programmation des composants logiques programmables tels que PLD, CPLD ou

FPGA. On peut ainsi décrire une application automate par un graphe d'états; le logiciel génère après compilation du code VHDL permettant la simulation, puis l'inscription dans le composant cible.

Ces logiciels tels que WARP2 sont peu coûteux (environ 500F) et permettent une approche structurée des problèmes d'automatismes séquentiels ou autres, tout en assurant une lisibilité et une portabilité garantie par la syntaxe de haut niveau de type VHDL.

3 Classification des machines d'états :

3.1 Machines d'états asynchrones et synchrones:

On distingue deux types généraux de machines d'états selon la classification bien connue ; celles dont les états évoluent au rythme d'une horloge qui sont dites synchrones, et celles asynchrones, dont les états évoluent selon les combinaisons des entrées.

Par principe les machines asynchrones dont le temps de réaction est déterminé par la logique combinatoire d'acquisition des entrées, sont plus rapides ; les machines synchrones ont un temps de réponse variable qui peut aller jusqu'à une période d'horloge et parfois plus. (Mais on peut toujours augmenter la fréquence d'horloge…)

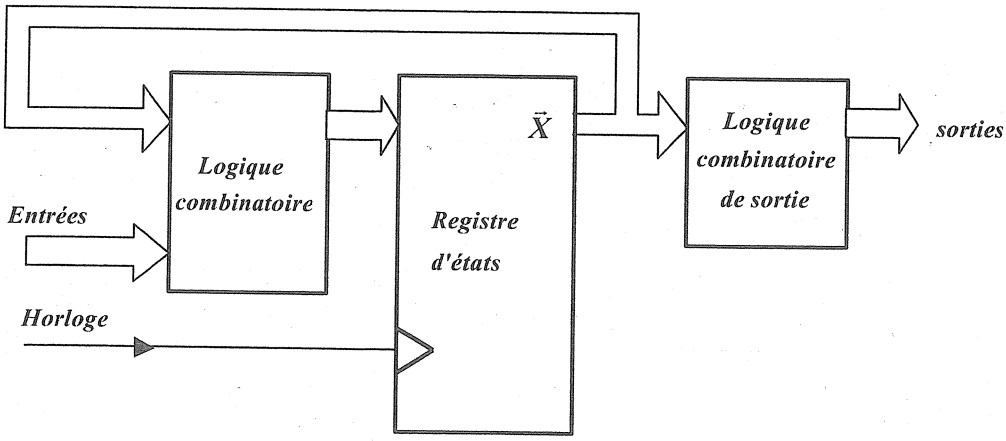
Malgré cela, les machines d'états synchrones sont plus utilisées car à l'inverse des machines asynchrones, elles garantissent une durée minimale à chaque état évitant des états instables ou de durée insuffisante.

Nous n'envisagerons dans la suite de cet article que les seules structures synchrones.

3.2 Machines de MOORE et de MEALY:

On distingue deux grandes catégories de machines d'états selon que les sorties de la machine sont générées à partir du seul vecteur d'état, ou à partir du vecteur d'état et de l'état logique des entrées :

- **La machine de MOORE:**

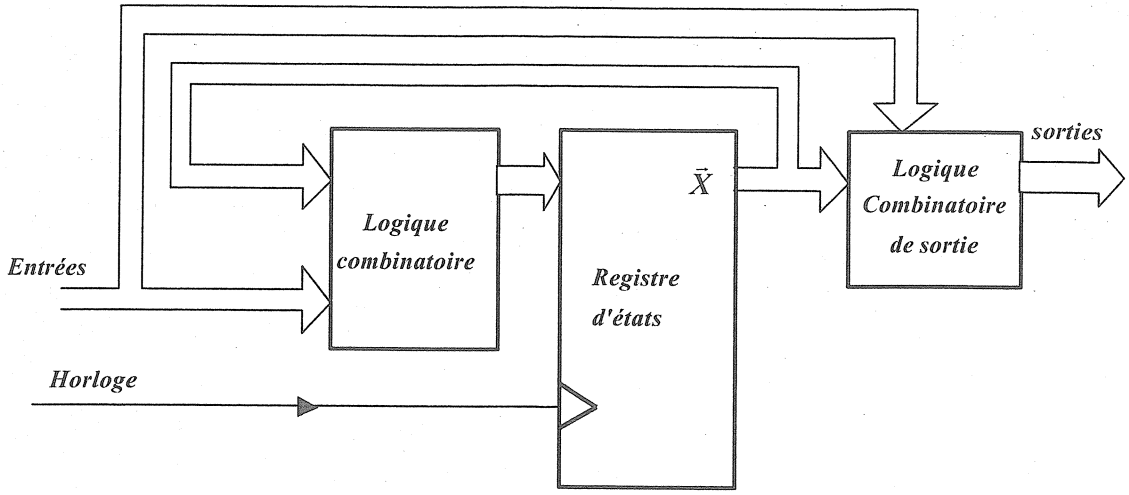


La machine de Moore est constituée d'un registre d'états qui gère l'état courant du système en fournissant un ***vecteur d'état*** $\vec{X}$ formé de ***n bits d'états*** dont le rôle est en quelque sorte de mémoriser la trajectoire au cours du temps.

L'évolution du vecteur d'état dépend de son état antécédent et des conditions d'entrée.

L'évolution des sorties ne dépend que de l'état courant du vecteur d'état.

- la machine de MEALY:



A la différence de la machine de MOORE, ***la machine de MEALY génère des sorties qui dépendent non seulement de l'état courant mais également de l'état des entrées.***

On remarquera que le temps de réaction entre entrée et sortie d'une machine de MOORE est entièrement déterminé par la période de l'horloge, alors que celui d'une machine de MEALY peut être plus court puisque les entrées peuvent modifier les sorties de manière combinatoire. C'est une donnée qui peut plaider en faveur de ce type de machine lorsque l'impératif est celui de la rapidité de réaction aux sollicitations d'entrées.

4 Exemple d'application simple :

On se propose de réaliser le séquenceur de commande d'un moteur pas à pas à aimants permanents et à deux phases alimentées en mode bipolaire; soit pour le schéma de principe électromagnétique:

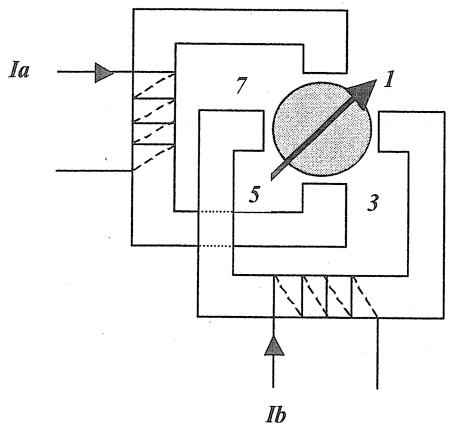
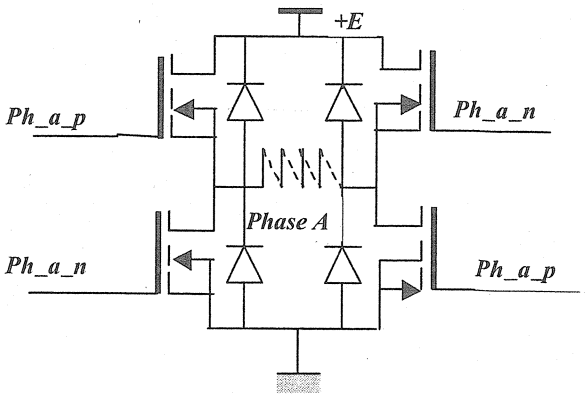
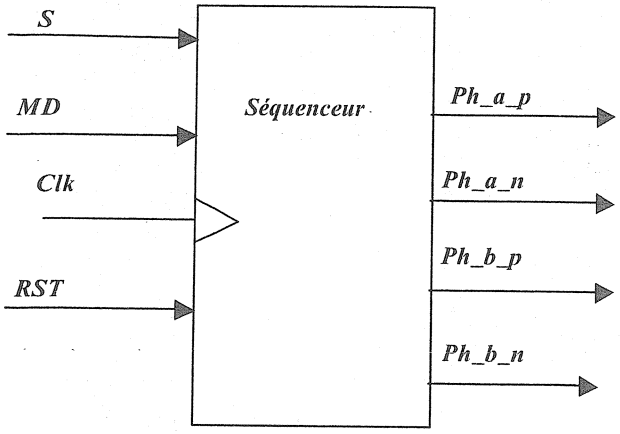


Schéma de principe de commande du moteur



Alimentation par hacheur de la phase A

On souhaite obtenir deux sens de rotation et deux modes de contrôle, soit : en pas entiers avec les deux phases alimentées sur chaque pas (1,3,5,7), soit en demi-pas avec une seule phase alimentée sur les pas pairs (2,4,6,8) et deux phases alimentées sur les pas impairs (1,3,5,7). Soit:



- La machine d'états possède 3 entrées:
  - ***MD*** Choix du mode de fonctionnement:
    - MD=0 mode pas entiers.
    - MD=1 mode demi-pas.
  - ***S*** Choix du sens de rotation:
    - S=0 sens horaire.
    - S=1 sens anti-horaire.
  - ***RST*** Entrée d'initialisation:
    - Active à l'état haut sur le pas 1
- La machine d'états possède 4 sorties:
  - ***Ph_a_p*** Commande du hacheur phase A:
    - Ia>0 si Ph_ap=1.
  - ***Ph_b_p*** Commande du hacheur phase B:
    - Ib>0 si Ph_bp=1.
  - ***Ph_a_n*** Commande du hacheur phaseA:
    - Ia<0 si Ph_an=1.
  - ***Ph_b_n*** Commande du hacheur phase B:
    - Ib<0 si Ph_bn=1.

Nous pouvons établir le tableau de fonctionnement suivant:

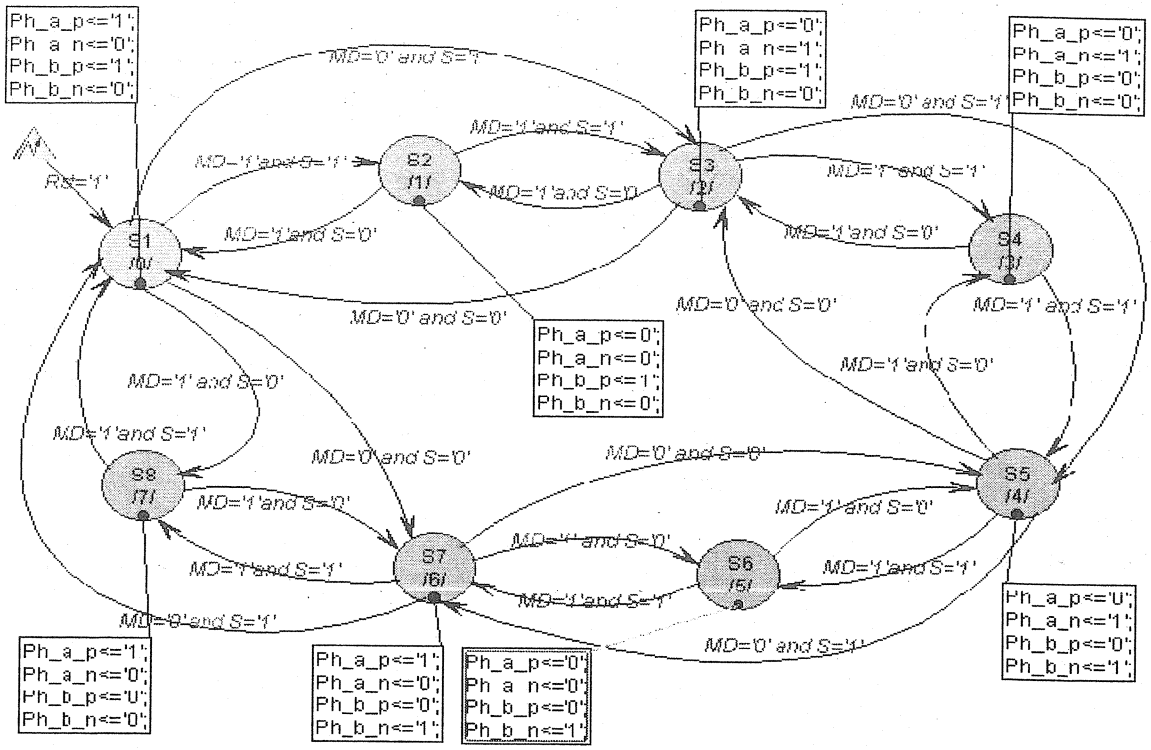
Pas	Ia phase A	Ib phase B	Sortie Ph_a_p	Sortie Ph_a_n	Sortie Ph_b_p	Sortie Ph_b_n
1	+ Ia	+ Ib	1	0	1	0
2	0	+ Ib	0	0	1	0
3	- Ia	+ Ib	0	1	1	0
4	- Ia	0	0	1	0	0
5	- Ia	- Ib	0	1	0	1
6	0	- Ib	0	0	0	1
7	+ Ia	- Ib	1	0	0	1
8	+ Ia	0	1	0	0	0

Nous pouvons résoudre ce problème par une méthode classique de recherche des équations logiques pour un séquenceur synchrone à bascule D, et faire une description bas niveau en VHDL.

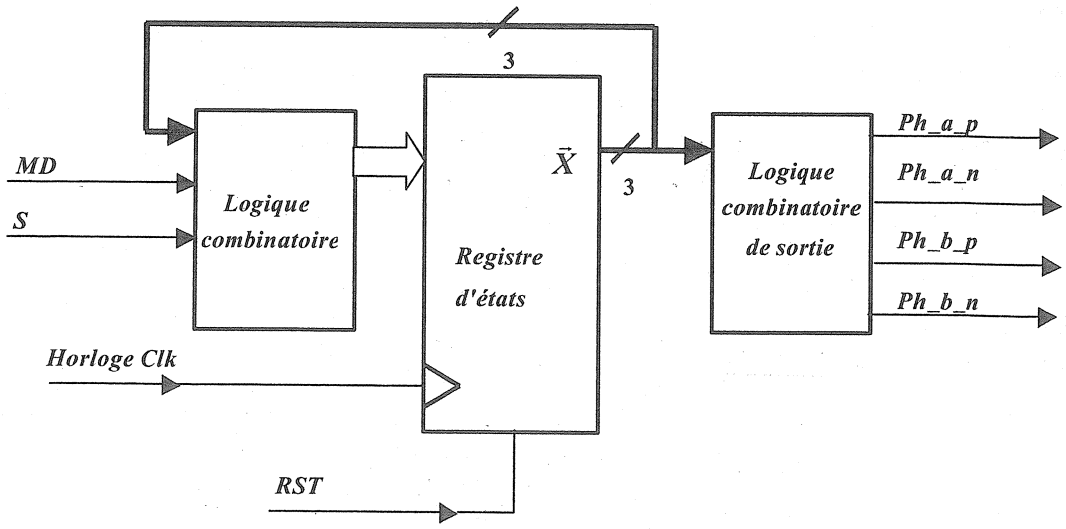
Signalons également que VHDL permet une description comportementale de haut niveau du système particulièrement lisible et qu'on peut éventuellement combiner les deux niveaux de description.

Pour les besoins de cet article, nous choisissons une description par graphe d'états et une approche "machines d'états". L'outil utilisé ici, WARP2 de CYPRESS, permet cette description par l'intermédiaire d'un éditeur de graphes d'états particulièrement facile d'utilisation.

Remarquons que la machine présente 8 états stables notés : {S1,S2,S3,S4,S5,S6,S7,S8}; nous pouvons en déduire que le vecteur d'état $\vec{X}$ sera composé d'au moins trois bits d'états. Ce que nous pouvons représenter par le graphe d'état suivant :



La structure de la machine est de type MOORE, et nous voyons que le registre d'état (ou vecteur d'état) est constitué d'états numérotés (ici une combinaison de trois bits d'états) qui représentent l'état courant, soit:



L'entrée RST a été déclarée comme RESET asynchrone, on aurait pu choisir un Reset synchrone, le choix peut être conditionné par les possibilités de la cible (PLD ou CPLD).

A partir de la description Graphe d'états, le logiciel génère un fichier VHDL directement exploitable; c'est à dire qu'après compilation, on obtient soit un fichier JEDEC directement utilisable pour la programmation d'un PLD (ici un 22V10), soit un fichier permettant la programmation sur site du composant cible le permettant (CPLD par exemple) à l'aide d'un kit fourni par CYPRESS avec le logiciel WARP2.

Voici le codage des états retenu par le logiciel:

```
State encoding (user codes) for 'sreg0' is:
        s1 :=   "000";
        s2 :=   "001";
        s3 :=   "010";
        s4 :=   "011";
        s5 :=   "100";
        s6 :=   "101";
        s7 :=   "110";
        s8 :=   "111";
```

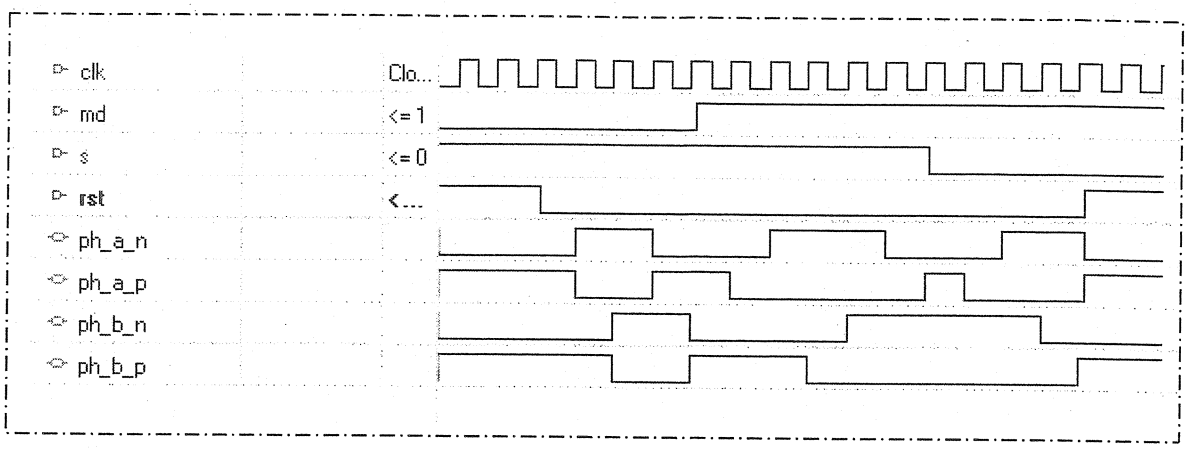
Voici les équations de sorties générées:

```
ph_a_n =
    sreg0SBV_0.Q * /sreg0SBV_1.Q * /sreg0SBV_2.Q
  + /sreg0SBV_0.Q * sreg0SBV_1.Q

ph_a_p =
    /sreg0SBV_0.Q * /sreg0SBV_1.Q * /sreg0SBV_2.Q
  + sreg0SBV_0.Q * sreg0SBV_1.Q

ph_b_n =
    sreg0SBV_0.Q * /sreg0SBV_2.Q
  + sreg0SBV_0.Q * /sreg0SBV_1.Q

ph_b_p =
    /sreg0SBV_0.Q * /sreg0SBV_2.Q
  + /sreg0SBV_0.Q * /sreg0SBV_1.Q
```

Equations des bits d'états générés:

```
sreg0SBV_0.D =
  /sreg0SBV_0.Q * sreg0SBV_1.Q * /sreg0SBV_2.Q * /md * s
  + /sreg0SBV_0.Q * sreg0SBV_1.Q * sreg0SBV_2.Q * md * s
  + sreg0SBV_0.Q * /sreg0SBV_2.Q * md * s
  + /sreg0SBV_0.Q * /sreg0SBV_1.Q * /sreg0SBV_2.Q * /s
  + sreg0SBV_0.Q * sreg0SBV_2.Q * /s
  + sreg0SBV_0.Q * sreg0SBV_2.Q * /md
  + sreg0SBV_0.Q * sreg0SBV_1.Q * /s
  + sreg0SBV_0.Q * /sreg0SBV_1.Q * s

sreg0SBV_0.AR =
    rst

sreg0SBV_0.SP =
    GND

sreg0SBV_0.C =
    clk

sreg0SBV_1.D =
  sreg0SBV_1.Q * /sreg0SBV_2.Q * md * s
  + /sreg0SBV_1.Q * sreg0SBV_2.Q * md * s
  + /sreg0SBV_1.Q * /sreg0SBV_2.Q * /s
  + sreg0SBV_1.Q * sreg0SBV_2.Q * /s
  + /sreg0SBV_1.Q * /sreg0SBV_2.Q * /md
  + sreg0SBV_1.Q * sreg0SBV_2.Q * /md
sreg0SBV_1.AR =
    rst
sreg0SBV_1.SP =
    GND
sreg0SBV_1.C =
    clk

sreg0SBV_2.D =
    sreg0SBV_2.Q * /md
  + /sreg0SBV_2.Q * md

sreg0SBV_2.AR =
    rst

sreg0SBV_2.SP =
    GND

sreg0SBV_2.C =
    clk
```

Avant programmation ou téléchargement, le logiciel permet une simulation, nous la reproduisons ici, elle permet de vérifier la conformité de la machine d'état au cahier des charges:



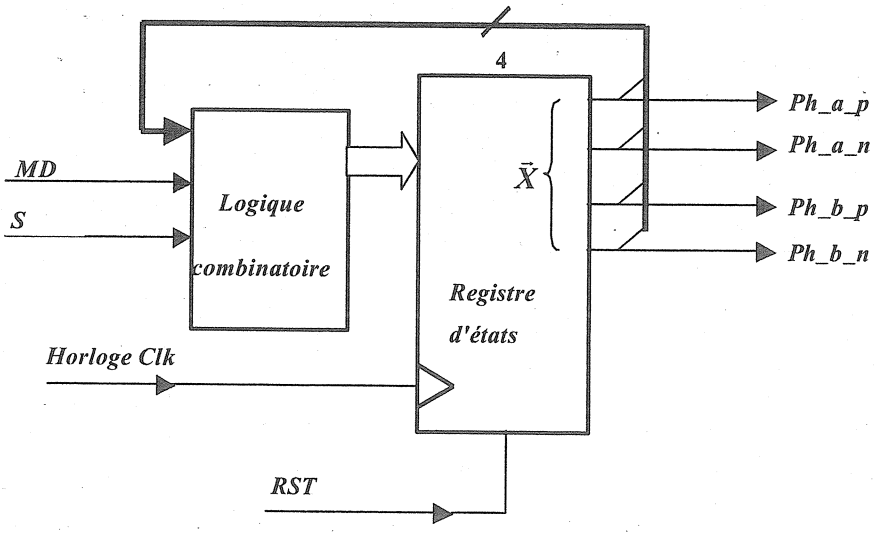
Remarque : Machine à états codés par les sorties :

Nous avons vu que le logiciel respecte la structure générale de la machine de MOORE. Le vecteur d'état est codé selon un code 3 bits qui est ici un simple code binaire naturel. La génération des sorties se fait par décodage des codes d'états et cela conditionne d'ailleurs le routage final dans le composant cible (PLD, CPLD ou FPGA).

Une autre voie est possible et optimale ici, elle consiste à construire le vecteur d'état à partir des sorties et ainsi chaque bit d'état commande une sortie.

Comme le circuit demande 4 sorties (Ph_a_p, Ph_a_n, Ph_b_p, Ph_b_n), le vecteur d'état sera constitué de 4 bits au lieu de 3 mais nous n'aurons pas besoin de la logique combinatoire de sortie. On obtiendra alors un routage différent avec un avantage essentiel qui n'est pas déterminant pour notre application, mais qui peut souvent l'être dans d'autres, c'est que les sorties seront parfaitement synchrones.

Voici donc la structure de la machine d'états codés par les sorties :



Le graphe d'états ne change pas, si ce n'est l'attribution nouvelle du vecteur d'état qui est acceptée par le logiciel. On obtient les mêmes résultats au niveau fonctionnel.

5 Machines d'états et GRAFCET :

5.1 Mise en situation:

Par définition, nous dirons que le GRAFCET représente une machine d'états. C'est un outil de description puissant et il ne s'agit pas ici de mettre les deux modèles en concurrence.

Il nous paraît plus intéressant, dans les limites possibles de cet article, de relever les rapports qu'entretiennent les deux types de description et d'essayer d'associer l'une à l'autre.

Le GRAFCET est trop souvent associé à la "mono technologie" de l'automate programmable industriel ; c'est regrettable car ses règles sont très précises et en font un outil de description et d'analyse très intéressant qui permet d'aborder de nombreux problèmes séquentiels avec rigueur et méthode.

De nombreux problèmes d'automatismes logiques mais également d'interfaces, de systèmes d'acquisition peuvent être abordés à l'aide d'un GRAFCET sans pour autant faire l'objet d'une réalisation par automate programmable industriel, soit parce que la complexité ne l'impose pas, soit pour d'autres impératifs plus matériels.

On peut ainsi implanter un GRAFCET dans un composant logique programmable de type PLD (CPLD ou FPGA) ; dans ce cas, il sera nécessaire de traduire le GRAFCET par un Graphe d'états afin de générer comme il a été montré plus haut un fichier VHDL, permettant la simulation de la machine obtenue et son inscription dans une cible.

5.2 Passage du GRAFCET vers un graphe d'états :

En fait, si le passage du graphe d'états vers le GRAFCET est toujours possible, il n'en est pas de même de la traduction GRAFCET vers graphe d'états. Il ne faut pas oublier que par définition un automate (une machine d'états) ne peut être que dans un et un seul état à un instant donné ; ce qui à priori exclut la traduction en graphe d'états de tout GRAFCET présentant un parallélisme.

Effectivement c'est un des avantages du GRAFCET sur les modèles d'états que de pouvoir représenter le parallélisme ; dans ce cas, plusieurs étapes sont actives mais le problème peut être contourné de plusieurs façons:

- On peut mettre en œuvre une machine d'états principale pouvant se scinder en plusieurs sous-machines fonctionnant en parallèle puis se regroupant.
- On peut remarquer que plusieurs étapes actives ne le sont pas forcément strictement en parallèle.

5.3 Exemple de GRAFCET linéaire synchrone sans parallélisme:

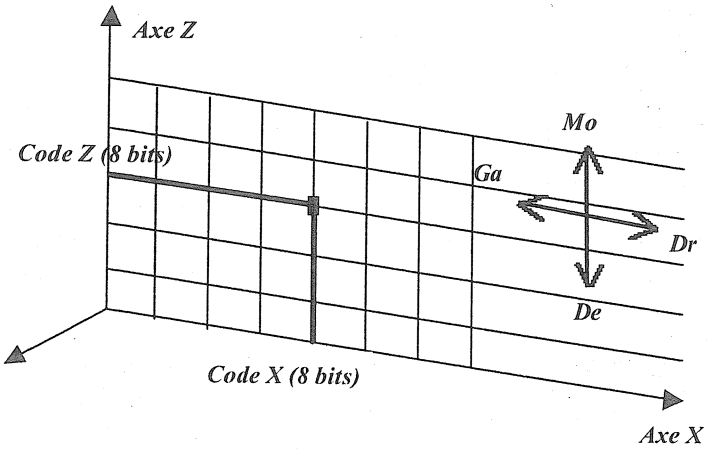
Supposons qu'on veuille réaliser l'automatisme de contrôle d'un trans-gerbeur composé de Nz par Nx places ; soit, Nx colonnes selon l'axe X et Nz lignes selon l'axe Z.

La position de chaque place est repérée par deux codeurs incrémentaux ; le nombre de points à compter selon chaque axe est fixé par deux codes d'entrées représentés par deux mots 8 bits:

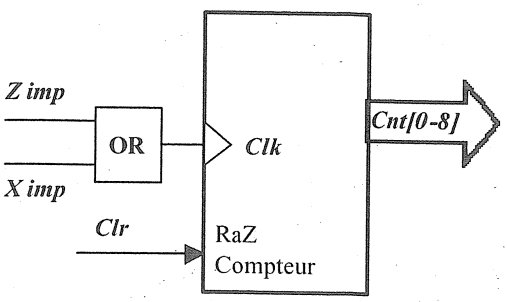
- Code Z, représente la ligne à atteindre selon l'axe Z exprimée en impulsions à compter.
- Code X, représente la colonne à atteindre selon l'axe X exprimée en impulsions à compter.

On initialise un cycle de recherche à partir d'une variable dcy (départ cycle, active à l'état logique 1). Arrivé à la position (Z, X) recherchée, une variable F (active à l'état logique1) donne l'ordre de retour en position initiale ,soit Z=X=0.

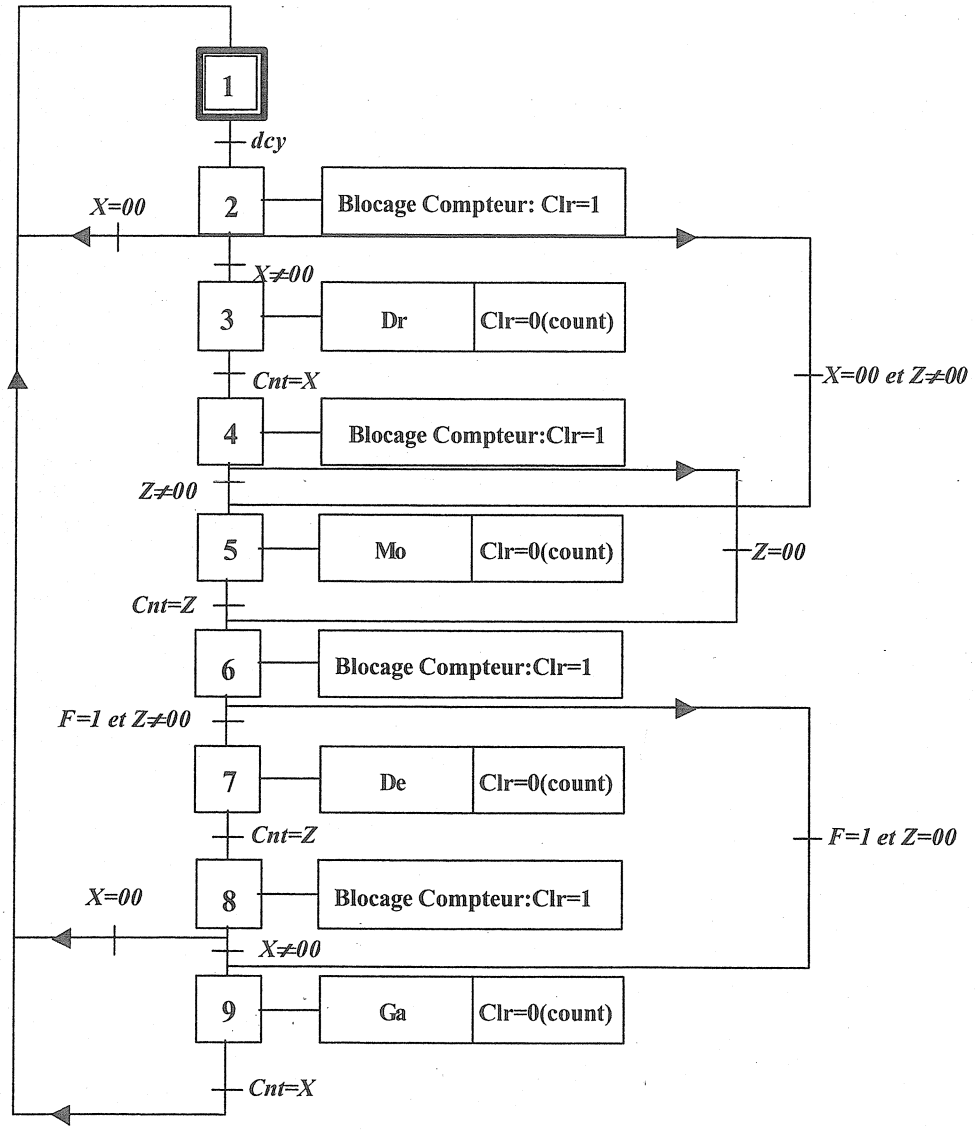
La partie opérative permettant les déplacements du chariot est constituée de 2 actionneurs électriques permettant les mouvements Dr(droite), Ga(Gauche), Mo(Montée), De(Descente). On exclut le fonctionnement simultané de deux actionneurs.



Structure fonctionnelle retenue pour le compteur :



Le GRAFCET suivant décrit le fonctionnement souhaité, notons qu'il s'agit d'un modèle synchrone c'est à dire que les transitions sont conditionnées par une horloge de fréquence élevée par rapport aux évolutions du système et notée CLK_s (non représentée), un signal RST permet l'initialisation sur l'étape 1.



On constate que ce GRAFCET est composé de 9 étapes.

On associe naturellement un état du graphe à une étape du GRAFCET, et une transition du graphe d'états à une transition du GRAFCET ; ici, il est néanmoins nécessaire de préciser que cette opération requiert la plus grande attention et une bonne connaissance des règles du GRAFCET car certains GRAFCET, même sans parallélisme, peuvent difficilement se réduire à un graphe d'états.

Pour la représentation machine d'états, plusieurs solutions sont possibles pour le codage du vecteur d'état:

Si on choisit une structure machine de MOORE (une machine de MEALY supposerait la présence d'actions conditionnelles), on peut envisager d'associer à chaque étape du GRAFCET un état codé du vecteur d'état; notons que dans ce cas 4 bits d'états suffisent pour coder les 9 étapes.

Nous préférons un ***vecteur d'états décodés qui associe à chaque état un bit et un seul du vecteur d'état.*** Cette description plus proche de la notion GRAFCET qui associe une mémoire d'étape à chaque étape, est dénommée par les anglo-saxons ***codage "ONE HOT ONE"*** (un seul bit d'état chaud/actif à un instant donné).

Cette représentation du vecteur offre plusieurs avantages outre son rapprochement avec le GRAFCET :

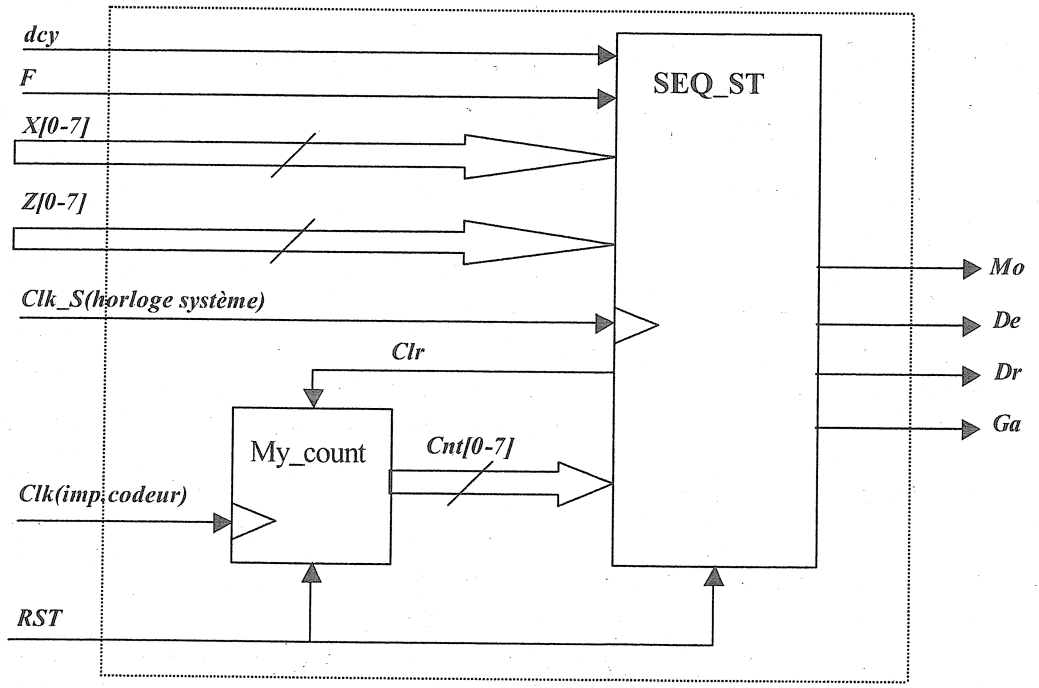
- Une telle machine sera la réplique exacte de la machine représentée par le graphe (un état = un bit) ; cela permet d'estimer rapidement de la complexité du système.
- La génération des sorties sera plus simple car les sorties sont activées par un état unique ou une combinaison d'états uniques plus simple que celle obtenue à partir des bits codés d'un vecteur habituel. (Le routage généré sera plus simple).
- Le codage "one hot one" permet également de scinder la machine d'états en sous-machines pouvant fonctionner parallèlement et donc traiter le parallélisme.

Le tableau suivant montre le vecteur associé à chaque type de machine et la représentation du vecteur d'états dans chaque cas pour le problème abordé ici :

Etape du GRAFCET	Machine à états codés		Machine à états décodés "one hot one"	
	Code décimal vecteur	Registre d'états(4 bits)	Code décimal vecteur	Registre d'états(9 bits)
1	0	0000	1	000000001
2	1	0001	2	000000010
3	2	0010	4	000000100
4	3	0011	8	000001000
5	4	0100	16	000010000
6	5	0101	32	000100000
7	6	0110	64	001000000
8	7	0111	128	010000000
9	8	1000	256	100000000

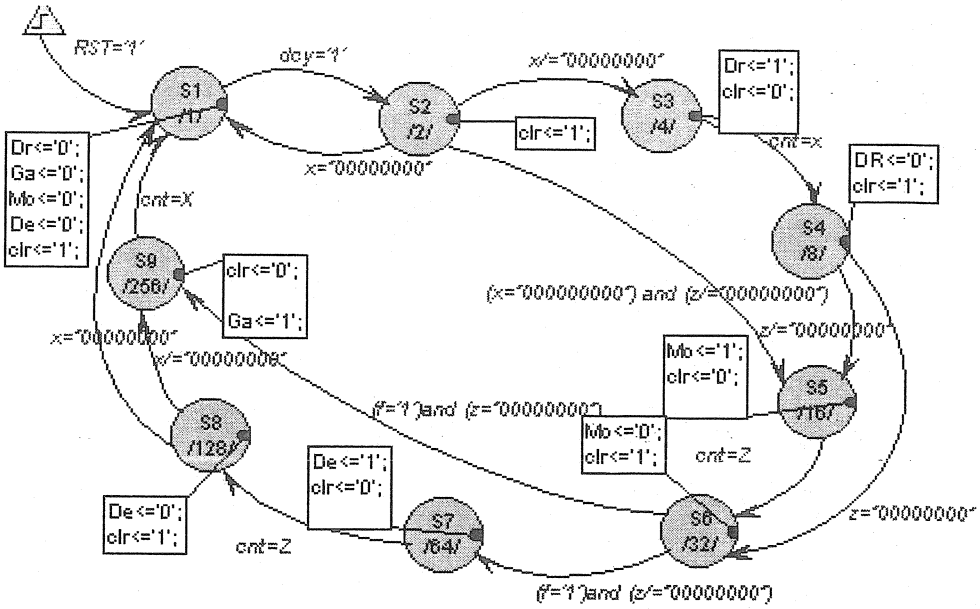
L'architecture générale du système comprend la machine d'états principale notée **SEQ_ST** et un compteur 8 bits **My_count** dont les sorties 8 bits sont notées **cnt.** La modularité propre au langage VHDL permet d'incorporer ce compteur comme un composant qui après avoir été décrit dans un paquetage extérieur peut être accédé à partir de notre architecture appelante.

Le bloc-diagramme ci-dessous montre l'association des deux modules:



L'ensemble après compilation peut être implanté dans un composant CPLD de type CYPRESS 37128.

Voici le Graphe d'états représentant la machine, les vecteurs 8 bits sont notés en binaire, et il fait apparaître les sorties associées à chaque état:



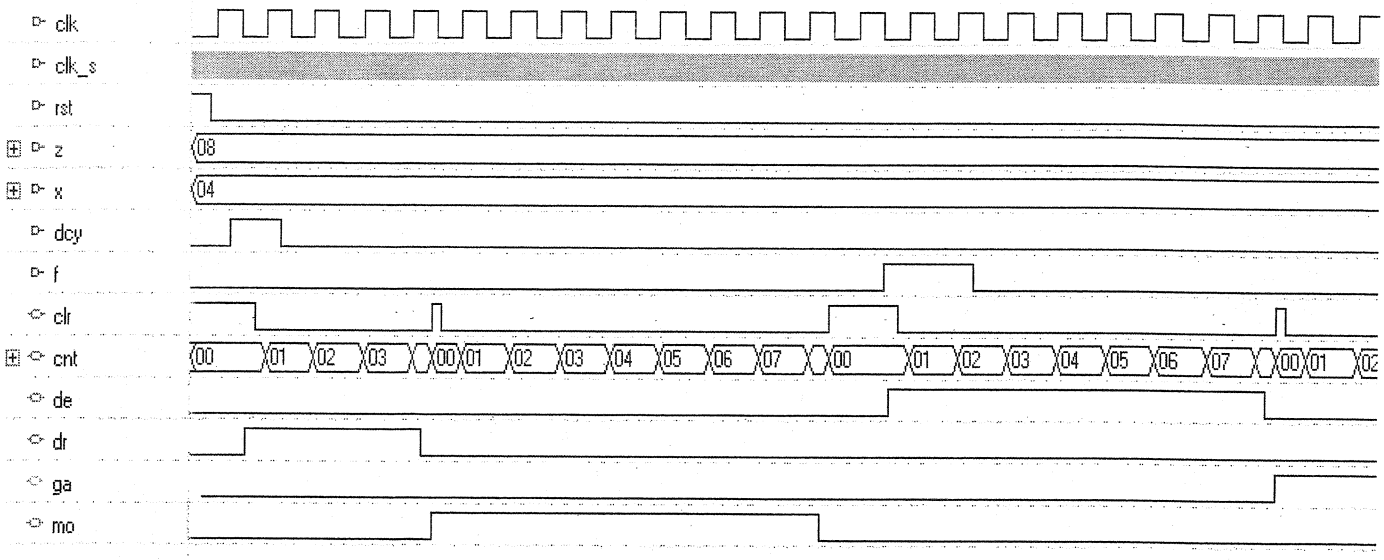
Voici partiellement le fichier VHDL généré:

```
library IEEE;
use IEEE.std_logic_1164.all;
use work.mycntpkg.all;--déclaration librairie composant(ici count)
entity axe_z_1 is
 port (CLK: in STD_LOGIC;
      CLK_s: in STD_LOGIC;
      dcy: in STD_LOGIC;
      f: in STD_LOGIC;
      RST: in STD_LOGIC;
      X: in STD_LOGIC_VECTOR (7 downto 0);
      z: in STD_LOGIC_VECTOR (7 downto 0);
      DE: out STD_LOGIC;
      DR: out STD_LOGIC;
      GA: out STD_LOGIC;
      Mo: out STD_LOGIC;
      clr: inout STD_LOGIC;
      cnt: inout STD_LOGIC_VECTOR (7 downto 0));
end;
architecture axe_z_1_arch of axe_z_1 is
-- ONE HOT ENCODED state machine: Sreg0
type Sreg0_type is (S1, S2, S3, S4, S5, S6, S7, S8, S9);
attribute enum_encoding of Sreg0_type: type is
          "000000001 " &          -- S1
          "000000010 " &          -- S2
          "000000100 " &          -- S3
          "000001000 " &          -- S4
```

```
        "000010000 " &          -- S5
        "000100000 " &          -- S6
        "001000000 " &          -- S7
        "010000000 " &          -- S8
        "100000000";            -- S9
signal Sreg0: Sreg0_type;
begin
--concurrent signal assignments
--diagram ACTIONS;
U0: count port map(clk, rst, clr, cnt(7 downto 0));--instanciation du compteur count
Sreg0_machine: process (CLK_s)
begin
if CLK_s'event and CLK_s = '1' then
        if RST='1' then
                Sreg0 <= S1;
                clr<='1';
        else
        case Sreg0 is
                when S1 =>
                        clr<='1';
                        if dcy='1' then
                                Sreg0 <= S2;
                        end if;
                when S2 =>
                        clr<='1';
                        if x="00000000" then
                                Sreg0 <= S1;
                        elsif (x="000000000") and (z/="00000000") then
                                Sreg0 <= S5;
                        elsif x/="00000000" then
                                Sreg0 <= S3;
                        end if;
                when S3 =>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                when others =>
                        null;
        end case;
        end if;
end if;
end process;
-- signal assignment statements for combinatorial outputs
Dr_assignment:
Dr <= '1' when (Sreg0 = S3) else
   '0' when (Sreg0 = S4) else
   '0';
Ga_assignment:
Ga <= '1' when (Sreg0 = S9) else
   '0';
```

```
Mo_assignment:
Mo <= '1' when (Sreg0 = S5) else
    '0' when (Sreg0 = S6) else
    '0';
De_assignment:
De <= '1' when (Sreg0 = S7) else
    '0' when (Sreg0 = S8) else
    '0';
end axe_z_1_arch;
```

Une simulation a été effectuée avec pour exemple Z=08 (soit 8 impulsions en Z) et X=04:



6 Conclusion :

L'approche machines d'états permet d'aborder de nombreuses applications, et notamment dans le domaine des automatismes séquentiels ; elle ne doit pas être opposée à l'approche GRAFCET mais plutôt considérée comme complémentaire et ciblant des applications plus modestes ne nécessitant pas l'utilisation d'un automate programmable industriel.

La plupart des outils modernes de synthèse logique (VHDL) permettent ce type de description et offrent des facilités de développement et de mise au point tels que WARP2, simples à mettre en œuvre et peu coûteux.

Une réflexion plus approfondie sur les transpositions GRAFCET/Machines d'états devrait permettre l'utilisation de descriptions GRAFCET pour l'implantation de celui-ci dans des cibles en logique programmable.

7 Bibliographie :

-"Du GRAFCET aux réseaux de PETRI" DAVID&ALLA (HERMES).

-"VHDL Introduction à la synthèse logique" P.LARCHER (EYROLLES).

-"VHDL for programmable logic"K.SKAHILL(CYPRESS).

LES BOND GRAPH, UNE AIDE PRECIEUSE POUR LA MODELISATION DES SYSTEMES ELECTROMECANIQUES

Jean FAUCHER
Laboratoire d'Electrotrotechnique et d'Electronique Industrielle – INPT - ENSEEIHT
2, rue Camichel, 31071 TOULOUSE Cedex

La technique du Bond Graph est basée sur la représentation des échanges énergétiques au sein d'un système. Elle permet d'établir de façon systématique le Graphe de Causalité et les équations d'état qui caractérisent le comportement dynamique du système. Une introduction à cette technique est présentée et illustrée par quatre exemples de systèmes électromécaniques.

Introduction

L'établissement des équations dynamiques des systèmes électromécaniques recèle de nombreuses difficultés. Le jeune virtuose et le vieux briscard s'en sortent souvent bien, parfois d'ailleurs de façon intuitive. Mais tel n'est souvent pas le cas pour le modélisateur occasionnel. Il est alors intéressant, sinon indispensable, de pouvoir disposer de méthodes systématiques qui peuvent en particulier être proposées aux apprenants. Les méthodes avec représentation graphique sont particulièrement bienvenues par la vision globale et synthétique qu'elles proposent, nous connaissons tous le succès des schéma-bloc. Le Graphe Informationnel Causal (GIC) propose une représentation des flux d'information orientés par la causalité et se révèle un outil remarquable pour l'établissement des lois de commande et pour la simulation dynamique des systèmes, sans hypothèse de linéarité. Nous considérons dans cet article que l'établissement du GIC représente l'objectif de modélisation, les équations d'état s'en déduisant sans ambiguïté. Plusieurs écueils attendent le modélisateur mis en présence d'un nouveau système électromécanique :

- Comment décomposer le système en sous systèmes ?
- Comment caractériser les liens entres les sous systèmes ?
- Quelles sont les variables indispensables et comment définir les conventions de signe qui leurs sont attachées ?

Les deux premiers points sont cruciaux et il est bien évident, en particulier pour la méthode du GIC, que l'expérience joue un rôle essentiel. Il est en effet souvent possible de « reconnaître » dans le système, des éléments déjà vus et modélisés sous forme de processeurs entrées-sorties. Le GIC se prête particulièrement bien à l'agrégation de modèles préalablement connus. Nous allons cependant nous placer dans la situation d'un système complètement nouveau pour le modélisateur et décrire une méthode qui nous conduit sans ambiguïté vers le/les GIC possibles. Cette méthode est la méthode des Bond Graph (nous adoptons l'abréviation BG) ou Graphes de Liens en français. Définie et formalisée depuis relativement peu de temps (PAYNTER 1961, KARNOPP, ROSENBERG 1975 et 1983), elle reprend avec succès diverses méthodes systématiques de mise en équations utilisées par les mécaniciens. L'étude que nous présentons "oublie" la méthode dite des "équations de Lagrange" dont est directement issu le Bond Graph. Cet oubli est volontaire par les possibilités immenses offertes par un traitement graphique pour l'établissement de modèles. Nous sommes tout à fait d'accord avec les "gardiens du temple" : OUI la méthode des équations de Lagrange permet d'effectuer la même modélisation que les BG, mais dans quel inconfort! La parenté entre les équations de Lagrange et les Bond Graph est totale. Nous faisons dans cette article une présentation simplifiée pour répondre aux besoins les plus courants des modélisateurs occasionnels de l'Electromécanique. Si le lecteur prend goût à cette technique, nous ne pouvons que l'inciter à aller au delà en consultant les ouvrages spécialisés.

Principes du Bond Graph.

Un objet physique au sein d'un ensemble d'objets voit son comportement influencé par son environnement, réciproquement il influence lui-même cet environnement. Cette influence croisée objet-environnement se traduit généralement par des échanges énergétiques. A partir de cette constatation, le Bond Graph a pour principe de représenter sous forme graphique les échanges énergétiques entres les objets constitutifs d'un système physique. Le

système est alors considéré comme constitué d'« objets énergétiques communicants ». Un tel objet possède un nombre fini de « portes d'entrée-sortie énergétiques » points de départ et d'arrivée des « routes » (***liens = Bond*** en anglais) de transport des énergies qui le relient aux autres objets. Dans notre exposé, les liaisons thermiques éventuelles sont implicites et considérées comme résultant d'échanges unilatéraux avec le milieu extérieur qui n'intervient que comme récepteur. elles ne sont donc pas représentées. Les convertisseurs thermiques (moteurs thermiques, effet Peltier...) ne sont pas examinés dans cet exposé.

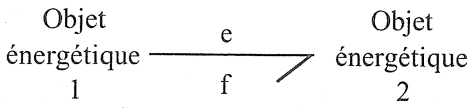


Figure 1

A un lien sont associées *2* ***variables*** dont le produit est la puissance transportée (par exemple : *tension-courant , force-vitesse*).

Le lien est orienté vers l'objet qui «reçoit » une puissance positive. L'orientation d'un lien est laissée au choix du grapheur (l'algèbre est un progrès indéniable). Il convient cependant de choisir le sens positif d'un lien en faisant preuve d'un certain sens physique. La Figure 1 donne la représentation d'un lien. Les deux variables associées à la puissance transportée sont (nous indiquons entre parenthèses la dénomination correspondante pour les variables généralisées de Lagrange) :

e = variable d'effort (force généralisée p°)

f = variable de flux (vitesse généralisée q°)

puissance portée par le lien = **e.f**

Variables Généralisées	*Variables BG*	*Electricité*	*Mécanique Translation*	*Mécanique Rotation*
q		Q charge électrique	x position	θ position angulaire
$\frac{dq}{dt} = q^{\circ}$	f *flux*	i Courant	x° vitesse de translation	$\Omega = \theta^{\circ}$ vitesse angulaire
p		Φ flux magnétique	$P = M.x^{\circ}$ moment	$P = J.\Omega$
$\frac{dP}{dt} = p^{\circ}$	e *effort*	v tension	F force	C couple

Figure 2

Les variables Bond Graph que nous rencontrons dans les systèmes électromécaniques sont triées en variables d'effort et variables de flux suivant le tableau de la Figure 2.

Nous avons souligné dans l'introduction notre volonté de démarrer la modélisation sans connaissance a priori. Il est donc nécessaire de décomposer le système considéré en objets élémentaires non décomposables (*quanta*) connus *a priori*.
Ces objets sont en nombre limité et classés en fonction de leur nature énergétique

- récepteurs d'énergie, accumulateurs ou dissipateurs : **Eléments BG à une ou plusieurs « portes »**
- **sources BG** d'effort ou de flux
- transmetteurs d'énergie : **Jonctions BG,**

Eléments BG

Les éléments BG n'ont qu'un seul lien avec le reste du système considéré, nous adoptons systématiquement une convention récepteur (Figure 3). Les éléments BG sont dénommés en fonction de la nature de leur comportement énergétique. On distingue alors 3 cas :

dissipatif **accumulateur d'énergie potentielle** **accumulateur d'énergie cinétique**

Figure 3

Ces cas ne préjugent en rien de la nature, mécanique, électrique ou thermique de l'objet physique auquel se réfère cet élément. La modélisation par Equations de Lagrange, donc par Bond Graph, est en effet une modélisation fédératrice qui s'applique sous la même forme à des systèmes hétérogènes.

Elément R

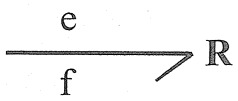


Figure 4

Il s'agit d'un élément dissipatif. L'élément impose une relation de contrainte entre la variable d'effort **e** et la variable de flux **f** :

$$\text{fonction}(f, e) = 0$$

La représentation de l'élément est donnée par la Figure 4
choix du signe du lien : élément récepteur (naturel)

exemple électrique : résistance [fonction(v,i)=0]
exemple mécanique : frottement visqueux [fonction(F,$\dot{x}$)=0]

Elément I

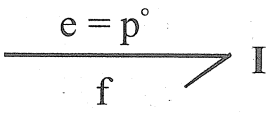


Figure 5

Il s'agit d'un élément accumulateur d'énergie cinétique.
L'élément impose une relation de contrainte entre la variable de flux **f** et le moment généralisé **p** (avec $e = \dot{p}$)

$$\text{fonction}(p, f) = 0$$

Sa représentation est donnée à la Figure 5.
choix du signe du lien : élément récepteur (plus pratique)

exemple électrique : inductance [fonction(flux magnétique,courant)=0]
exemple mécanique : masse en translation [fonction(moment cinétique,vitesse)=0]

Elément C

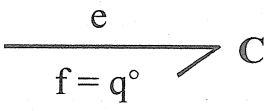


Figure 6

Il s'agit d'un élément accumulateur d'énergie potentielle. L'élément impose une relation de contrainte entre la variable d'effort **e** et la position généralisée **q** (avec $f = \dot{q}$)

$$\text{fonction}(q, e) = 0$$

Sa représentation est donnée à la Figure 6.
choix du signe du lien : élément récepteur (plus pratique)

exemple électrique: condensateur [fonction(charge électrique,tension)=0]
exemple mécanique : ressort [fonction(allongement,force)=0]

source d'effort

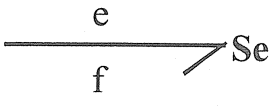


Figure 7

L'élément source impose l'effort. Le flux est imposé par le reste du système.

Sa représentation est donnée à la Figure 7.
choix du signe du lien : arbitraire (suivant convenance)

exemple électrique : source de tension
exemple mécanique : source de couple

Source de flux

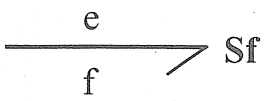


Figure 8

L'élément source impose le flux. L'effort est imposé par le reste du système

Sa représentation est donnée à la Figure 8
choix du signe du lien : arbitraire (suivant convenance)

exemple électrique : source de courant
exemple mécanique : source de vitesse

Jonctions

Les jonctions ne font que transmettre l'énergie sans en emmagasiner ni en dissiper. Elles assurent les liaisons « multiportes » et traduisent les contraintes de connexion entre les objets : 2 objets mécaniques solidaires sont entraînés à la même vitesse, 2 objets électriques connectés en parallèle sont soumis à la même tension. On distingue :

- Jonction 0 : connexion d'éléments soumis au même effort (Figure 9)
- Jonction 1 : connexion d'éléments soumis au même flux (Figure 10)

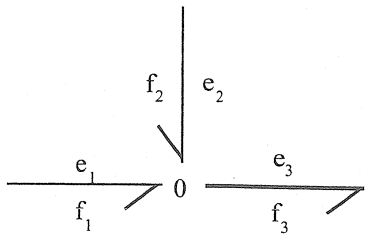


Figure 9

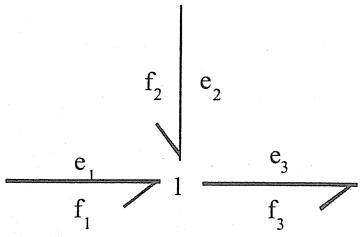


Figure 10

En tenant compte de l'orientation des liens pour les deux exemples présentés, on obtient :

$$\text{jonction 0}\begin{cases} e_1 = e_2 = e_3 \\ f_1 + f_2 - f_3 = 0 \end{cases} \qquad \text{jonction 1}\begin{cases} e_1 + e_2 - e_3 = 0 \\ f_1 = f_2 = f_3 \end{cases}$$

Il nous reste enfin à décrire deux jonctions particulières à deux portes: **Transformateur (TF)** et **Gyrateur (Gy).**
La jonction transformateur (Figure 11) implique les relations :

$$\begin{cases} e_1 = g(e_2) \\ f_1 = h(f_2) \end{cases} \quad \text{avec} \quad e_1 f_1 = e_2 f_2$$

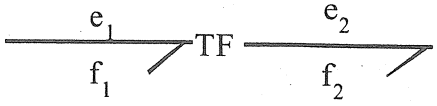


Figure 11

Un exemple électrique classique de jonction TF est le transformateur électrique parfait pour lequel nous avons :

primaire $\begin{cases} \text{variable d'effort : tension } v_1 \\ \text{variable de flux : courant } i_1 \end{cases}$

secondaire $\begin{cases} \text{variable d'effort : tension } v_2 \\ \text{variable de flux : courant } i_2 \end{cases}$

Nous obtenons les relations : $\begin{cases} v_1 = k.v_2 \\ i_1 = \dfrac{1}{k} i_2 \end{cases}$

Un exemple mécanique tout aussi classique est le réducteur tournant parfait pour lequel nous aurons :

primaire $\begin{cases} \text{variable d'effort : couple } C_1 \\ \text{variable de flux : vitesse } \Omega_1 \end{cases}$

secondaire $\begin{cases} \text{variable d'effort : couple } C_2 \\ \text{variable de flux : vitesse } \Omega_2 \end{cases}$

Nous obtenons les relations : $\begin{cases} C_1 = k.C_2 \\ \Omega_1 = \dfrac{1}{k} \Omega_2 \end{cases}$

La jonction gyrateur (Figure 12) implique des relations croisées :

$$\begin{cases} e_1 = g(f_2) \\ f_1 = h(e_2) \end{cases} \quad \text{avec} \quad e_1 f_1 = e_2 f_2$$

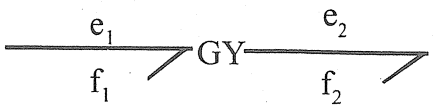


Figure 12

Un exemple électromécanique de jonction Gy est bien connu des électrotechniciens puisqu'il s'agit de la partie « conversion électromécanique parfaite » du moteur à courant continu :

primaire $\begin{cases}\text{variable d'effort : fem E}\\ \text{variable de flux : courant d'induit i}\end{cases}$

secondaire $\begin{cases}\text{variable d'effort : couple C}\\ \text{variable de flux : vitesse }\Omega\end{cases}$

Nous avons les relations : $\begin{cases}E = k.\Omega\\ i = \frac{1}{k}C\end{cases}$

Couplages électromécaniques

Un couplage électromécanique s'effectue par l'intermédiaire d'une énergie emmagasinée : énergie magnétique ou énergie électrostatique. On obtient alors un élément multiportes électriques et mécaniques. Considérons par exemple un convertisseur électromagnétique, Nous obtenons un comportement de type I pour les liens électriques et un comportement de type C pour les liens mécaniques (Figure 13). Considérons par exemple une bobine (tension v, courant i) et un noyau ferromagnétique mobile (force F, position x). Nous avons :

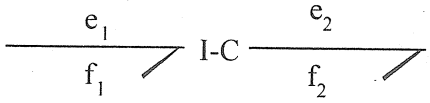


Figure 13

$$\begin{cases}e_1 = v = \frac{d\Phi}{dt}\\ f_1 = i\end{cases} \quad \text{et} \quad \begin{cases}e_2 = F\\ f_2 = \frac{dx}{dt}\end{cases}$$

et les relations de contraintes suivantes :

$$\begin{cases}F = \text{fonction}(\Phi, x)\\ i = \text{fonction}(\Phi, x)\end{cases}$$

Causalité

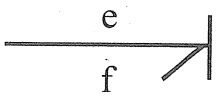


Figure 14

La causalité est représentée sur un lien BG par un trait vertical du coté où l'effort est imposé par le lien (Figure 14).
Nous ne retenons que la causalité intégrale qui seule a une signification physique. L'intégration temporelle est en effet une opération strictement causale (le résultat à un instant donné ne dépend que des informations passées). il est bien entendu que l'opération inverse : la dérivation n'est alors plus causale et n'a donc pas de signification physique.
Dans ces conditions nous obtenons les conséquences de causalité suivantes.

Elément R : causalité indifférente induite par le contexte ;

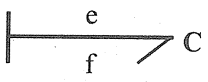


Figure 16

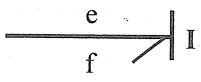


Figure 15

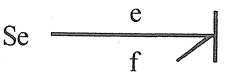


Figure 176

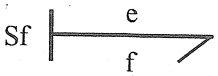


Figure 18

Elément C : trait causal du coté opposé à l'élément C (Figure 16).
Elément I : trait causal coté élément I (Figure 15) ;

pour les sources, la causalité va de soit (Figure 17 et Figure 18)
Pour les jonctions, nous obtenons les règles de causalité suivantes :

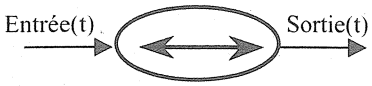
- jonction 0 : un seul trait causal (l'effort commun ne peut être imposé que par un lien)
- jonction 1 : un seul non-trait causal (le flux commun ne peut être imposé que par un lien)
- jonction TF : un seul trait causal (les deux liens sont de nature causale inverse pour la jonction)
- jonction GY : 2 traits causaux ou 0 trait causal (les deux liens sont de même nature causale pour la jonction)

Relation entre BG et Graphe Informationnel Causal (GIC)

Le Bond Graph étant établi et les causalités indiquées (si nécessaire, les éventuels conflits de causalité doivent être levés à ce moment là en reconsidérant le découpage en éléments de base), le GIC est établi de façon simple et sans ambiguïté et se superpose au BG dont il ne conserve que l'aspect informationnel en explicitant les processeurs.
Rappelons quelques principes de base des GIC.

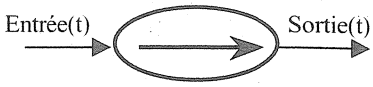
- Le GIC traite des flux des informations portées par les variables et orientées de la cause vers l'effet ;

- Les transformations d'informations sont effectuées par des processeurs ;
- On distingue :



Processeur rigide

Figure 18



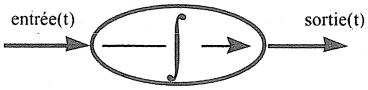
Processeur causal

Figure 19

- processeurs causaux (orientation par la causalité temporelle) (Figure)
- processeurs rigides (pas de causalité temporelle) (Figure)

La modélisation par GIC qui correspond à la modélisation BG est directement construite à partir des processeurs de base suivants :

-intégrateur (causal) (Figure 19)
-sommateur (rigide) (Figure 20)



processeur intégral

Figure 19

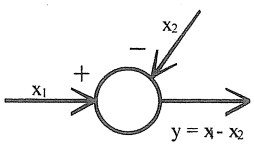


Figure 20

auxquels s'ajoutent des processeurs rigides traduisant des relations fonctionnelles (Figure).
Plutôt qu'un long discours théorique, il est temps de traiter quelques exemples.

Exemples BG et GIC

Nous examinons 4 exemples caractéristiques de notre domaine qui illustrent la démarche de modélisation : BG, GIC puis écriture des équations d'état. Nous détaillerons seulement les deux premiers exemples.

Exemple de circuit électrique

Nous considérons le circuit électrique de la Figure 21. Le BG se construit à partir des remarques suivantes :

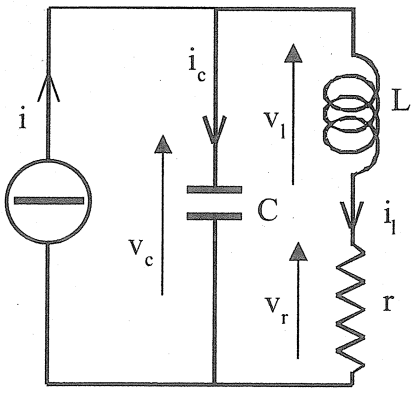


Figure 21

- jonction 0 (v_c) condensateur // inductance-résistance // source
- jonction 1 (i_l) inductance série résistance
- 1 élément I (Inductance)
- 1 élément C (Condensateur)
- 1 élément R (résistance)
- 1 source de flux (source de courant)

Les causalités sont d'abord inscrites sur les sources et éléments C et I pour lesquels elles sont imposées. Les autres causalités (R) s'en déduisent à partir des règles sur les jonctions (1 seul trait causal sur la jonction 0, 1 seul non-trait causal sur la jonction 1). avec les relations :

$i - i_c - i_l = 0$ (jonction 0) et $v_c - v_l - v_r = 0$ (jonction 1)

Nous obtenons le Bond Graph de la Figure 22. Nous n'avons aucun choix pour les causalités. Le GIC s'en déduit directement (Figure 23).

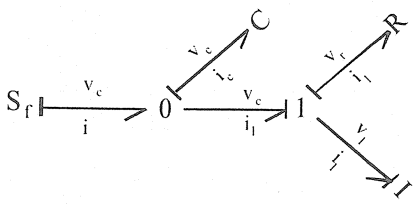


Figure 22

Les équations d'état se construisent de la façon suivante : chaque sortie d'intégrateur est une variable d'état et son entrée la dérivée de la variable d'état. On aura donc autant d'équations différentielles d'ordre 1 que de variables d'état. Le second membre de chaque équation se construit à partir de l'entrée de l'intégrateur correspondant en « remontant » les trajectoires d'information jusqu'à rencontrer une variable d'état ou une entrée globale.
Il est à noter que les processeurs rigides que nous rencontrons sur le trajet, autres que les sommateurs, ne sont pas nécessairement linéaires (ils ne le sont même sûrement pas).
Dans l'exemple considéré nous supposons cependant la linéarité de

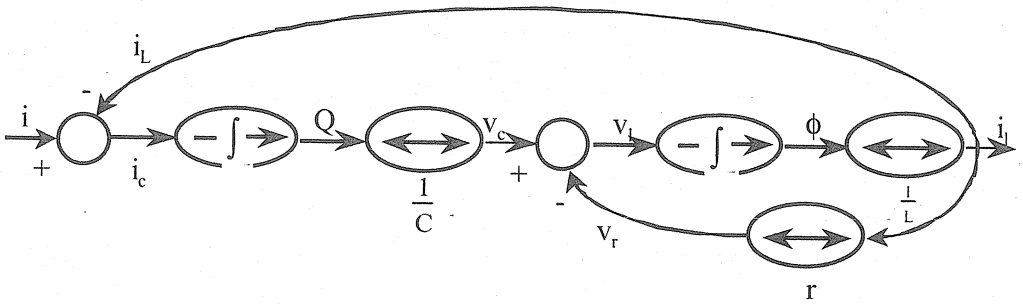


Figure 23

tous les processeurs. Nous obtenons les équations d'état suivantes :

$$\frac{dQ}{dt} = i_c = i(t) - \frac{1}{L}\Phi$$

$$\frac{d\Phi}{dt} = v_L = \frac{1}{C}Q - \frac{r}{L}\Phi$$

exemple de système mécanique

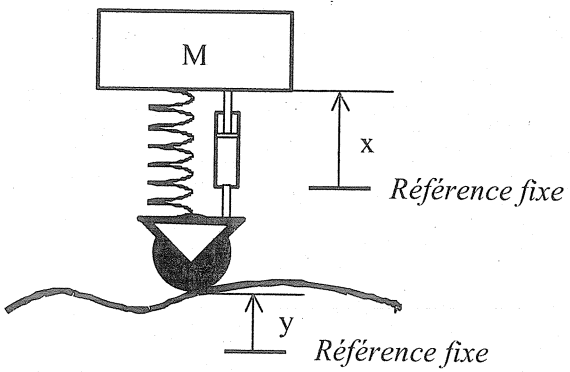


Figure 24

Nous considérons le dispositif de suspension de la roulette d'un robot mobile (Figure 24). Ce dispositif est constitué d'un amortisseur et d'un ressort. La masse suspendue est M. Nous considérons que le déplacement est vertical et que le robot se déplace sur la Terre. Nous supposons le ressort et l'amortisseur sans masse (leurs masses suspendues sont intégrées à la masse M)
Le BG se construit à partir des considérations suivantes :

- Les points mécaniquement solidaires sont entraînés à la même vitesse (jonctions 1) ;
- La force à une extrémité du ressort est intégralement transmise à l'autre extrémité (jonction 0)
- Il en est de même pour l'amortisseur (jonction 0) ;
- Le ressort accumule de l'énergie potentielle (Elément C)
- L'amortisseur dissipe de l'énergie (Elément R) ;
- La masse suspendue accumule de l'énergie cinétique (Elément I) ;
- Le sol se comporte comme une source de flux (la vitesse $\dot{y}$ est imposée par le profil du sol et la vitesse horizontale) ;

Nous possédons maintenant toutes les pièces du puzzle et pouvons construire le Bond Graph (Figure 25).
Nous avons les relations suivantes :

jonction 0 (même effort F)

$\dot{y}$ est imposée;
$\dot{x}$ = vitesse verticale de la masse suspendue ;
$\dot{d}$ = vitesse de compression du ressort et de l'amortisseur ;
F = force transmise à la masse suspendue

$$\dot{y} - \dot{d} - \dot{x} = 0$$

jonction 1 (même vitesse $\dot{d}$)

F_a = force transmise à l'amortisseur;
F_r = force transmise au ressort ;

$$F - F_a - F_r = 0$$

jonction 1 (même vitesse $\dot{x}$)

Mg = force due à la pesanteur (source S_e)
F_M = force inertielle transmise à la masse M

$$F - F_M - Mg = 0$$

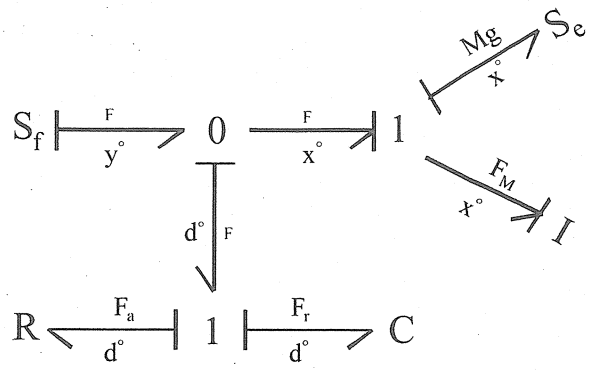


Figure 25

Il est à noter que là encore, nous n'avons aucun choix pour le placement des traits causaux. il s'avère en fait qu'il est relativement rare d'avoir le choix et beaucoup moins rare de se trouver devant un conflit de causalité qui oblige éventuellement à revoir le découpage en éléments. Le GIC se déduit facilement du BG (Figure 26).

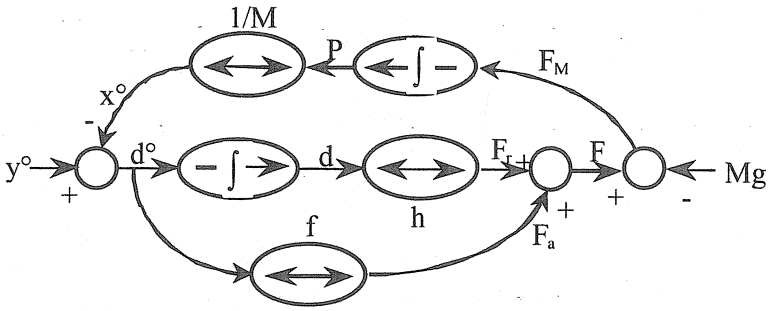


Figure 26

Les équations d'état s'en déduisent : (nous appelons f et h les fonctions caractéristiques respectivement de l'amortisseur et du ressort)

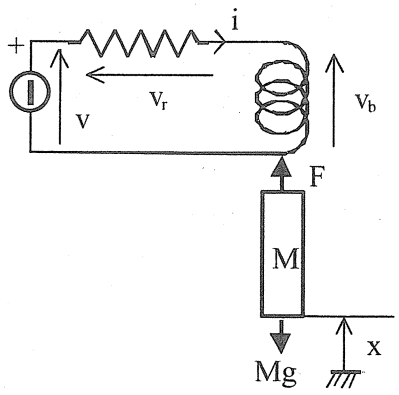


Figure 27

$$\frac{dP}{dt} = F_M = h(d) + f(\dot{y}(t) - \frac{1}{M}P) - Mg$$

$$\frac{dd}{dt} = \dot{d} = -\frac{1}{M}P + \dot{y}(t)$$

suspension magnétique (Figure 27)

Nous considérons une bobine parcourue par un courant i et attirant verticalement un noyau de masse M. Le noyau est soumis à la pesanteur terrestre. Le frottement mécanique est négligeable.

Nous obtenons le BG de la Figure 28 et le GIC de la Figure 29.

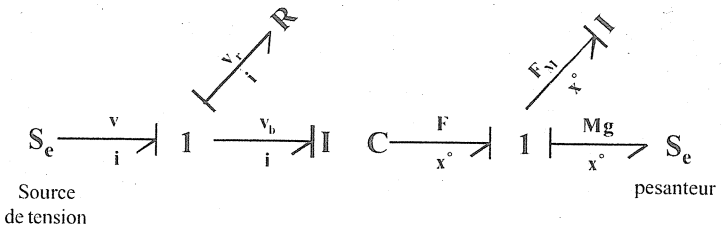


Figure 28

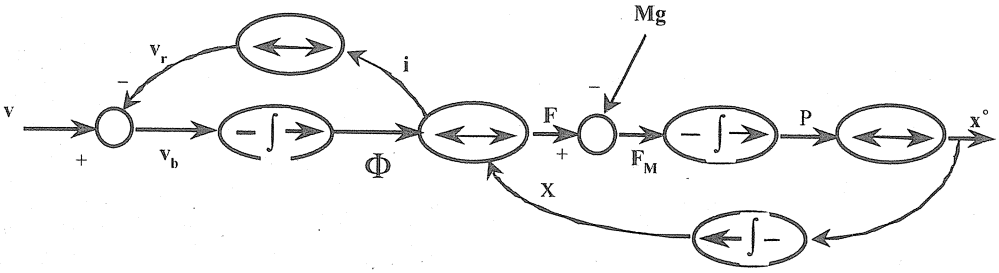


Figure 29

treuil entraîné par une machine à courant continu

Nous choisissons maintenant un exemple nettement plus difficile. On considère une machine à courant continu à commande d'induit qui entraîne le treuil d'un monte-charge par l'intermédiaire d'un arbre long élastique (Figure 31). Les paliers et ventilateurs induisent des frottements visqueux (liés à la vitesse de rotation) tant sur l'arbre du moteur que sur l'arbre du treuil. Le Bond Graph obtenu est présenté par la Figure 30. L'induit du moteur (courant *i*) est modélisé par une résistance en série avec une inductance et une source de fem *E* fonction de la vitesse de rotation. Le couple électromagnétique est noté T_m. Nous prenons en compte de façon séparée : l'inertie et les frottements coté moteur, l'inertie et les frottements du treuil seul, l'inertie de la charge levée et bien entendu le poids de cette dernière. Nous prenons également en compte l'élasticité de l'arbre de transmission (Θ est l'angle de torsion de l'arbre élastique) et l'élasticité de l'élingue (*d* est l'allongement de celle-ci).

Cet exemple, relativement complexe illustre bien à notre avis les 2 principales qualités du Bond Graph: l'aisance de la modélisation et la mise en évidence des éventuels conflits de causalité. il convient en particulier de noter la facilité que cette technique apporte pour le délicat problème des conventions de signe sur les grandeurs mécaniques. Il n'est en effet besoin que de préciser les conventions de signe sur les transferts de puissance, ce qui ne pose pas de problème. Les conventions de signes sur les grandeurs d'effort et de flux s'en déduisent implicitement.

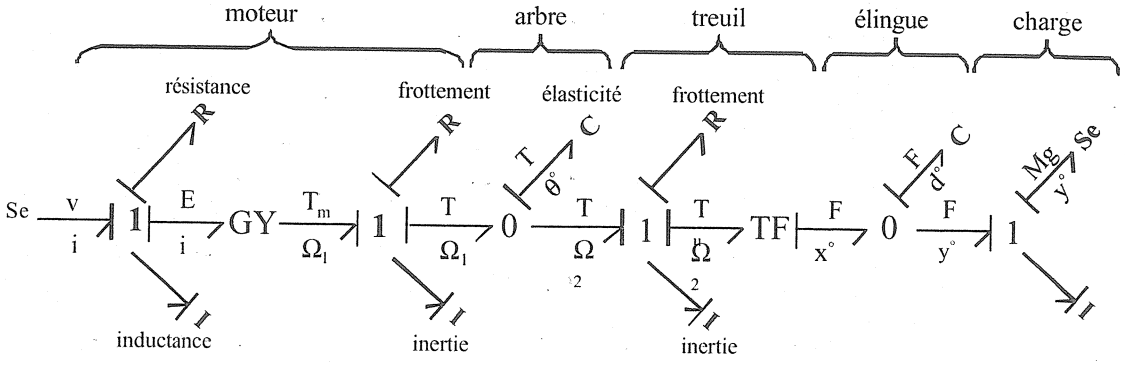


Figure 30

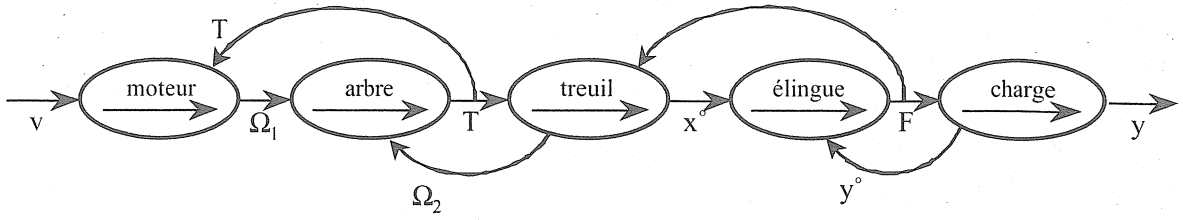


Figure 32

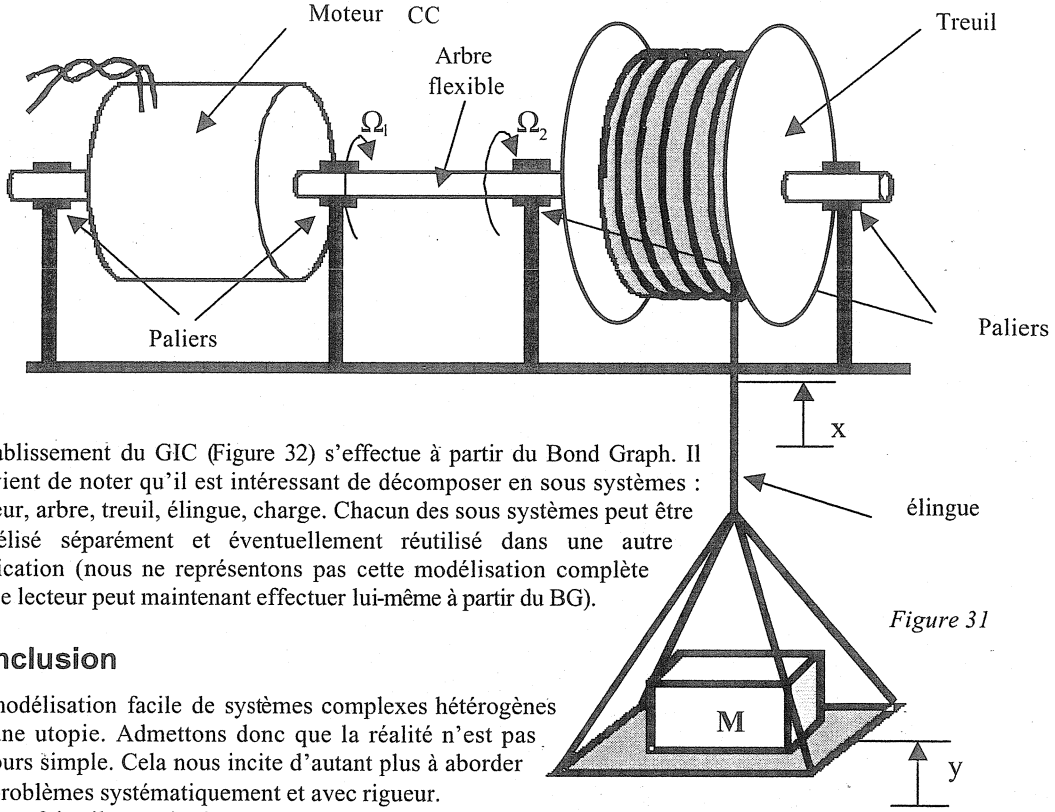


Figure 31

L'établissement du GIC (Figure 32) s'effectue à partir du Bond Graph. Il convient de noter qu'il est intéressant de décomposer en sous systèmes : moteur, arbre, treuil, élingue, charge. Chacun des sous systèmes peut être modélisé séparément et éventuellement réutilisé dans une autre application (nous ne représentons pas cette modélisation complète que le lecteur peut maintenant effectuer lui-même à partir du BG).

Conclusion

La modélisation facile de systèmes complexes hétérogènes est une utopie. Admettons donc que la réalité n'est pas toujours simple. Cela nous incite d'autant plus à aborder ces problèmes systématiquement et avec rigueur.
Pour ce faire il est sain de mettre en œuvre des concepts solidesde la physique. Les principes qui sont en filigrane dans cet exposé sont ceux de la thermodynamique et mécanique physique : Principe de Hamilton (dit aussi principe de moindre action), principe de conservation de l'énergie et équations de Lagrange. La technique du Bond Graph permet de mettre en œuvre ces principes de façon graphique quasi concrète et d'aboutir au GIC, donc aux équations du mouvement du système. L'introduction de cette méthode dans l'enseignement est cependant largement fonction du niveau de l'apprenant. La méthode, malgré son aspect visuel, demande en effet une forte capacité d'abstraction. Cette réserve étant faite, je me tourne vers l'enseignant et souligne combien cette méthode associée à celle des GIC peut lui être utile pour préparer une modélisation en dégageant les points importants qui orienteront son discours pédagogique. Ceci n'implique donc pas qu'il faille enseigner à tous les Bond Graph et les GIC, mais qu'il s'agit d'outils parmi d'autres qui permettent de trouver son chemin vers une modélisation presque sereine de la dynamique des systèmes électromécaniques.

courte Bibliographie

- Les Bond Graphs, G. Dauphin Tanguy, Ed Hermès, Traité IC2
- System dynamics, modeling and simulation of mecatronic systems, D. C.Karnopp, & all Wiley Interscience publication, 2000.
- Simulation by Bond Graph, J. Thoma, Springer Verlag, 1991- Simulation by Bond Graph, J. Thoma, Springer Verlag, 1991
- Mécanique , L. LANDAU et F. LIFCHITZ, éditions MIR , MOSCOU
- Systèmes Automatiques, J.-P. HAUTIER, J.-P. CARON éditions Ellipses , Paris, 1997
- Le Graphe Informationnel Causal, J.-P. HAUTIER, J. FAUCHER, BUP 785, 1996

COMMANDE EN BOUCLE FERMEE DE MOTEURS PAS A PAS

Principe et application

Franck BETIN, Daniel PINCHON
Centre de Robotique, d'Electrotechnique et d'Automatique
Institut Universitaire de Technologie de l'Aisne
15 avenue François Mitterrand, Les terrasses du Mail, 02880 Cuffies
Tel 03 23 76 40 24, fax 03 23 76 40 25, e-mail : franck.betin@iut.u-picardie.fr

Après avoir rappelé le fonctionnement d'un moteur pas à pas en boucle ouverte, cet article propose une commande de ce moteur en boucle fermée par logique floue avec un exemple d'implantation de cette commande sur un micro-contrôleur de la famille Intel

Introduction

Dans les années 1960, l'actionneur synchrone pas à pas a vu s'ouvrir un créneau très porteur avec le développement de l'informatique. L'aspect incrémental de sa commande (une impulsion d'horloge = un pas) est en effet particulièrement bien adapté aux données numériques de l'ordinateur puisqu'un simple train d'impulsions suffit pour positionner correctement le moteur.

Aujourd'hui, l'essor des composants de l'électronique de puissance et les progrès des systèmes de traitement de l'information ont permis à l'actionneur électrique pas à pas d'élargir considérablement son champ d'applications, notamment dans les domaines de la robotique et de la commande numérique de machines-outils. Pour ces deux types d'applications, la précision dynamique est primordiale. En effet, l'outil mû par l'actionneur pas à pas doit suivre une trajectoire précise avec une vitesse donnée tout en fournissant un effort et ceci quelles que soient les variations de la configuration mécanique. Ces variations brusques peuvent en effet engendrer la dégradation des performances dynamiques du moteur pas à pas, voire le décrochage. Dans ce cas, une commande en boucle ouverte ne convient pas et il faut se résoudre à employer une commande en boucle fermée.

La rapidité de l'actionneur pas à pas ne permet pas l'emploi de techniques gourmandes en calcul telle que la commande adaptative par régulateur auto-ajustables. Aussi, dans cet article, nous proposons-nous d'appliquer la commande par logique floue. Cette technique ne nécessite que peu de calculs à effectuer en temps réel et présente des caractéristiques très intéressantes quant à la robustesse vis à vis des variations de la chargemécanique.

Dans la première partie de cet article, nous rappellerons brièvement le principe de rotation d'un moteur pas à pas, le type d'alimentation employée ainsi que les principes de génération des commandes en boucle ouverte. Nous verrons également, à l'aide d'un profil de déplacement trapézoïdal, l'insuffisance d'une commande en boucle ouverte quant à la précision lors des phases d'accélération, de décélération et sur les paliers à vitesse constante.

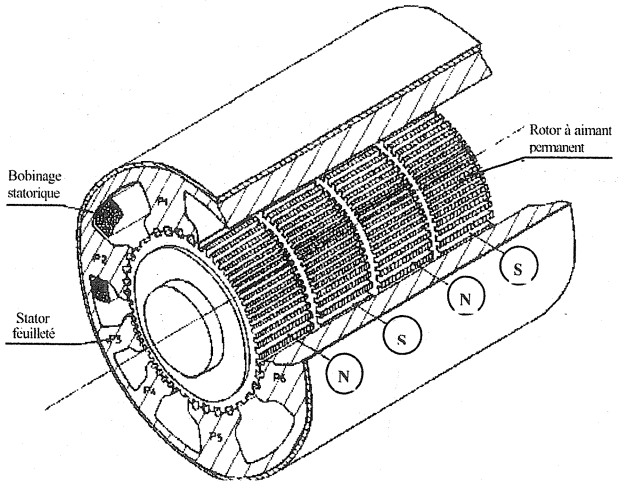
La seconde partie de cet article sera consacrée à la commande en boucle fermée du moteur pas à pas et plus particulièrement à la commande par logique floue de la vitesse de l'actionneur considéré. Les performances de cette commande seront mises en évidence à l'aide de la simulation des dynamiques du moteurlors d'un profil de déplacement trapézoïdal.

Dans la troisième partie de cet article, nous aborderons l'implantation en temps réel de la commande par logique floue sur un microcontrôleur 80C196kc de la famille Intel. Ce composant, cadencé à 16 MHz et disposant de 16 bits de données, est un standard dans le domaine de la commande des moteurs électriques. Nous présenterons les astuces utilisées pour minimiser les temps de calcul et vérifierons alors de manière expérimentale les performances de ce régulateur flou à l'aide d'un banc d'essais.

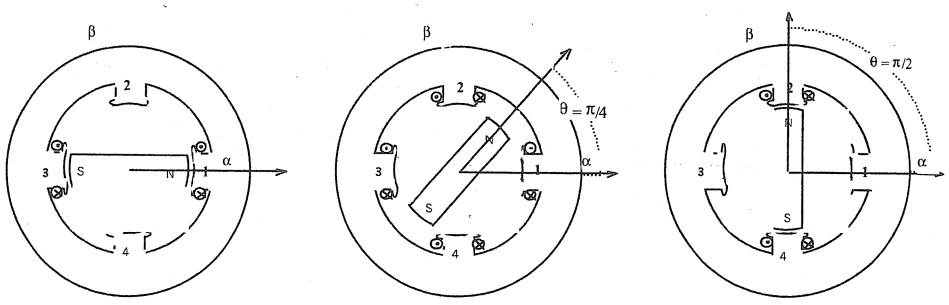
1. Généralités sur le moteur pas à pas

1.1 Principe de rotation

Un moteur pas à pas est un transducteur permettant de transformer une énergie électrique en une énergie mécanique qui se commande aisément en position ou en vitesse. A chaque impulsion ou modification élémentaire de la configuration d'alimentation correspond, au niveau du rotor, un déplacement angulaire élémentaire appelé pas. La figure 1.a représente la coupe d'un moteur pas à pas hybride, tandis que les figures 1.b à 1.d représentent sa structure simplifiée. Le rotor est constitué d'un aimant permanent et le stator comporte quatre plots sur lesquels sont bobinées deux phases notées α et β. Si la phase α est alimentée par un courant constant I_n, le rotor se positionne dans l'axe du champ magnétique créé (figure 1.b). Pour faire avancer le rotor d'un pas, il suffit de couper le courant dans la phase α et de l'établir dans la phase β (figure 1.d). De la même manière, pour faire avancer le rotor d'un demi-pas, on alimente simultanément la phase α et la phase β (figure 1.c).



a) Structure complète d'un moteur pas à pas



b) $I_\alpha = I_n$ $I_\beta = 0$ c) $I_\alpha = I_n$ $I_\beta = I_n$ d) $I_\alpha = 0$ $I_\beta = I_n$

Fig. 1 : Structure complète et principe de fonctionnement du moteur pas à pas

1.2 Mode d'alimentation

Comme nous venons de le voir précédemment, la rotation du moteur s'effectue en alimentant les bobines du stator par permutation circulaire en suivant une configuration bien particulière. Ces différentes configurations définissent les modes d'alimentation [Abignoli-90]. Lors des simulations et des essais, notre moteur est alimenté en mode quatre, c'est-à-dire que le courant est multiplié par un facteur $\sqrt{2}$ lorsqu'une seule phase est alimentée, afin de permettre un fonctionnement régulier au niveau du couple (figure 2).

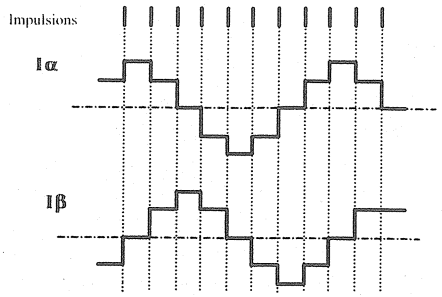


Fig. 2 : Alimentation en mode 4

1.3 Profil de déplacement

Dans le milieu industriel, le moteur pas à pas est commandé en fréquence, c'est-à-dire que la commande appliquée au séquenceur est constituée d'un train d'impulsions. La vitesse du rotor est alors égale à la fréquence de ce train d'impulsions. Aussi, cette fréquence n'est pas constante, ce qui permet de générer des phases d'accélération et de décélération. Pour les applications nécessitant une grande précision, notamment en robotique et en commande numérique de machines-outils, les phases d'accélération et de décélération sont linéaires, ce qui correspond à un profil de déplacement trapézoïdal. Cette technique est appréciée car elle permet d'obtenir des fréquences de rotation élevées en employant un algorithme de calcul des durées des pas relativement simple. En effet, le calculateur ne calcule pas la fréquence des impulsions mais la durée entre deux impulsions successives qui s'exprime de la manière suivante [Deloizy-86] :

$$\begin{aligned} t_{i+1} &= t_i \,.\, (1\text{-}a.t_i^2) && \text{lors de la phase d'accélération} \\ t_{i+1} &= t_i = 1/F_f && \text{lors du palier à vitesse constante} \\ t_{i+1} &= t_i \,.\, (1\text{+}d.t_i^2) && \text{lors de la phase de décélération} \end{aligned} \qquad (1)$$

où t_i et t_{i+1} sont respectivement les durées des pas i et i+1, a est la constante d'accélération (en pas/s²), d est la constante de décélération (en pas/s²) et F_f est la fréquence du palier (en pas/s). La durée entre les deux premières impulsions de commande est fixée à $1/F_0$ où F_0 est la fréquence de démarrage qui doit être contenue dans la zone de *start-stop* bien connue des spécialistes du moteur pas à pas. Avec cet algorithme, la durée du pas i+1 est calculée pendant le franchissement du pas i. Ainsi, la commande n'est pas calculée à une période d'échantillonnage fixe mais à une période égale à la durée du pas.

1.4 Modélisation

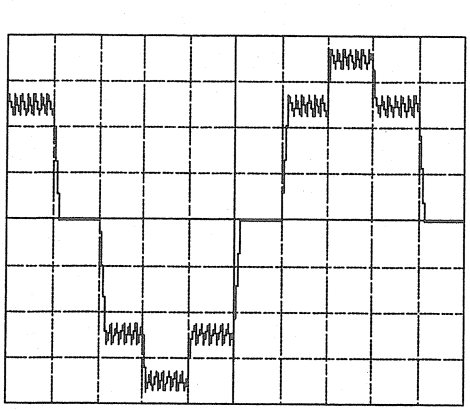
Afin de simuler les dynamiques du moteur pas à pas, nous utilisons les équations électromécaniques suivantes [Kuo-79][Goeldel-84] :

$$\begin{aligned} U_\alpha &= R.I_\alpha + L.\frac{d.I_\alpha}{dt} - K_e.\sin è.\frac{d\theta_m}{dt} \\ U_\beta &= R.I_\beta + L.\frac{d.I_\beta}{dt} + K_e.\cos è.\frac{d\theta_m}{dt} \\ C_m &= -K_h.(I_á.\sin è - I_â.\cos è) - K_d.\sin(4.è) \\ J.\frac{dÙ_m}{dt} &= C_m - F.Ù_m - C_c - C_r.\mathrm{sign}(Ù_m) \end{aligned} \qquad (2)$$

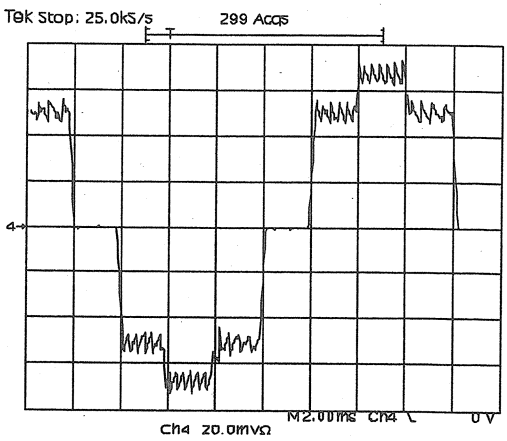
où I_α, I_β, U_α, U_β sont les courants et les tensions dans les phases α et β, L et R sont l'inductance et la résistance des enroulements des phases statoriques, K_h est le coefficient de couple hybride, K_d est le coefficient de couple de détente, K_e est le coefficient de force contre électromotrice, Z_r est le nombre de dents rotoriques, J est l'inertie, F est le coefficient de frottement visqueux, Ω_m est la vitesse du rotor, θ_m est la position mécanique du rotor, θ est la position électrique du rotor ($\theta = Z_r.\theta_m$), C_m est le couple moteur, C_c est le couple de charge et C_r le couple de frottement sec. Ces équations sont résolues à l'aide d'un algorithme de Runge Kutta d'ordre 4 programmé en langage C.

1.5 Validation du modèle

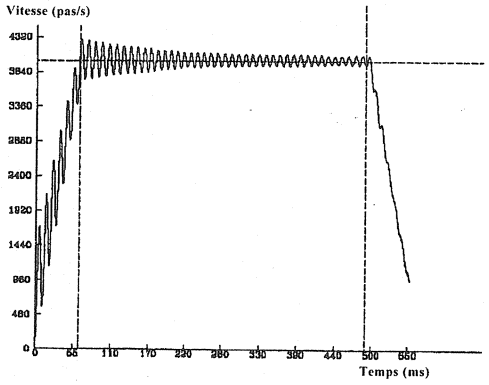
Les figures 3.a et 3.b représentent les courants simulés et expérimentaux dans une phase lorsqu'un train d'impulsions d'une fréquence de 500 Hz est injecté en entrée du séquenceur tandis que les figures 3.c et 3.d représentent les vitesses simulées et expérimentales lors d'un profil de déplacement trapézoïdal.



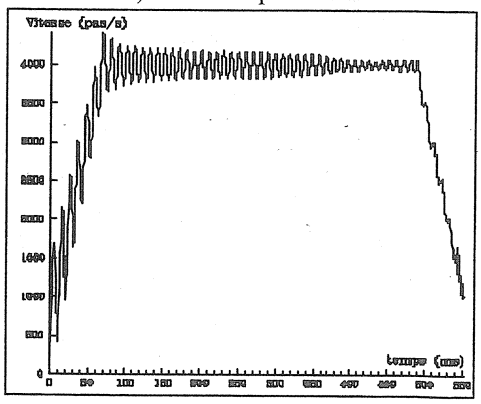
a) Courant simulé



b) Courant expérimental



c) Vitesse simulée



d) Vitesse expérimentale

Fig. 3 : Réponses simulées et expérimentales

Ces figures assurent la validation du modèle électrique et mécanique (2) et mettent en exergue les problèmes de suivi de trajectoire lorsque l'on utilise une commande en boucle ouverte (figures 3.c et 3.d).

2. Commande en boucle fermée

La commande en boucle fermée consiste à comparer la consigne avec la sortie du système afin de calculer la commande à imposer au système. Aussi, la vitesse de l'actionneur pas à pas sera mesurée à l'aide d'un codeur optique placé sur l'arbre moteur et l'algorithme de commande sera basé sur la théorie de la logique floue mis au point par Zadeh en 1965 [Zadeh-65]. La figure 4 représente le schéma de commande employé. Cette structure de régulation est connue sous le nom de « Feedforward » ou commande par anticipation. Le contrôleur flou n'intervient qu'en cas d'écart constaté entre la fréquence demandée et la vitesse effective.

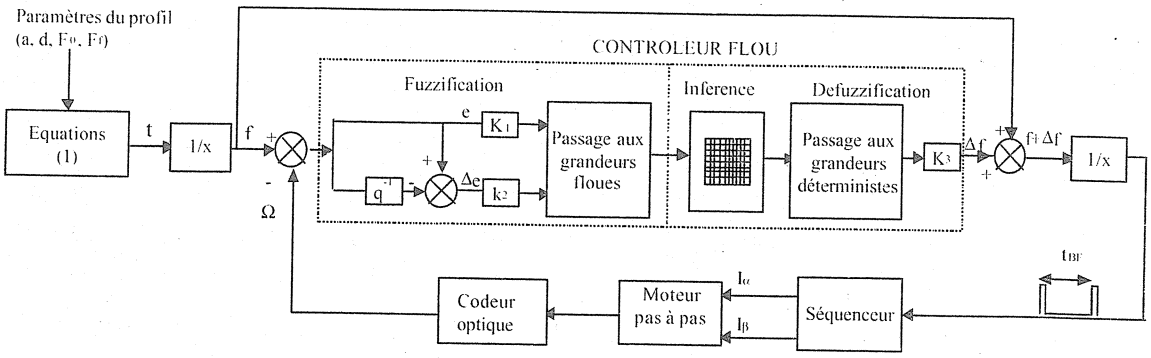


Fig. 4 : Schéma de commande en boucle fermée

A partir des équations (1), on obtient la durée des pas moteur t ainsi que la fréquence f de consigne. On soustrait à cette dernière la vitesse Ω du moteur pour obtenir l'erreur qui est l'unique entrée du régulateur flou que nous allons maintenant décrire.

2.1 Principe de la régulation floue

La particularité de cette technique vient du fait que, par opposition aux stratégies classiques de contrôle, la régulation par logique floue ne traite pas une relation mathématique bien définie mais utilise des inférences avec plusieurs règles fondées sur des variables linguistiques. Ces règles floues décrivent les observations et les réactions qu'aurait un opérateur humain lors du pilotage manuel du processus. On peut distinguer dans la structure d'un régulateur flou trois fonctions principales [Mandani-74]: la fuzzification, l'inférence et la défuzzification.

La fuzzification consiste en une projection des variables physiques réelles sur des ensembles flous. La première étape de la fuzzification consiste en une normalisation des grandeurs d'entrées sur un univers de discours à l'aide de *facteurs d'échelle*. Ensuite il faut réaliser le passage des grandeurs physiques normalisées en variables floues ; on utilise pour cela des *variables linguistiques* représentées par des *fonctions d'appartenance*. Ces variables linguistiques sont choisies de façon à modéliser les observations d'un être humain sur l'évolution d'un procédé. En effet, un opérateur utilisera des termes linguistiques plutôt que des valeurs numériques pour exprimer cette évolution. Celui-ci qualifiera l'erreur de positive, négative, nulle, petite, grande ou moyenne.

L'inférence lie les grandeurs d'entrée fuzzifiées à la variable de sortie exprimée elle aussi sous forme linguistique, selon un certain nombre de règles floues. Les règles floues sont exprimées à l'aide des termes SI et ALORS. Ceux-ci permettent de relier la (ou les) *condition(s)* à la *conclusion*. Si une règle comporte plusieurs conditions, celles-ci sont liées soit par l'opérateur ET soit par l'opérateur OU. Aussi existe-t-il plusieurs alternatives pour réaliser les opérateurs ET et OU. On a alors introduit la notion de *méthode d'inférence*. Celle-ci détermine la réalisation des différents opérateurs dans une inférence.

Enfin, la défuzzification consiste en une transformation de l'information floue issue de l'inférence en une information déterministe directement applicable au processus.

Dans la partie suivante, nous présentons l'algorithme de commande floue utilisé pour réguler la vitesse du moteur pas à pas ainsi que les méthodes employées pour déterminer les paramètres du correcteur flou. Ensuite, nous présentons les effets simulés de la régulation floue sur un profil de déplacement trapézoïdal.

2.2. Présentation de l'algorithme flou employé

Afin de réguler la vitesse de l'entraînement considéré, nous utilisons un correcteur flou dont les grandeurs d'entrée sont l'erreur et la variation de l'erreur notées respectivement e et Δe telles que :

$$e(k) = f(k) - \Omega(k) \qquad \Delta e(k) = e(k) - e(k-1) \qquad (3)$$

La sortie du contrôleur flou n'est pas la variable de contrôle mais son incrément. En effet la commande est obtenue en additionnant la sortie du contrôleur flou avec la consigne f. La commande floue proposée peut être alors assimilée à une action proportionnelle intégrale. La vitesse étant mesurée tous les pas angulaires, le système est donc échantillonné en position et non en temps. Par conséquent, l'indice k rencontré dans les équations (3) représente le $k^{ième}$ pas et non l'instant kT où T serait une période d'échantillonnage temporelle.

Les facteurs d'échelle K_1 et K_2 permettant la normalisation des deux grandeurs d'entrées ont été obtenus à l'aide de la

réponse dans le plan de phase (figure 5) lors d'essais en boucle ouverte. Ceux-ci sont en effet choisis de manière à ce que la trajectoire des grandeurs normalisées couvre tout le plan de phase formé par l'univers de discours.

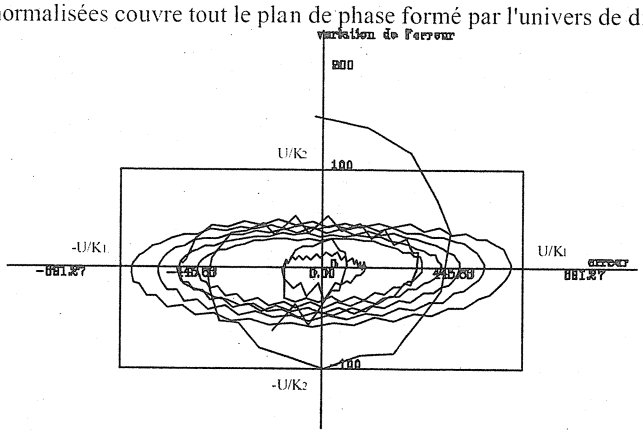


Fig. 5. Trajectoire dans le plan de phase

Afin de déterminer le nombre optimal de termes linguistiques à employer, nous avons simulé des correcteurs possédant plus ou moins de valeurs linguistiques [Betin-00]. Il s'est avéré de cette étude que, de trois à sept termes flous, l'ajout de deux nouvelles grandeurs linguistiques permet d'éliminer des oscillations. Par contre, au-dessus de sept termes, l'ajout de variables linguistiques ne permet plus d'éliminer d'oscillations mais permet seulement de diminuer légèrement l'amplitude de l'oscillation restante. Nous avons donc retenu pour notre correcteur une configuration contenant sept termes linguistiques : GN, MN, PN, ZE, PP, MP et GP (PN : petit négatif, GN : grand négatif....). Les fonctions d'appartenance de ces sept variables linguistiques sont symétriques par rapport à l'axe des ordonnées et présentent un taux de recouvrement de 50 % comme la plupart des contrôleurs flous rencontrés dans la littérature (voir figure 6).

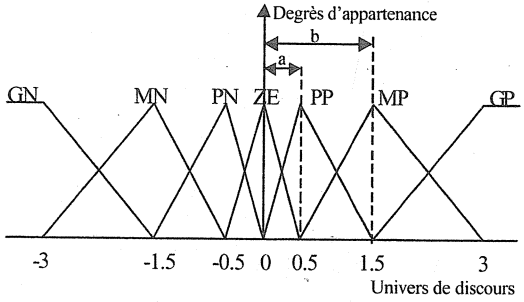


Fig. 6 : Configuration retenue

Les règles d'inférence utilisées sont dérivées de celles du mode glissant [Mac Vicar-76] réputé pour sa robustesse. En effet, on cherche à faire glisser le point de fonctionnement, représenté dans le plan de phase, le long d'une droite de manière à atteindre la consigne sans oscillation. Pour cela, on applique une commande positive d'un coté de la droite et négative de l'autre afin de ramener le point de fonctionnement vers la droite. La matrice d'inférence de ces règles est représentée en figure 7. Ces règles d'inférence sont symétriques et complètes (toutes les positions de la matrice sont occupées). Ces règles sont associées à la méthode d'inférence max-min bien connue dans la littérature pour son caractère non linéaire [Buhler-94].

A partir de la sortie exprimée sous forme floue, nous employons la méthode du centre de gravité afin d'obtenir une grandeur déterministe. Pour des fonctions d'appartenance discrètes, la sortie u obtenue à l'aide de la méthode du centre de gravité s'exprime de la manière suivante :

$$u = \frac{\sum_{i=1}^{i=n} ui \cdot \mu(ui)}{\sum_{i=1}^{i=n} \mu(ui)} \qquad (4)$$

où n est le nombre d'ensemble flous utilisés pour la sortie, u_i est le centre du $i^{ème}$ ensemble flou et $\mu(u_i)$ est la valeur d'appartenance associée. Cette technique présente l'avantage de prendre en compte toutes les règles contrairement à la méthode du maximum.

e \ Δe	*GN*	*MN*	*PN*	*EZ*	*PP*	*MP*	*GP*
GP	EZ	PP	MP	GP	GP	GP	GP
MP	PN	EZ	PP	MP	GP	GP	GP
PP	MN	PN	EZ	PP	MP	GP	GP
EZ	GN	MN	PN	EZ	PP	MP	GP
PN	GN	GN	MN	PN	EZ	PP	MP
MN	GN	GN	GN	MN	PN	EZ	PP
GN	GN	GN	GN	GN	MN	PN	EZ

Fig. 7 : Matrice d'inférence utilisée

Enfin pour conclure sur cette partie consacrée à la détermination du régulateur flou, la figure 8 représente la sortie normalisée Δf_n du correcteur flou proposé en fonction de l'erreur normalisée e_n et de la variation de l'erreur normalisée Δe_n. Cette figure met en exergue le caractère non linéaire du régulateur flou.

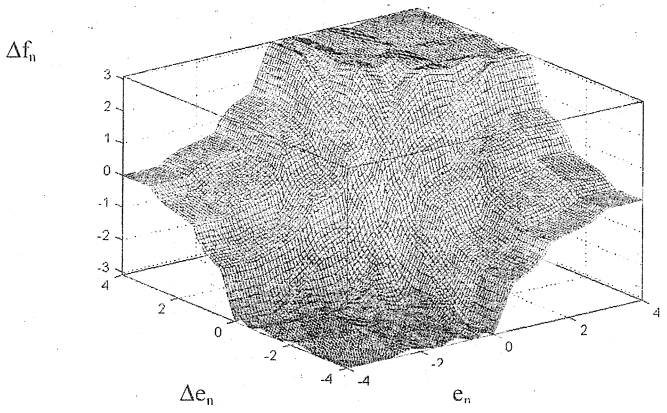


Fig. 8 : Caractéristique du régulateur flou

2.3 Simulation

Dans cette partie, l'algorithme de commande floue présenté précédemment est maintenant simulé sur un profil de déplacement trapézoïdal. La figure 9 représente la réponse avec le régulateur flou lors du profil de déplacement déjà présenté en boucle ouverte (figure 3.c). En comparant les réponses en boucle ouverte et avec le régulateur flou, on voit nettement que le correcteur flou permet d'éliminer très rapidement les oscillations sur le régime transitoire et sur le régime permanent. On peut donc considérer que le correcteur flou présente de bonnes propriétés en poursuite.

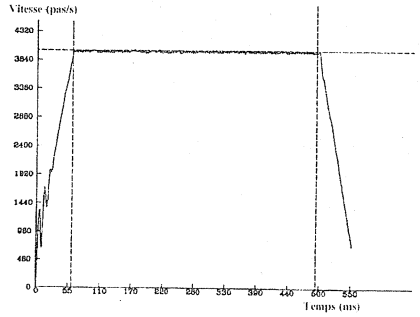


Fig. 9 : Réponse simulée avec le régulateur flou

3. Implantation de la commande en boucle fermée

L'implantation d'une commande en boucle fermée pour un moteur pas à pas est difficilement envisageable sur un micro-ordinateur classique du fait de la rapidité de l'actionneur. En effet, à la vitesse de 4000 pas/s, le calculateur ne dispose que de 250 µs pour déterminer la commande floue. Nous nous sommes donc orientés vers un système minimal à environnement de développement croisé, à savoir le microcontrôleur INTEL 80C196Kc. Ce composant bon marché, possédant 16 bits de données et cadencé à 16 MHz, a été principalement choisi pour sa structure HSO (High Speed Output) qui permet de matérialiser très rapidement des signaux de commande.

3.1 Présentation du 80C196kc

Pour concevoir et tester les logiciels, l'utilisateur dispose d'une carte d'évaluation du 80C196Kc, d'un assembleur, d'un compilateur C, d'un éditeur des liens, d'un désassembleur.
Les programmes sont écrits en langage C ou en assembleur sur un micro-ordinateur du type PC-AT. L'utilisation du langage C pour l'écriture du code simplifie cette étape, cependant, quelques routines seront écrites en assembleur de manière à optimiser le temps d'exécution. En effet, seule l'écriture en langage machine permet d'obtenir un code minimum. En conséquence, nous utiliserons le langage C pour les routines de bas niveaux et l'assembleur pour les interruptions et les calculs. Une fois les programmes compilés, ceux-ci sont téléchargés vers la carte d'évaluation par la liaison série. Une deuxième liaison série nous permet de recueillir les données telles que la vitesse ou la commande qui sont stockées dans la RAM du 80C196Kc lors du fonctionnement. Après être transférées, ces données sont traitées et les réponses peuvent être visualisées sur un deuxième micro-ordinateur.
La partie logicielle implantée sur le microcontrôleur réalise le calcul en temps réel de la durée des pas (consigne), la détermination de la vitesse de rotation du moteur, le calcul en temps réel de la commande selon l'algorithme de la régulation floue et la génération des impulsions de commande. Nous allons maintenant détailler ces quatre tâches réalisées par le microcontrôleur.

3.2 Calcul en temps réel de la consigne

Le profil de déplacement étant du type trapézoïdal, l'équation (1) est employée pour calculer la durée des pas moteurs. Aussi, cette équation peut être calculée soit en temps différé soit en temps réel. Pour le calcul en temps différé, les durées des pas moteurs sont précalculées et stockées sous forme de table dans la mémoire du système. Lors de l'exécution de la routine d'interruption, la durée du pas est alors chargée dans le timer. Cette technique permet d'alléger la routine d'interruption puisque le calcul de la durée des pas moteur est effectué hors ligne mais requiert une place importante augmentant avec le nombre de pas. La taille mémoire du 80C196Kc étant réduite (232 octets), l'emploi de cette technique nécessiterait l'ajout de boîtiers RAM. Aussi avons-nous préféré employer le calcul en temps réel. Néanmoins, cette seconde méthode pose des problèmes d'un autre ordre. En effet, il faut calculer pendant l'exécution d'un pas en cours la durée du pas suivant. Aussi, lorsque la vitesse de rotation croît, le temps autorisé pour le calcul diminue. Il faut alors optimiser la séquence de calcul afin de minimiser son temps. Pour cela, cette routine de calcul a directement été écrite en langage machine sur le microcontrôleur. Le codage de la durée du pas i (t_i) étant effectué par le timer 1 cadencé à 1 MHz (F_c = 1 MHz), la durée t_i est donc représentée par le nombre N_i ($N_i = t_i.F_c$) et l'expression à calculer lors de l'accélération devient alors :

$$N_{i+1} = N_i(1 - a \cdot N_i^2 / F_c^2) \qquad (5)$$

Ainsi, le calcul de la durée d'un pas moteur lors de l'accélération en utilisant l'équation (5) s'effectue en 13 µs sur le microcontrôleur considéré.

3.3 Détermination de la vitesse de rotation

Dans la littérature, on distingue cinq méthodes permettant la détermination de la vitesse de rotation à partir des signaux logiques émis par le codeur : par moyennage, par comptage, par mesure du temps, par comparaison de fréquence ou par comparaison de phase. Le choix de la méthode est directement lié au procédé et au type de commande. Ainsi pour notre actionneur, la détermination de la vitesse de rotation par mesure du temps semble être la plus judicieuse. En effet, le moteur pas à pas n'étant pas commandé à une fréquence fxe mais à une fréquence égale à la fréquence de rotation, il est nécessaire de calculer la vitesse à chaque pas moteur.
Le principe de cette méthode est élémentaire. Un timer délivre des impulsions à fréquence de répétition fixe F. Un compteur dénombre ces impulsions pendant le temps qui sépare deux impulsions fournies par le codeur optique. Le

nombre N obtenu est proportionnel au temps T_c séparant ces deux impulsions du codeur, donc inversement proportionnel à la vitesse du moteur. La vitesse s'exprime alors de la manière suivante où R_m est la résolution du moteur et R_c est la résolution du codeur.

$$\Omega(\text{pas/s}) = \frac{1}{N} \cdot F_c \cdot \frac{R_m}{R_c} \qquad (6)$$

Pour l'application considérée, nous employons le timer 2 programmé en mode rapide délivrant des impulsions à 8MHz et la valeur de N sera obtenue à chaque front montant de la phase A du codeur optique à l'aide d'un registre programmé en mode capture.

3.4 Calcul de la commande selon l'algorithme flou

Afin de simplifier le programme de contrôle et d'écourter le temps de calcul de la commande floue, nous utilisons une table de décision fixe. Celle-ci contient les valeurs que peut prendre la sortie pour les différentes combinaisons des signaux d'entrée [Lee-90]. Ainsi, on ne recalcule pas la sortie du correcteur à chaque pas d'échantillonnage mais on cherche dans la table de décision la sortie correspondant aux conditions d'entrées les plus proches de notre point de fonctionnement. Aussi, le nombre de niveaux de quantification, qui dépend bien évidemment de la taille mémoire du microcontrôleur utilisé, doit être déterminé soigneusement pour ne pas détériorer les performances du contrôle flou. En effet, une quantification trop grossière engendre une perte de précision de la commande et une quantification trop fine nécessite un encombrement mémoire important.

3.5 Génération des impulsions de commande

Pour matérialiser les impulsions de commande, on utilise une sortie HSO du 80C196Kc programmé en interruption. Le principe, illustré figure 10, consiste à insérer successivement des 0 et des 1 dans le registre de commande de la ligne HSO (hso_command) et de charger la durée de l'impulsion et la durée du pas dans le registre de temporisation (hso_time). Pour boucler le système, le registre d'interruption de la ligne hso est programmé de manière à ce qu'une interruption survienne à chaque front descendant de la ligne HSO.

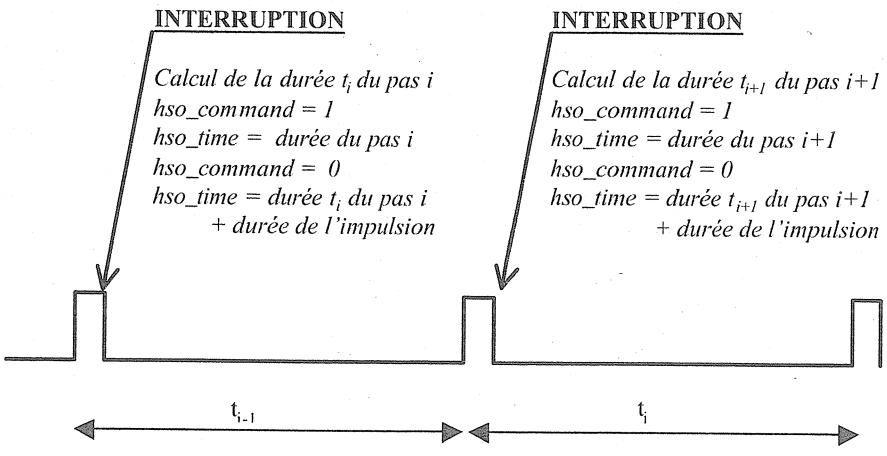


Fig. 10 : Génération du signal de commande

3.6 Résultats expérimentaux

Afin de valider expérimentalement les performances en poursuite de notre correcteur flou, nous utilisons le même profil de déplacement que celui employé lors de la simulation (figure 9). On peut remarquer en comparant les figures 9 et 11 que les résultats expérimentaux concordent parfaitement avec les résultats obtenus en simulation. En effet les oscillations sont très rapidement éliminées avec le contrôleur flou. Seules deux oscillations subsistent sur la phase d'accélération ; celles-ci sont dues au fait que le moteur est démarré en boucle ouverte. En effet, la première commande est égale à $1/F_0$.

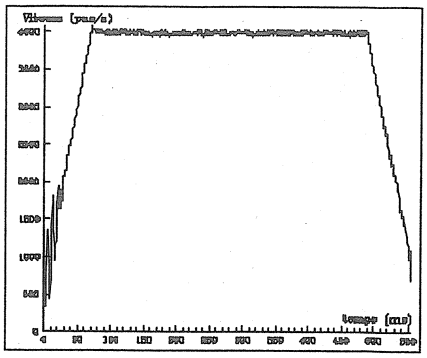


Fig. 11 : Réponse expérimentale avec correcteur flou

Conclusion

Dans cet article, une commande en boucle fermée pour moteur pas à pas est dans un premier temps simulée puis implantée en temps réel. Les résultats obtenus expérimentalement confirment ceux obtenus en simulation. En effet, les oscillations présentes en boucle ouverte sur le palier à vitesse constante et sur les phases d'accélération et de décélération sont complètement éliminées avec la commande en boucle fermée.

Aussi, outre les remarquables performances de cette commande en boucle fermée, l'atout majeur de cet algorithme de commande vient du fait qu'il ne nécessite que très peu de manipulations mathématiques. En effet, en utilisant une table de décision fixe et calculée hors ligne, le calcul de la consigne, la détermination de la vitesse, le calcul de la commande et la matérialisation du signal de commande ne requièrent que 150 µs sur le microcontrôleur Intel 80C196kc dont la structure est peu propice aux calculs.

Aussi, une étude a permis de mettre en exergue la robustesse de cette commande en boucle fermée lorsque la configuration mécanique de la charge entraînée varie [Betin-99].

Cet algorithme a été également testé sur différents moteurs pas à pas. Les résultats se sont avérés tout aussi satisfaisants quelle que soit la puissance du moteur. Aussi, pour des applications liées à la robotique ou la commande numérique de machine outils, cette commande en boucle fermée pourrait être directement intégrée sur l'actionneur pas à pas sans surcoût puisqu'un seul microcontrôleur bon marché est suffisant. Une étude a d'ailleurs démontré que l'emploi d'un codeur optique présentant uniquement 400 impulsions par tour permettait d'obtenir des résultats fort satisfaisants.

Bibliographie

[Abignoli-90] M. ABIGNOLI, C. GOELDEL, Moteurs pas à pas, Technique de l'ingénieur, D 3960, pages 1 à 21, 1990.

[Betin-99] F. BETIN, M. DELOIZY, D. PINCHON, C. GOELDEL, "Régulation par logique floue de la vitesse d'un entraînement à moteur pas àpas", Revue Internationale de Génie Electrique, Editions Hermès, vol. 2, n°4/1999, pp. 439-463

[Betin-00] F. BETIN, D. PINCHON, G.A CAPOLINO, "Fuzzy logic applied to speed control of a stepping motor drive", IEEE Transactions on Industrial Electronics, vol. 47, n°3, June 200, pp. 610-622.

[Buhler-94] H. R. BUHLER, "Réglage par logique floue", Presses Polytechniques et Universitaires Romandes, 1994.

[Deloizy-86] M. DELOIZY, "Moteur pas à pas: Régimes transitoires linéaires et exponentiels", Rapport de DEA, LAM faculté des sciences de Reims,1986.

[Goeldel-84] C. GOELDEL, "Contribution à la modélisation, à l'alimentation et à la commande des moteurs pas à pas.", Thèse de doctorat ès Sciences, mars 1984, Institut National Polytechnique de Lorraine, Nancy.

[Kuo-79] B. C. KUO,"Step motors and control systems SRL", Publishing Company, Urbana-Champaign, Illinois, 1979.

[Lee-90] C. C. LEE, "Fuzzy logic in control systems: Fuzzy logic controller-Part 1", IEEE transactions on systems, man and cybernetics, vol. 20, n° 2, April 1990.

[Mac vicar-76] P. J. Mac Vivar Whelan, "Fuzzy sets for man-machine interaction", In. J. Man-Machine Studies, Vol. 8, pp. 687-697, 1976.

[Mandani-74] E. MANDANI, "Application of fuzzy algoriyhms for control of a single dynamic plant", Proc. of IEEE, vol. 121, pp 1585-88, 1974.

[Zadeh-65] L. A. ZADEH, "Fuzzy Sets", Information and Control, vol. 8, pp. 338-353, 1965.

Annexe

Les paramètres du moteur pas à pas (Stebon SDT1101-350-60) sont les suivants :

Courant nominal dans une phase	I = 5.6 A
Tension nominale aux bornes d'une phase	U = 91 V
Inductance d'une phase	L = 4.6 mH
Résistance d'une phase	R = 0.28 Ω
Coefficient de couple moteur	K_h = 0.464 N.m/A
Coefficient de couple de détente	K_d = 0.12 Nm
Coefficient de force électromotrice	Ke = 0.464 N.m/A
Nombre de dents rotoriques	Zr = 50
Inertie du rotor	J = 3.65 10^{-4} kg.m^2
Coefficient de frottement visqueux	F = 0.011 N.m.s/rad

LA PUISSANCE REACTIVE INSTANTANEE

Georges MANESSE
Professeur des Universités au Conservatoire National des Arts et Métiers
292, rue Saint-Martin – 75141 Paris Cedex 03 – Laboratoire d'Electricité Industrielle

Résumé :
Un concept efficace pour dimensionner un filtre actif parallèle triphasé sur un réseau de distribution, trois fils, déséquilibré en courant.

Cet article propose une méthode de calcul qui, lorsque tensions et courants d'un réseau triphasé sont sinusoïdaux, répond au problème classique de l'équilibrage en courant à facteur de puissance unitaire. Il s'appuie sur la définition d'une puissance instantanée complexe dont les parties réelles et imaginaires représentent respectivement la puissance instantanée délivrée par la source et la puissance réactive instantanée consommée par la charge. Dans le cas général, la source de tension est caractérisée par ses composantes symétriques et la charge fixe les composantes des courants en ligne. Des formules simples fournissent l'amplitude et la phase des courants absorbés par un filtre actif pour que le réseau délivre des courants en ligne en phase avec ses tensions simples de manière à former deux systèmes de tensions et des courants parfaitement équilibrés.

I. PUISSANCES ACTIVE ET REACTIVE INSTANTANEES EN MONOPHASE

I.1. Calcul direct

La puissance instantanée s'obtient en formant le produit de la tension v(t) aux bornes d'un récepteur consommateur d'énergie par le courant i(t) qui le traverse au même instant :

$$p = v \cdot i$$

p s'exprime en Watts (W), v en volts (V) et i en Ampères (A).
La puissance réactive instantanée est donnée par une formule analogue qui substitue à v(t) la tension retardée d'un quart de période $v(t - \frac{T}{4})$.

$$q(t) = v(t - \frac{T}{4}) \cdot i(t)$$

q s'exprime en vars (voltampères réactifs), v en volts (V) et i en Ampères (A)
Lorsque v et i sont de forme sinusoïdale, en choisissant une origine des temps telle que $v(t=0) = V\sqrt{2}$
il vient :

$$\begin{cases} v(t) = V\sqrt{2} \cdot \cos(\omega t) \\ i(t) = I\sqrt{2} \cdot \cos(\omega t - \varphi) \\ v(t - \frac{T}{4}) = V\sqrt{2} \cdot \sin(\omega t) \\ p(t) = 2VI \cdot \cos(\omega t) \cdot \cos(\omega t - \varphi) \\ q(t) = 2VI \cdot \sin(\omega t) \cdot \cos(\omega t - \varphi) \end{cases}$$

Dans ces conditions, et avec l'origine des temps choisie, les expressions de p(t) et q(t) présentent deux formes semblables qui se déduisent l'une de l'autre par substitution du sinus au cosinus :

$$\begin{aligned} p(t) &= VI \cdot \cos\varphi + VI \cdot \cos(2\omega t - \varphi) \\ q(t) &= VI \cdot \sin\varphi + VI \cdot \sin(2\omega t - \varphi) \end{aligned}$$

La quantité $p + jq = \underline{p}$ s'obtient directement en appliquant les formules d'Euler :

$$\underline{p}(t) = VI \cdot e^{j\varphi} + VI \cdot e^{j(2\omega t - \varphi)}$$

Cette quantité complexe est alors par définition la puissance instantanée complexe dont les parties réelle et imaginaire sont la puissance instantanée de valeur moyenne P et la puissance réactive instantanée de valeur moyenne Q.

I.2. Plan (p, q)

En construisant dans le plan (p, q) la trajectoire suivie par les points de coordonnées p et q, on peut voir l'évolution des puissances instantanées active et réactive en fonction du temps.

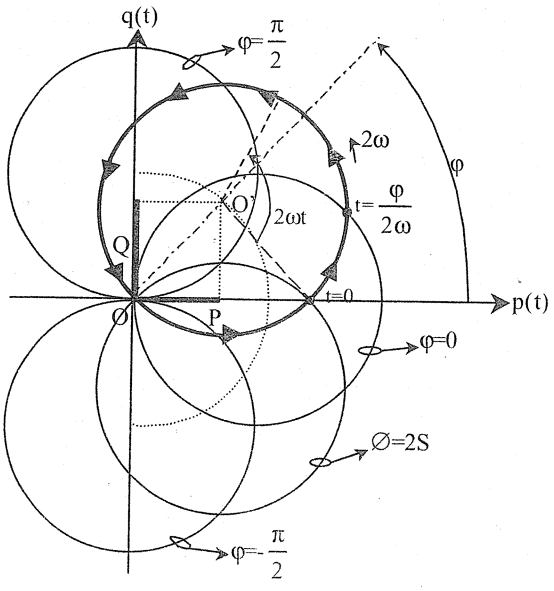
On vérifie en effet à tout instant la relation :

$$\boxed{(p - VI\cdot\cos\varphi)^2 + (q - VI\cdot\sin\varphi)^2 = (VI)^2}$$

Soit, en introduisant la puissance moyenne ou active P=VI cosφ, la puissance réactive Q=VI sinφ, et la puissance apparente S=VI

$$\boxed{(p-P)^2 + (q-Q)^2 = S^2}$$

Quelle que soit la valeur de φ, à V et I constants, la trajectoire obtenue est un cercle de rayon S décrit à la vitesse angulaire 2ω dans le sens trigonométrique, qui passe toujours par l'origine, et dont le centre O' a pour coordonnées P et Q.



Pour une installation caractérisée par la valeur de S, la puissance active maximale est obtenue lorsque φ = 0 donc à cosφ = 1 : on a alors P = S.

I.3. Puissance instantanée complexe

Les notations complexes évitent un traitement algébrique direct des lignes trigonométriques en substituant aux grandeurs instantanées v, i, p et q soit des fonctions du temps cissoïdales (en $e^{j\omega t}$), soit des grandeurs complexes normalisées sous la forme d'amplitudes complexes de tensions et courants $(\underline{V},\underline{I})$ selon les conventions en usage en Electrotechnique.

A partir de :

$$\begin{cases} v(t) = V\sqrt{2}\cdot\cos(\omega t + \psi) \\ i(t) = I\sqrt{2}\cdot\cos(\omega t + \psi - \varphi), \end{cases}$$

On pose :

$$\begin{cases} \underline{v}(t) = V\sqrt{2}\cdot e^{j(\omega t+\psi)} \\ \underline{i}(t) = I\sqrt{2}\cdot e^{j(\omega t+\psi-\varphi)}, \end{cases}$$

Soit :

$$\begin{cases} \underline{v}(t) = \underline{V}\sqrt{2}\cdot e^{j(\omega t)} \\ \underline{i}(t) = \underline{I}\sqrt{2}\cdot e^{j(\omega t)} \end{cases}$$

Lorsque :

$$\begin{cases} \underline{V} = V\cdot e^{j\psi} \\ \underline{I} = I\cdot e^{j(\psi-\varphi)}. \end{cases}$$

L'angle ψ est donné par le choix de l'origine des temps, tandis que l'angle φ désigne le déphasage arrière du courant sur la tension.

Les puissances instantanées active et réactive s'écrivent alors :

$$\begin{cases} p(t) = \frac{1}{2}\Re_e(\underline{v}\cdot\underline{i}^* + \underline{v}\cdot\underline{i}) \\ q(t) = \frac{1}{2}\Re_e(\underline{v}(t-\frac{T}{4})\cdot\underline{i}^* + \underline{v}(t-\frac{T}{4})\cdot\underline{i}) \end{cases}$$

Or

$$v(t-\frac{T}{4}) = \underline{V}\sqrt{2}\cdot e^{j(\omega t-\frac{\pi}{2})} = -j\underline{V}\sqrt{2}\cdot e^{j\omega t}$$

D'où

$$\begin{cases} p(t) = \Re_e(\underline{V}\cdot\underline{I}^* + \underline{V}\cdot\underline{I}\cdot e^{j2\omega t}) \\ q(t) = I_m(\underline{V}\cdot\underline{I}^* + \underline{V}\cdot\underline{I}\cdot e^{j2\omega t}) = \Re_e(-j\underline{V}\cdot\underline{I}^* - j\underline{V}\cdot\underline{I}\cdot e^{j2\omega t}) \end{cases}$$

On aboutit, dans ces conditions, à la définition d'une puissance instantanée complexe $\underline{p}$ dont la partie réelle et la partie imaginaire sont respectivement la puissance instantanée p(t) = vi et la puissance réactive instantanée $q(t) = v(t-\frac{T}{4})i$

$$\boxed{\underline{p}(t) = p + jq = \underline{V}\cdot\underline{I}^* + \underline{V}\cdot\underline{I}\cdot e^{j2\omega t}}$$

Le plan (p, q) est donc le plan complexe, ce qui justifie pour la puissance réactive instantanée l'appellation de « puissance imaginaire instantanée ».

On retrouve évidemment les résultats précédents établis avec $\psi = 0$

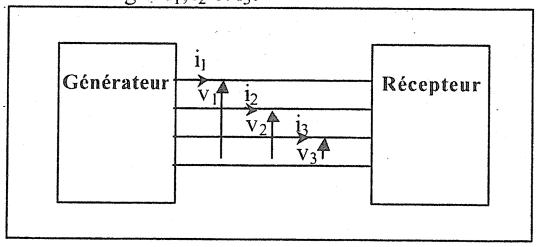
$$\begin{cases} p(t) = VI \cdot \cos\varphi + VI \cdot \cos(2\omega t - \varphi) \\ q(t) = VI \cdot \sin\varphi + VI \cdot \sin(2\omega t - \varphi) \end{cases}$$

On dispose maintenant d'un outil de traitement des déséquilibres par la méthode des composantes symétriques qui sont, comme chacun sait, des amplitudes complexes associées à des grandeurs sinusoïdales.

II. PUISSANCES ACTIVE ET REACTIVE INSTANTANEES EN TRIPHASE

II.1. Puissance réactive instantanée

Plaçons nous dans le cas général d'une distribution triphasée à quatre fils (trois phases et un neutre) caractérisée par trois tensions simples v_1, v_2, v_3 et trois courants en ligne i_1, i_2 et i_3.



Notons $[v_{123}]$ la matrice colonne $[v_1, v_2, v_3]^t$ et $[i_{123}]=[i_1, i_2, i_3]^t$.

La puissance instantanée consommée par le récepteur s'écrit :

$$p = v_1 \cdot i_1 + v_2 \cdot i_2 + v_3 \cdot i_3 = [v_{123}]^t \cdot [i_{123}]$$

et sous forme complexe :

$$\underline{p}(t) = \underline{S}_1 + \underline{S}_2 + \underline{S}_3 + (\underline{V}_1 \cdot \underline{I}_1 + \underline{V}_2 \cdot \underline{I}_2 + \underline{V}_3 \cdot \underline{I}_3) \cdot e^{j2\omega t}$$

ou

$$\underline{p}(t) = [\underline{V}_{123}]^t \cdot [\underline{I}^*_{123}] + [\underline{V}_{123}]^t \cdot [\underline{I}_{123}] \cdot e^{j2\omega t}$$

On définit alors la puissance réactive instantanée consommée par l'expression générale :

$$q = \frac{1}{\sqrt{3}}[(v_2 - v_3)i_1 + (v_3 - v_1)i_2 + (v_1 - v_2)i_3]$$

Remarque : Cette expression, qui rappelle le montage permettant la mesure de la puissance réactive au moyen de trois wattmètres, ne recouvre pas dans tous les cas de figure la définition de q donnée en monophasé. Elle est cependant bien adaptée aux réseaux de distribution comportant impérativement trois fils et offre l'avantage de ne pas demander d'opérateurs de décalage ou de déphasage ni de référence de potentiel pour être matérialisée en temps réel.

Lorsque le système des tensions simples est sinusoïdal et équilibré, on observe l'équivalence de la formule proposée à l'expression :

$$q(t) = v_1(t - \frac{T}{4})i_1 + v_2(t - \frac{T}{4})i_2 + v_3(t - \frac{T}{4})i_3$$

II.2. Calcul de la puissance instantanée

Si le lien énergétique générateur – récepteur est caractérisé par ses composantes symétriques en tension et en courant, il convient, dans l'expression de p(t), d'introduire les formules de passage définies par la matrice de Fortescue d'ordre 3 :

$$[F_3] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & a^2 & a \\ 1 & a & a^2 \end{bmatrix} \quad \text{avec } a = e^{j\frac{2\pi}{3}}$$

On obtient alors :

$$[\underline{V}_{123}] = [F_3] \cdot [\underline{V}_{hdi}] \text{ et } [\underline{I}_{123}] = [F_3] \cdot [\underline{I}_{hdi}]$$

avec $[\underline{V}_{hdi}] = [\underline{V}_h, \underline{V}_d, \underline{V}_i]^t$ et $[\underline{I}_{hdi}] = [\underline{I}_h, \underline{I}_d, \underline{I}_i]^t$

L'expression :

$$\underline{p}(t) = [\underline{V}_{123}]^t \cdot [\underline{I}^*_{123}] + [\underline{V}_{123}]^t \cdot [\underline{I}_{123}] \cdot e^{j2\omega t}$$

devient :

$$\underline{p}(t) = [\underline{V}_{hdi}]^t \cdot [F_3]^t \cdot [F_3^*] \cdot [\underline{I}^*_{hdi}] + [\underline{V}_{hdi}]^t \cdot [F_3]^t \cdot [F_3] \cdot [\underline{I}_{hdi}] \cdot e^{j2\omega t}$$

or $[F_3]^t[F_3^*] = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}$ et $[F_3]^t[F_3] = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 3 & 0 \end{bmatrix}$

donc :

$$\underline{p}(t) = 3(\underline{V}_h \underline{I}^*_h + \underline{V}_d \underline{I}^*_d + \underline{V}_i \underline{I}^*_i) + 3(\underline{V}_h \underline{I}_h + \underline{V}_d \underline{I}_i + \underline{V}_i \underline{I}_d) e^{j2\omega t}$$

Dans le cas le plus fréquent d'un réseau équilibré en tension qui alimente une charge déséquilibrée en courant, on a $\underline{V}_h = \underline{V}_i = \underline{0}$

et :

$$\underline{p}(t) = 3\underline{V}_d \cdot \underline{I}^*_d + 3\underline{V}_d \cdot \underline{I}_i \cdot e^{j2\omega t}$$

On retrouve le résultat classique selon lequel le déséquilibre en courant provoqué par la charge se traduit par un terme de puissance fluctuante dont l'amplitude est égale au produit $3V_dI_i$.

II.3. Calcul de la puissance réactive instantanée

L'expression de définition de q(t) peut se mettre sous la forme matricielle suivante :

$$q(t)=\frac{1}{\sqrt{3}}[v_{123}]^t\cdot\begin{bmatrix}0&-1&1\\1&0&-1\\-1&1&0\end{bmatrix}\cdot[i_{123}]$$

Il en résulte l'expression de la puissance réactive instantanée complexe :

$$\underline{q}(t)=\frac{1}{\sqrt{3}}\left([\underline{V}_{123}]^t\cdot[K]\cdot[\underline{I}^*_{123}]+[\underline{V}_{123}]^t\cdot[K]\cdot[\underline{I}_{123}]\cdot e^{j2\omega t}\right)$$

dans laquelle :

$[K]=\begin{bmatrix}0&-1&1\\1&0&-1\\-1&1&0\end{bmatrix}$ est une matrice de couplage constante.

En introduisant les composantes symétriques, on obtient :

$$\underline{q}(t)=\left([\underline{V}_{hdi}]^t\cdot[F_3]^t\cdot[K]\cdot[F_3^*]\cdot[\underline{I}^*_{hdi}]+[\underline{V}_{hdi}]^t\cdot[F_3]^t\cdot[K]\cdot[F_3]\cdot[\underline{I}_{hdi}]\cdot e^{j2\omega t}\right)$$

Le calcul des produits de matrices intermédiaires conduit aux résultats suivants :

$$\frac{1}{\sqrt{3}}[F_3]^t\cdot[K]\cdot[F_3^*]=\begin{bmatrix}0&0&0\\0&-3j&0\\0&0&3j\end{bmatrix}$$

$$\frac{1}{\sqrt{3}}[F_3]^t\cdot[K]\cdot[F_3]=\begin{bmatrix}0&0&0\\0&0&-3j\\0&3j&0\end{bmatrix}$$

ce qui donne en développant la forme matricielle :

$$\boxed{\underline{q}(t)=3(-j\underline{V}_d\underline{I}_d^*+j\underline{V}_i\underline{I}_i^*)+3(-j\underline{V}_d\underline{I}_i+j\underline{V}_i\underline{I}_d)e^{j2\omega t}}$$

Pour un réseau équilibré en tension et déséquilibré en courant, on obtient :

$$\boxed{\underline{q}(t)=3j\cdot\underline{V}_d\cdot\underline{I}_d^*-3j\cdot\underline{V}_d\cdot\underline{I}_i\cdot e^{j2\omega t}}$$

On constate l'analogie de ce résultat avec celui obtenu en monophasé.

On vérifie alors que :

- Seule la composante directe du courant intervient dans le calcul des puissances moyennes active et réactive ;
- Seule la composante inverse du courant fixe l'amplitude de la puissance fluctuante.

On a en effet :

$$\begin{cases}p(t)=3\Re_e(\underline{V}_d\cdot\underline{I}_d^*+\underline{V}_d\cdot\underline{I}_i\cdot e^{j2\omega t})\\q(t)=3I_m(\underline{V}_d\cdot\underline{I}_d^*+\underline{V}_d\cdot\underline{I}_i\cdot e^{j2\omega t})\end{cases}$$

soit, en posant

$$\begin{cases}\underline{S}_d=S_d e^{j\varphi_d}\\Arg\underline{V}_d-Arg\underline{I}_i=\varphi_{di}\\Arg\underline{V}_d-Arg\underline{I}_d=\varphi_d\\v_1=V_d\sqrt{2}\cos\omega t\end{cases}$$

$$\boxed{\begin{aligned}p(t)&=3V_dI_d\cos\varphi_d+3V_dI_i\cos(2\omega t-\varphi_{di})\\q(t)&=3V_dI_d\sin\varphi_d+3V_dI_i\sin(2\omega t-\varphi_{di})\end{aligned}}$$

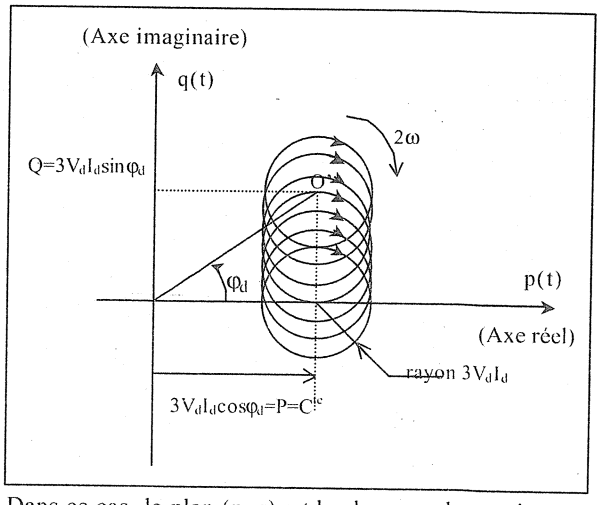
On retrouve une trajectoire analogue à celle obtenue en monophasé puisque :

$$\boxed{(p-P)^2+(q-Q)^2=(3V_dI_i)^2}$$

P et Q sont respectivement les valeurs moyennes de la puissance instantanée $\bar{p}$ et de la puissance réactive instantanée $\bar{q}$.

Le cercle, parcouru dans le même sens que précédemment à la vitesse angulaire 2ω, est encore centré sur O' de coordonnées $P=3V_dI_d\cos\varphi_d$ et $Q=3V_dI_d\sin\varphi_d$

Si l'on compense la puissance réactive totale à puissance moyenne P constante, on va se déplacer verticalement dans le plan (p,q). Lorsqu'on obtiendra la compensation de Q il restera à annuler le rayon du cercle en jouant sur I_i. On atteindra enfin un point d'ordonnée nulle et d'abscisse P pour lequel $\varphi_d=0$ et $I_i=0$.



Dans ce cas, le plan (p, q) est le plan complexe puisque $\underline{p}=p+jq$.

III. FILTRE PARALLELE DE COMPENSATION SIMULTANEE DU DESEQUILIBRE ET DU FACTEUR DE PUISSANCE D'UNE INSTALLATION TRIPHASEE

III.1. Position du problème

Considérons une installation alimentée par une source de tension équilibrée à trois conducteurs qui présente une composante directe $\underline{V}_d$ d'argument nul.

Les tensions simples instantanées s'écrivent alors :

$$[v_1,v_2,v_3]^t =$$

$$[V_d\sqrt{2}\cos\omega t,\ V_d\sqrt{2}\cos(\omega t-2\pi/3),\ V_d\sqrt{2}\cos(\omega t-4\pi/3)]$$

et les amplitudes complexes correspondantes :

$$[\underline{V}_1,\underline{V}_2,\underline{V}_3]^t = [\underline{V}_d\ ,\ a^2\underline{V}_d\ ,\ a\underline{V}_d]^t$$

Le système des courants absorbés par la charge est alors $[\underline{I}_{123}]$ caractérisé par les deux composantes directe et inverse $\underline{I}_d$ et $\underline{I}_i$ telles que :

$$\varphi_d = \arg \underline{V}_d - \arg \underline{I}_d$$

$$\varphi_{di} = \arg \underline{V}_d - \arg \underline{I}_i$$

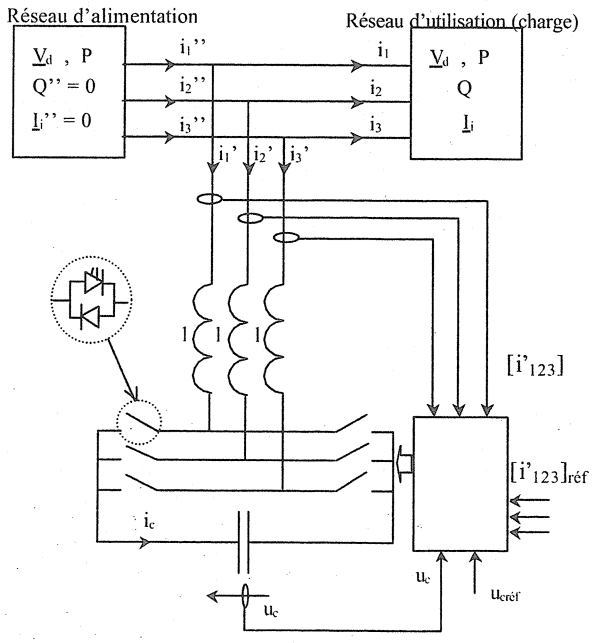
On a alors en fonction du temps $[i_1,i_2,i_3]^t$ avec :

$$i_1 = I_d\sqrt{2}\cos(\omega t-\varphi_d) + I_i\sqrt{2}\cos(\omega t-\varphi_{di})$$

$$i_2 = I_d\sqrt{2}\cos(\omega t-\varphi_d-2\pi/3) + I_i\sqrt{2}\cos(\omega t-\varphi_{di}-4\pi/3)$$

$$i_3 = I_d\sqrt{2}\cos(\omega t-\varphi_d-4\pi/3) + I_i\sqrt{2}\cos(\omega t-\varphi_{di}-2\pi/3)$$

En utilisant un filtre dont les courants d'entrée $[i'_1,i'_2,i'_3]^t$ sont asservis à un vecteur de référence $[i'_1,i'_2,i'_3]^t_{ref}$ on peut modifier les courants appelés au réseau $[i''_1,i''_2,i''_3]^t$ sans toucher à la charge.



Le problème consiste à déterminer $[i'_{123}]_{réf} = [i'_{123}]$ lorsque les courants absorbés par le filtre suivent parfaitement leurs consignes respectives avec les contraintes suivantes :

- la puissance active fournie par le réseau d'alimentation est exclusivement consommée par la charge P''=P
- si la charge consomme une puissance réactive Q, celle-ci est exclusivement fournie par le filtre actif et non par le réseau d'alimentation : Q'= -Q ou Q'' = 0
- le réseau d'alimentation est équilibré , d'où : $\underline{I}_i'' = 0 = \underline{I}_i + \underline{I}_i'$. La composante inverse du système des courants dans la charge est parfaitement compensée par le filtre et :

$$\boxed{\underline{I}_i' = -\underline{I}_i}$$

Il résulte de la contrainte Q''= 0 la relation $I_d'\sin\varphi_d' + I_d\sin\varphi_d = 0$, tandis que P'= 0 entraîne $I_d'\cos\varphi_d' = 0$. En regroupant ces deux expressions, on obtient :

$$\begin{cases}\varphi'_d = -\dfrac{\pi}{2}\,\mathrm{sign}(\varphi_d)\\ I'_d = |I_d\sin\varphi_d|\end{cases}$$

soit :

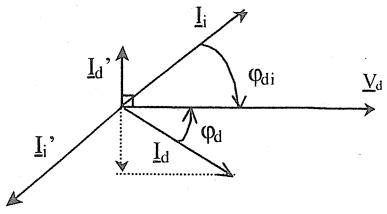
$$\boxed{\underline{I}'_d = j\frac{\underline{V}_d}{V_d}I_d\sin\varphi_d}$$

ce qui s'énonce :

La composante directe des courants absorbés par le filtre compense la composante directe des courants déwattés définis par la charge.

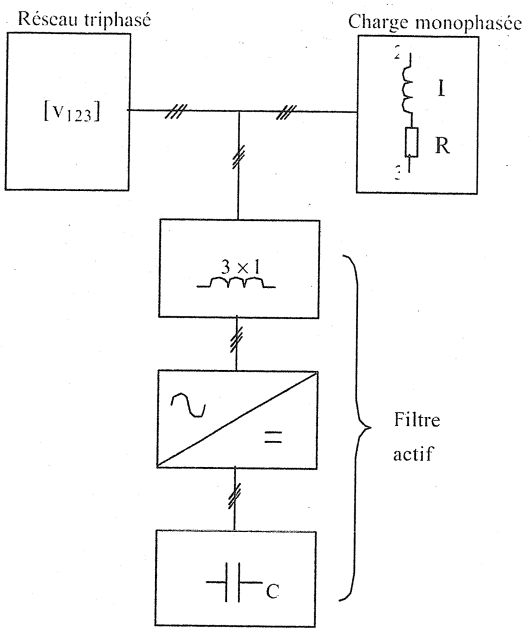
Les courants absorbés par le filtre sont alors donnés par les relations de Fortescue :

$$\begin{bmatrix}\underline{I}'_1\\ \underline{I}'_2\\ \underline{I}'_3\end{bmatrix} = \begin{bmatrix}1&1&1\\1&a^2&a\\1&a&a^2\end{bmatrix}\cdot\begin{bmatrix}\underline{I}'_h\\ \underline{I}'_d\\ \underline{I}'_i\end{bmatrix} = \begin{bmatrix}1&1&1\\1&a^2&a\\1&a&a^2\end{bmatrix}\cdot\begin{bmatrix}0\\ j\dfrac{\underline{V}_d}{V_d}I_d\sin\varphi_d\\ -\underline{I}_i\end{bmatrix}$$

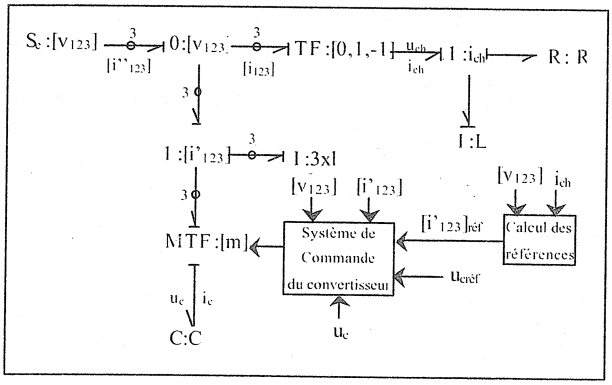


Equilibrage d'une charge inductive

Considérons un réseau triphasé associé à un filtre actif parallèle débitant sur une charge inductive monophasée branchée entre les phases 2 et 3 selon le schéma unifilaire ci-dessous qui traduit une approche essentiellement descriptive de l'objet industriel.



L'interconnexion des sources, les notations, les conventions de signe, ainsi que les orientations des flux d'informations apparaissent plus clairement sur le modèle dynamique instantané du système donné sous la forme d'un graphe de liens. On perçoit immédiatement la nécessité d'élaborer le vecteur de référence des courants absorbés par le filtre.



Caractérisons l'impédance de charge par son module et son argument:

$$\underline{Z}_c = R + jL\omega = R(1 + jtg\varphi) = \frac{R}{\cos\varphi}e^{j\varphi}$$

Exprimons les courants réseau créés par la charge:

$$\text{Si } \left[\underline{V}_1, \underline{V}_2, \underline{V}_3\right]^t = \left[\underline{V}_d, a^2\underline{V}_d, a\underline{V}_d\right]^t$$

$$\left[\underline{I}_1, \underline{I}_2, \underline{I}_3\right]^t = [0,1,-1]^t(a^2 - a)\frac{\underline{V}_d}{R}\cos\varphi \cdot e^{-j\varphi}$$

d'où l'on tire les composantes symétriques correspondantes;

$$\begin{bmatrix}\underline{I}_h\\ \underline{I}_d\\ \underline{I}_i\end{bmatrix} = \frac{1}{3}\begin{bmatrix}1 & 1 & 1\\ 1 & a & a^2\\ 1 & a^2 & a\end{bmatrix}\cdot\begin{bmatrix}0\\ 1\\ -1\end{bmatrix}(a^2 - a)\frac{\underline{V}_d}{R}\cos\varphi \cdot e^{-j\varphi}$$

$$\begin{bmatrix}\underline{I}_h\\ \underline{I}_d\\ \underline{I}_i\end{bmatrix} = \begin{bmatrix}0\\ 1\\ -1\end{bmatrix}\frac{\underline{V}_d}{R}\cos\varphi \cdot e^{-j\varphi}$$

Les formules précédemment établies fournissent alors les composantes symétriques des courants absorbés par le filtre :

$$\begin{cases}\underline{I}'_d = j\dfrac{\underline{V}_d}{V_d}I_d \sin\varphi_d = j\dfrac{\underline{V}_d}{R}\cos\varphi \cdot \sin\varphi\\ \underline{I}'_i = -\underline{I}_i = \dfrac{\underline{V}_d}{R}\cos\varphi \cdot e^{-j\varphi}\end{cases}$$

car $\varphi_d = \varphi$.

Il ne reste plus qu'à recomposer les courants absorbés par le filtre :

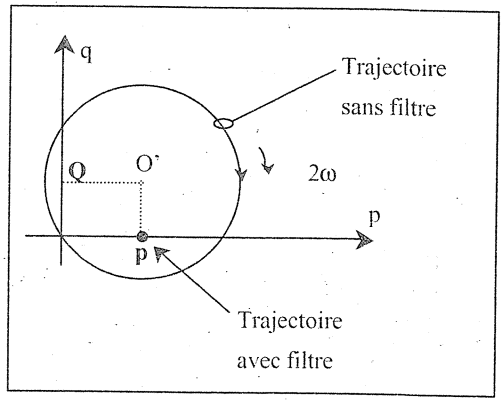
$$\begin{bmatrix}\underline{I}'_1\\ \underline{I}'_2\\ \underline{I}'_3\end{bmatrix} = \begin{bmatrix}1 & 1 & 1\\ 1 & a^2 & a\\ 1 & a & a^2\end{bmatrix}\cdot\begin{bmatrix}0\\ \underline{I}'_d\\ \underline{I}'_i\end{bmatrix} = \frac{\underline{V}_d}{R}\cos\varphi\begin{bmatrix}j\sin\varphi + e^{-j\varphi}\\ a^2 j\sin\varphi + ae^{-j\varphi}\\ aj\sin\varphi + a^2 e^{-j\varphi}\end{bmatrix}$$

$$\begin{bmatrix}\underline{I}'_1\\ \underline{I}'_2\\ \underline{I}'_3\end{bmatrix} = \frac{\underline{V}_d}{R}\cos\varphi \cdot \begin{bmatrix}\cos\varphi\\ a\cos\varphi + \sqrt{3}\sin\varphi\\ a^2\cos\varphi - \sqrt{3}\sin\varphi\end{bmatrix}$$

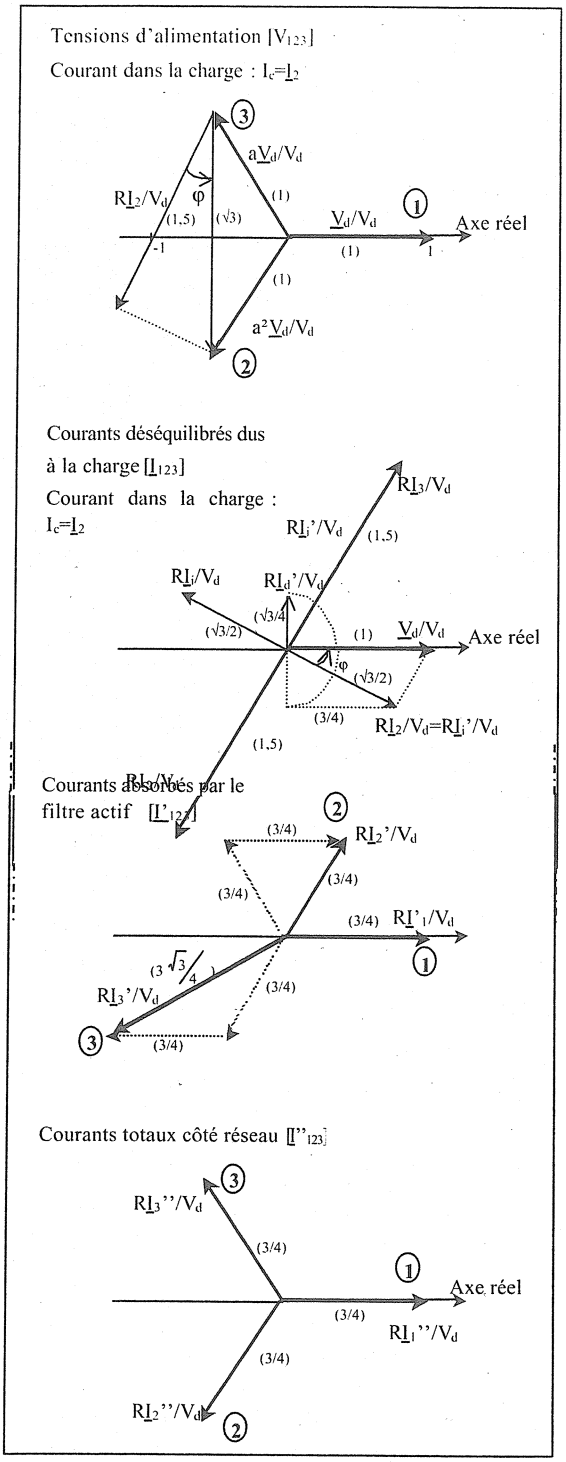
Avec l'origine des temps choisie pour les tensions simples du réseau $v_1(t) = V\sqrt{2}\cos\omega t$, il est aisé d'écrire les courants de référence du filtre :

$$\begin{cases}i'_{1réf} = \dfrac{V\sqrt{2}}{R}\cos^2\varphi \cdot \cos\omega t\\ i'_{2réf} = \dfrac{V\sqrt{2}}{R}\cos^2\varphi \cdot \left[\cos(\omega t + \dfrac{2\pi}{3}) + \sqrt{3} \cdot tg\varphi\cos\omega t\right]\\ i'_{3réf} = \dfrac{V\sqrt{2}}{R}\cos^2\varphi \cdot \left[\cos(\omega t - \dfrac{2\pi}{3}) - \sqrt{3} \cdot tg\varphi\cos\omega t\right]\end{cases}$$

On peut vérifier ces résultats sur le diagramme vectoriel suivant et constater que, dans le plan (p, q), la trajectoire présentée par la charge se réduit, pour le réseau, au point d'abscisse P et d'ordonnée nulle



lorsque le filtre intervient.



IV. CONCLUSION

Ce travail a pour but d'initier une réflexion sur la manière d'aborder l'enseignement de l'Electrotechnique, plus précisément sur la tendance naturelle à s'appuyer presque exclusivement sur des schémas équivalents comportementaux électriques qui, bien que simples et pratiques ne sont pas sans entraîner certaines déformations perverses.

Ainsi les schémas équivalents monophasés des charges triphasées symétriques, sous prétexte de gommer les difficultés de l'étude des distributions triphasées en reprenant les concepts en usage en monophasé, laissent le technicien et l'ingénieur totalement démunis face à la réalité des problèmes posés par la distribution et le transport de l'énergie qui, et on peut le regretter, n'interviennent jamais en monophasé .

Il convient par conséquent de développer des concepts et des méthodes qui permettent d'aborder l'étude des connexions énergétiques indépendamment du nombre de conducteurs utilisés et de la forme alternative ou continue des tensions et courants.

Cet objectif ne peut être atteint qu'en s'appuyant sur un principe physique dont l'universalité est reconnue, le principe de conservation de l'énergie et de sa dérivée temporelle qui, dans le domaine électrique représente une puissance instantanée. C'est la raison pour laquelle il m'a semblé intéressant de réactiver les concepts attachés aux grandeurs instantanées, notamment celui de la puissance fluctuante, pour renforcer les passerelles mathématiques reliant les informations lues sur nos appareils de mesure aux observations des oscillogrammes.

Il convient également de prendre conscience que les progrès accomplis dans le domaine des interrupteurs statiques rendent possible la maîtrise des formes d'onde des tensions et courants aussi bien aux stades de la production et du transport de l'énergie électrique que de sa consommation.

Cette réflexion est nécessaire pour expliquer l'émergence de nouveaux convertisseurs effectuant un prélèvement de l'énergie au réseau de distribution sous une forme quasi-optimale tant en terme de facteur de puissance que de forme d'onde.

Bibliographie :

MANESSE G Conversion de l'énergie électrique -Principes généraux - Application aux redresseurs de puissance SEE Club13 Cercle thématique 13-01 3EI97 20-21 Mars 1997 SUPELEC

HAUTIER J.P et CARON JP Convertisseurs statiques. Méthodologie causale de modélisation et de commande. Edition Technip Paris 1998

MANESSE G Les indicateurs de pollution harmonique dans les chaînes de conversion statiques . Comment faire la part des choses ? REE n°3 Mars1999 pp 44 à 51.

MANESSE G Transformateurs statiques – Principes de fonctionnement – Techniques de l'ingénieur . Traité de Génie Electrique D030 pp 1 à 48 Novembre 1999.

EXEMPLE DE REALISATION D'UN FILTRE DE COMPENSATION

Gilles FELD
ENS CACHAN

Georges MANESSE
CNAM PARIS

Résumé
Cet article illustre le concept de la puissance réactive instantanée, en proposant un montage expérimental qui rend compte des compensations possibles sur un réseau triphasé équilibré en tension. La commande du filtre actif capacitif utilisée réalise une absorption sinusoïdale.

Quand la charge est non équilibrée et éventuellement non linéaire, le montage proposé (fig 1) permet de compenser :

la puissance fluctuante.
la puissance réactive.
et éventuellement la puissance déformante

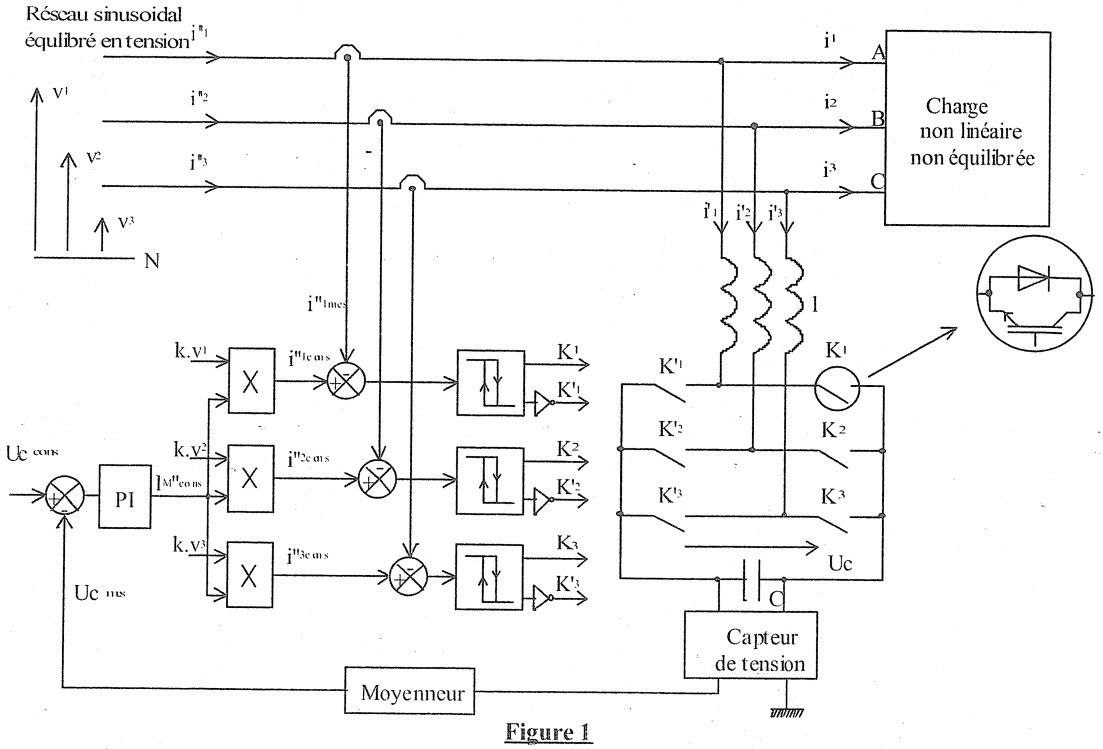


Figure 1

l = 3 mH C = 1200 µF V = 60 V Uc = 200 V

1) Principe de fonctionnement.

- Les correcteurs par hystérésis de courant permettent de garantir une erreur nulle sur les composantes de courant à 50 Hz. Dans ce cas, les courants fournis par le réseau seront en phase avec les tensions simples respectives et auront pour amplitude $I_M''_{cons}$.

En négligeant la composante de courant à la fréquence de découpage puis en supposant que les signaux d'entrée des multiplieurs ($k.v_1$,$k.v_2$ et $k.v_3$) soient normés et enfin que les chaînes de retour soient unitaires, la puissance fournie par le réseau s'écrit :

$$P = 3.V.I''.\cos\varphi'' = \frac{3}{\sqrt{2}}.V.I_M''_{cons}$$

- Le correcteur de la boucle externe de tension permet d'annuler l'erreur sur la valeur moyenne de la tension aux bornes du condensateur calculée sur un horizon de quelques périodes du secteur.

Cette erreur étant nulle, nous pouvons écrire que la puissance moyenne fournie par le réseau est égale à la puissance moyenne consommée par la charge (calculée sur quelques périodes du secteur).
Pour satisfaire cette condition, le régulateur de tension délivre une grandeur image de l'amplitude du courant fourni par le réseau telle que :

$$I_{M\,cons}'' = \frac{P_{charge}\cdot\sqrt{2}}{3.V}$$

Ce résultat est d'autant plus facile à obtenir que les dynamiques des deux boucles imbriquées de courant et de tension sont différentes.

2) Exemples.

2.1) Charge linéaire

La charge est constituée d'une résistance branchée entre les bornes A et B.

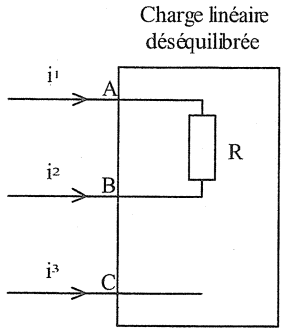


Figure 2

En supposant le compensateur non dissipatif, l'égalité des puissances permet d'écrire :

$$P = V.\sqrt{3}.I_1 = V.\sqrt{3}.I_2 = 3.V.I'' \Rightarrow I'' = \frac{Il}{\sqrt{3}}$$

On en déduit le diagramme vectoriel de la figure 3.

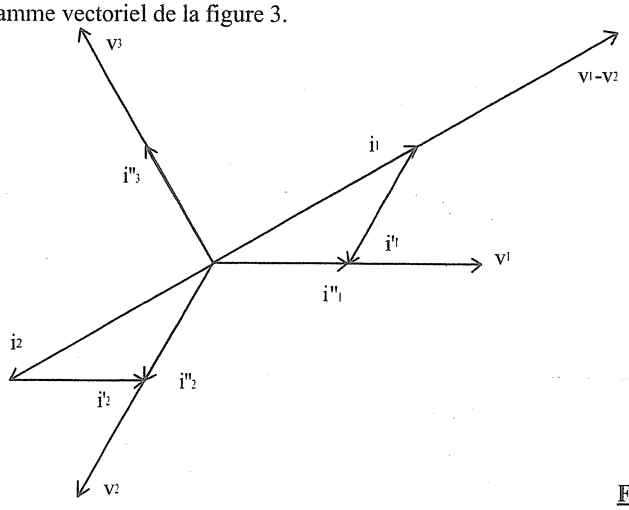


Figure 3

Les résultats expérimentaux et les formes d'ondes sont donnés sur la figure 4.

Remarque.

Les éléments constituant le compensateur n'étant pas parfaits, la relation sur les valeurs efficaces des courants n'est pas tout à fait satisfaite.
Les mesures donnent :

$$\frac{I''_{mes}}{Il_{mes}} = \frac{2,8}{4,4} = \frac{1}{1,6} > \frac{1}{\sqrt{3}}$$

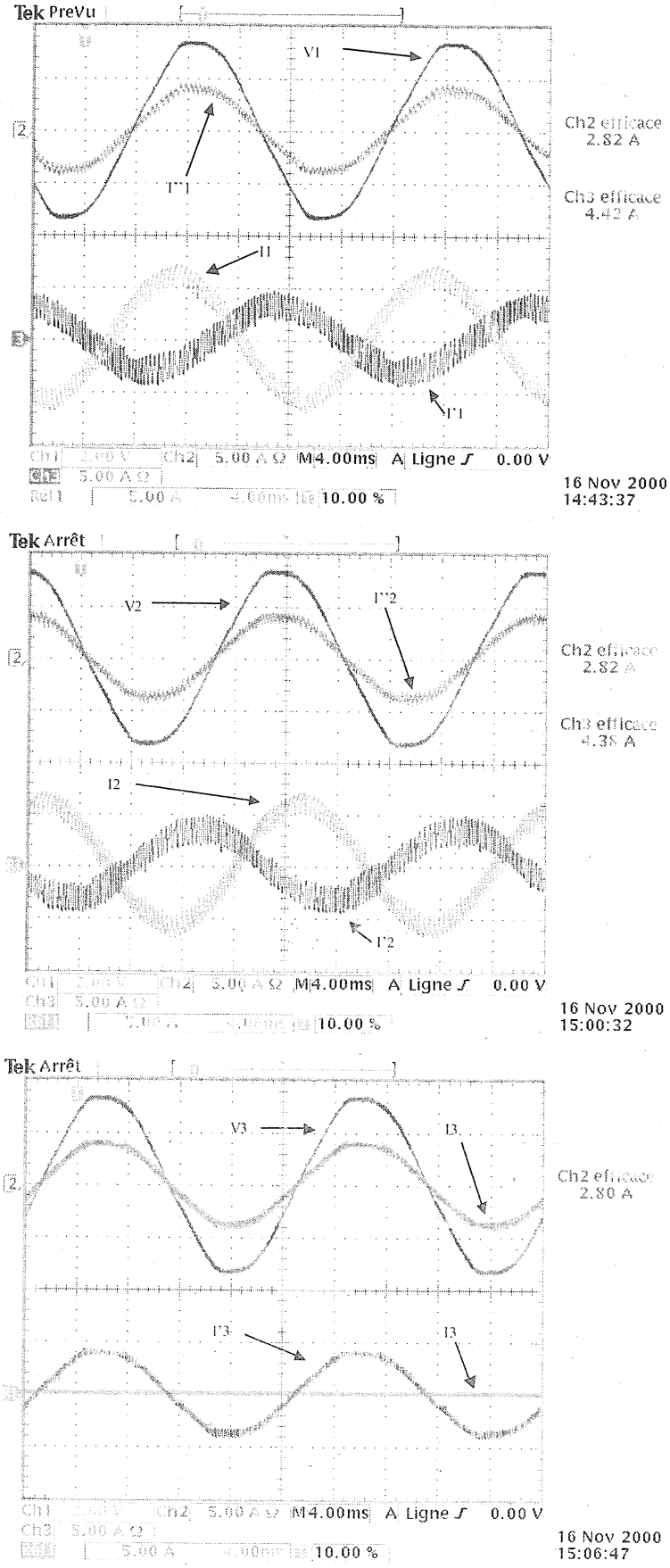

Tek PreVu
V1
I''1
I1
I'1
Ch2 efficace
2.82 A
Ch3 efficace
4.42 A
Ch1 5.00 V Ch2 5.00 A Ω M4.00ms A Ligne ƒ 0.00 V
Ch3 5.00 A Ω
10.00 %
16 Nov 2000
14:43:37
Tek Arrêt
V2
I''2
I2
I'2
Ch2 efficace
2.82 A
Ch3 efficace
4.38 A
Ch1 5.00 V Ch2 5.00 A Ω M4.00ms A Ligne ƒ 0.00 V
Ch3 5.00 A Ω
10.00 %
16 Nov 2000
15:00:32
Tek Arrêt
V3
I3
I'3
I3
Ch2 efficace
2.80 A
Ch1 5.00 V Ch2 5.00 A Ω M4.00ms A Ligne ƒ 0.00 V
Ch3 5.00 A Ω
10.00 %
16 Nov 2000
15:06:47

Figure 4

2.1) Charge non linéaire

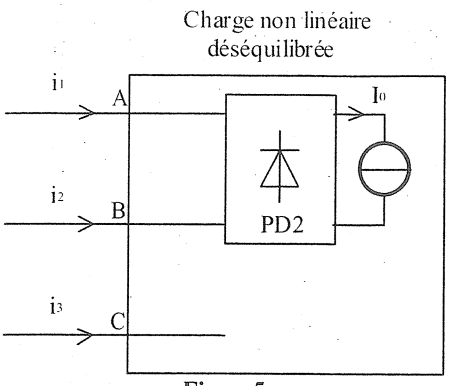
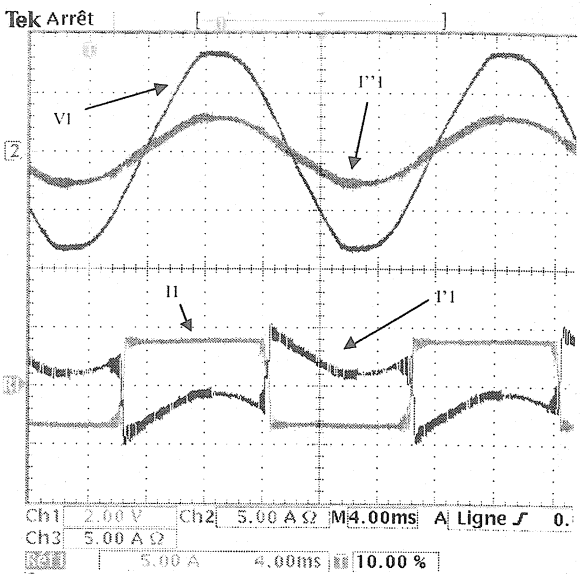
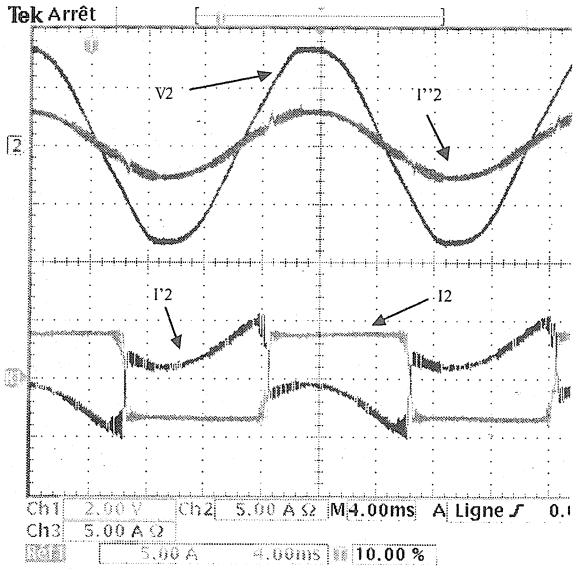
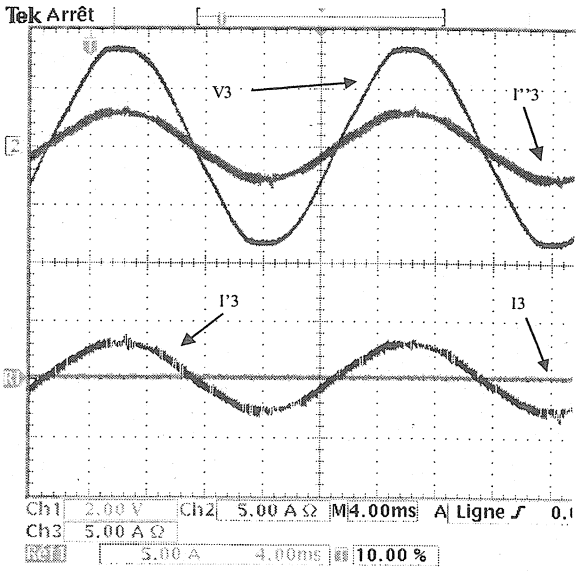


Figure5

Dans le cas d'une charge non linéaire branchée entre les bornes A et B, les formes d'ondes sont données par les figures suivantes :







3) Conclusion

L'augmentation de la fréquence de découpage permet de réaliser des générateurs de courant dont on sait contrôler la forme d'onde de façon à injecter des courants comportant à la fois une composante réactive et des harmoniques permettant d'absorber sur le réseau des courants équilibrés sinusoïdaux en phase avec les tensions respectives.

Ce compensateur fournit une solution élégante pour réaliser vu du réseau une charge idéale purement résistive et ceci quelle que soit la charge réelle.

En contre partie, le coût du compensateur (pour le distributeur ou l'utilisateur?) va augmenter avec les imperfections de la charge.

REALISATION D'UNE CARTE DE COMMANDE UNIVERSELLE POUR MONTAGES A THYRISTORS

Philippe MISSIRLIU
Lycée Newton-ENREA
1, pl. Jules Verne 92110 Clichy
PHILIPPE.MISSIRLIU@WANADOO.FR

1 Présentation :

Envoyer sur la gâchette de chacun des thyristors, d'un montage redresseur ou gradateur, une impulsion de largeur définie et ceci avec un retard déterminé par rapport au réseau d'alimentation. Voici un très vieux problème pour lequel bien des solutions existent. Quelques exemples :

- Montage à amplificateurs opérationnels.
- Circuit spécialisé (type TCA 785).
- Montage à base de circuits logiques comme celui proposé dans le numéro de mars 2000 de cette revue.
- Carte de commande à base de microcontrôleur.

Nous proposons ici notre version de cette dernière solution conçue avec les objectifs suivant :

- Une carte de commande universelle
- Une réalisation simple
- Un coût modique (596,00 FHT)
- Toutes les informations (y compris le source du programme) sont disponibles afin que chacun puisse comprendre et éventuellement adapter le fonctionnement de la carte.

2 Cahier des charges :

La carte de commande fourni les impulsions d'amorçage pour six thyristors avec les options suivantes :

- Montage redresseur ou gradateur
- Système triphasé direct ou inverse

De plus, dans le fonctionnement en redresseur on dispose du choix entre :

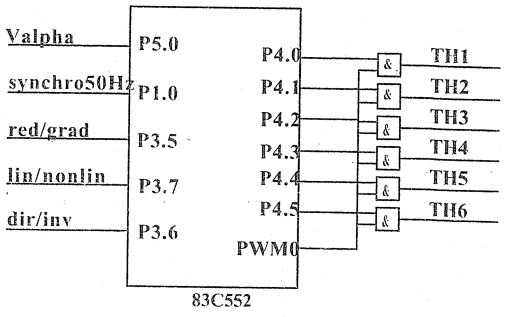
- Mode non linéaire : l'angle de retard à l'amorçage est proportionnel à la tension de commande.
- Mode linéaire : la tension moyenne aux bornes de la charge est proportionnelle à la tension de commande (utilisation d'une fonction arccosinus).

La valeur du retard à l'amorçage est rafraîchie 6 fois par période soit toutes les 3,3ms.
On prévoit des butées d'angle telles que $10° \leq alpha \leq 170°$.

Remarques :

- Nous avons fait le choix d'envoyer des impulsions d'amorçage pendant toute la durée de conduction possible d'un thyristor ce qui revient à générer des créneaux de 120° de largeur en redresseur et de 180° en gradateur.
- Dans tous les cas, on prend comme référence temporelle le passage par 0 en valeurs croissantes de la tension simple V_1.
- Pour commander un pont mixte il suffira de n'utiliser que les trois premiers signaux d'amorçage.
- Les signaux de commande des deux thyristors de la phase un en mode gradateur pourront être utilisés pour commander un pont redresseur monophasé.
- Le microcontrôleur fourni six créneaux (sortie P4.0 à P4.5) plus un signal carré de fréquence 11,8 kHz (sortie PWM0) qui, combinés, donnent des trains d'impulsions.

Schéma :



Les entrées :

NOM	NATURE	FONCTION
V ALPHA	Analogique 0-5V	Signal de commande
Synchro50 Hz	Logique	Image de la tension V_1
Red / $\overline{grad}$	Logique	Montage redresseur ou montage gradateur
Lin / $\overline{nonlin}$	Logique	Utilisation ou non de la fonction arccosinus
Dir / $\overline{inv}$	Logique	Ordre des phases

3 Le matériel :

3.1 Le microcontrôleur :

Le microcontrôleur 83C552 de Philips est un des nombreux dérivés du 8051. Nous l'avons choisi pour les raisons suivantes :

- Il possède les périphériques intégrés dont nous avions besoin : convertisseur analogique/numérique 10 bits, générateur de signaux MLI et un timer particulier (le timer 2) associé à des registres de comparaison et de capture.
- Une carte d'évaluation bon marché est disponible chez Farnell.
- La mémoire programme est une Eprom externe qui peut ainsi être programmée par n'importe quel programmateur.

3.2 La carte d'évaluation:

La carte d'évaluation, vendue en kit, se présente sous la forme d'un circuit imprimé double face avec trous métallisés comportant des emplacements pour le microcontrôleur et les mémoires EPROM et RAM externes, des entrées-sorties (boutons poussoirs, voyants, RS232), et une zone à wrapper (voir photo n°1).

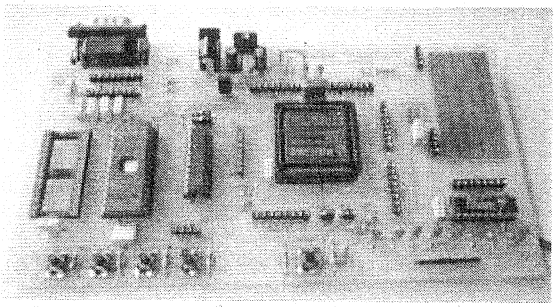


Photo n°1

Pour la réalisation de la carte de commande universelle seuls sont à monter sur le circuit imprimé le microcontrôleur et l'EPROM.

3.3 Son intégration dans un projet :

Le problème est le suivant : intégrer cette carte dans un rack format Europe, réaliser les interfaces d'entrée et de sortie. Nous procédons de la façon suivante :

- Découper la carte d'évaluation au format Europe (100x160 mm).
- Réaliser une carte (également au format Europe) munie d'un connecteur au format des racks utilisés au Lycée sur laquelle sont placées les interfaces.
- Monter ces deux cartes en « sandwich » à l'aide d'entretoises.
- Relier électriquement les deux couches du « sandwich » à l'aide d'un câble plat.

Le résultat est visible sur la photo n°2
Dans le cadre des projets de STS Electrotechnique cette solution a pour avantage de séparer le travail en deux parties distinctes :

- Une carte microcontrôleur industrielle fournie.
- Une carte interface entièrement conçue et réalisée par l'étudiant.

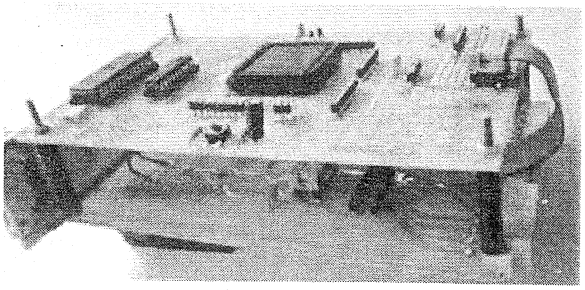


Photo n°2

4 Le logiciel :

Cette question est trop complexe et trop spécialisée pour être développée ici. Le programme a été écrit en C avec les outils RAISONANCE. On trouvera sur le site du lycée aussi bien le fichier binaire permettant de programmer l'EPROM pour ceux qui veulent utiliser directement la carte que le programme source avec des explications pour ceux qui désirent aller plus loin.

5 Exemple d'application :

Nous avons réalisé un projet intitulé « pupitre d'étude des montages à thyristor» qui utilise toutes les possibilités de la carte présentée ici. L'objectif était de pouvoir réaliser en TP tous les montages classiques à thyristors : redressement par pont mixte ou tout thyristor, gradateur, le tout soit en monomhasé soit en triphasé. Le projet est visible sur la photo n°3.

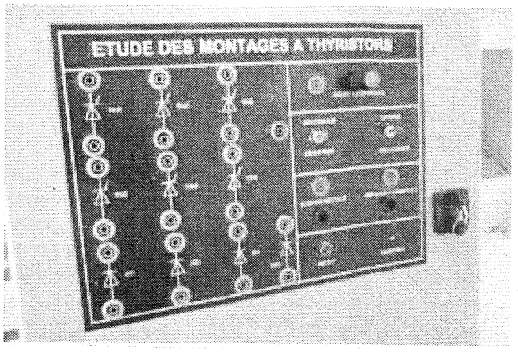


Photo n°3

6 Les ressources en ligne :

Nous avons mis sur le site internet de la section Electrotechnique du Lycée Newton-ENREA de Clichy (http://membres.tripod.fr/tsetclichy) toutes les ressources pour, non seulement réaliser la carte de commande universelle, mais aussi comprendre son fonctionnement :

- Le source du programme écrit en C.
- Le fichier format *.bin* pour programmer l'EPROM.
- Les références exactes du matériel à commander.
- Un texte reprenant et développant le contenu de cet article (en particulier le programme).
- Le rapport du projet présenté ci dessus.
- Des liens divers.

LIAISON NUMERIQUE RS 485 AU PROTOCOLE MODBUS

Laurent MARTIN, Jean-Pierre ODELOT *(stagiaires en STS CCF Electrotechnique 2000)*

David BRUNET, Philippe LE BRUN *(professeurs en STS Electrotechnique)*

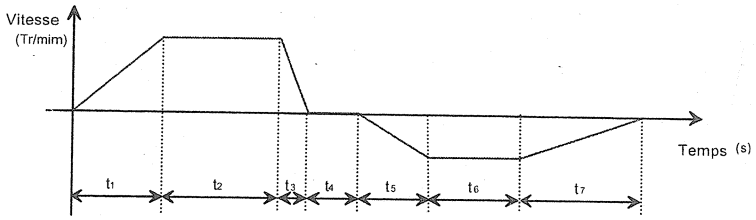
Lycée Louis ARMAND
173 Bd de Strasbourg
94736 NOGENT sur MARNE cedex
Florence.vadee@wanadoo.fr

Résumé :

Le dialogue entre différents équipements (Automates programmables, variateurs, régulateurs...) se fait maintenant le plus souvent grâce à des réseaux de terrain. Le protocole ouvert MODBUS est un des plus souvent rencontrés dans les milieux industriels, de nombreux constructeurs le proposent donc à leur catalogue. C'est le cas d'ABB pour l'automate programmable et d'OMRON pour le variateur de fréquence en U/f que nous avons fait dialoguer via une RS 485 au protocole MODBUS RTU.

Introduction

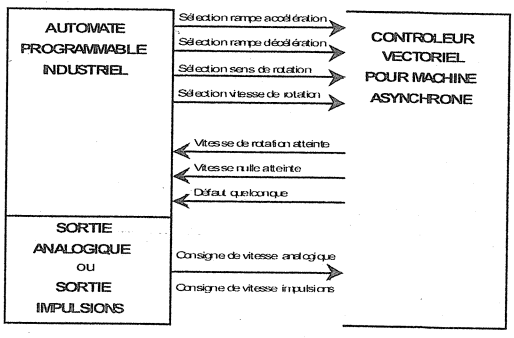
Sur un système de positionnement linéaire vertical mu par un moto-variateur asynchrone et un contrôleur vectoriel, on désire suivre un profile de vitesse conforme au graphique ci-dessous.



T1: Accélération montée durée 5 secondes
T2: Vitesse constante montée à 800 tr/min
T3: Freinage montée durée 2 secondes
T4: Attente durée 3 secondes
T5: Accélération descente durée 4 secondes
T6: Vitesse constante descente à 500 tr/min
T7: Freinage Descente durée 7 secondes

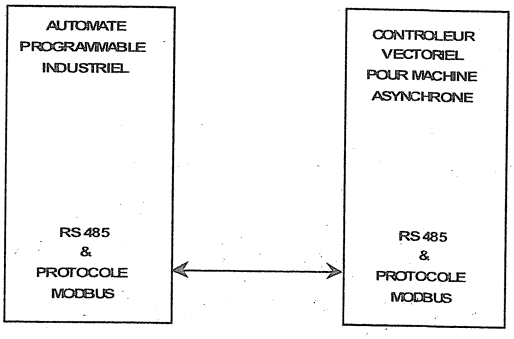
Dans une solution sans réseau de terrain le dialogue se fait par des sorties logiques (voir ci-contre) commutant des fonctions pré-programmées dans le contrôleur vectoriel ce qui impose :

- Une proximité géographique des deux appareils
- Un câblage important entre les deux matériels
- Pas ou peu de problèmes de connexion entre appareils
- Une programmation/configuration indépendante des deux matériels



Dans une solution avec réseau de terrain le dialogue se fait grâce à une seule liaison numérique (voir ci-contre) dont les supports et protocoles de communication doivent être identiques sur les deux matériels permettant :

- Un éloignement géographique des deux appareils
- Un câblage extrêmement simple entre les deux matériels
- Une programmation au niveau de l'automate (y compris la configuration et le paramétrage du contrôleur vectoriel)
- Une mise au point et une maintenance plus aisées
- Des fonctions supplémentaires possibles (supervision, télémaintenance, télé configuration...)



Présentation du protocole MODBUS®

Le protocole MODBUS® est un protocole de transmission de données régissant le dialogue entre un « maître » et un ou plusieurs « esclaves ».
Les échanges se font sur l'initiative du « maître », qui émet sa demande. Le « maître » peut s'adresser individuellement aux « esclaves » ou s'adresser à l'ensemble des « esclaves ». Les « esclaves » renvoient une réponse sauf dans le cas d'un message général.

Choix du mode de transmission

La communication au protocole MODBUS® peut s'effectuer selon deux modes de transmissions :

- ASCII : chaque octet composant une trame est codé avec 2 caractères ASCII (2 fois 8 bits)
- RTU : chaque octet composant une trame est codé sur 2 caractères hexadécimaux (2 fois 4 bits)

Le mode ASCII permet d'avoir des intervalles de plus d'une seconde entre les différents caractères sans que cela ne génère d'erreurs, alors que le mode RTU permet un débit plus élevé pour une même vitesse de transmission.
C'est le mode RTU qui est utilisé dans nos deux matériels.

Structure des messages

Les échanges « maître-esclaves » s'effectuent par l'envoi de message dont le format de base de la trame est le suivant :

ADRESSE (8 bits)	FONCTION (8 bits)	DONNEES (n x 16 bits)	CONTROLE D'ERREUR (16 bits)

L'ensemble des informations contenues dans le message est exprimé en hexadécimal.

Le maître s'adresse à l'esclave en mettant **l'adresse de cet esclave** dans le champ adresse du message.
Puis le **code de fonction** indiquant à ce dernier le type d'action qu'il doit réaliser. Dans le cas de notre application, nous utiliserons trois fonctions principales :

- Lecture de registre(s) (code de fonction $03_{(HEX)}$) ;
- Ecriture dans un ou plusieurs registre(s) (code de fonction $10_{(HEX)}$) ou message à diffusion générale (code de fonction $10_{(HEX)}$ et adresse esclave $00_{(HEX)}$);

Ensuite, le **champ de donnée(s) est codé sur *n* mots** codés en héxadécimal de 00 à FF, soit sur n octets. Suivant le code fonction, le champ de donnée contient diverses informations complémentaires permettant à l'esclave de décoder le message (voir les exemples d'échanges maître esclave de la page suivante).

Enfin, dans le cas du mode de transmission RTU, le **champ contrôle d'erreur** contient une valeur codée sur 16 bits. Cette valeur est le résultat d'un « Cyclical Redundancy Check » (CRC) calculé à partir du contenu du message.

Lorsque **l'esclave** renvoie sa réponse, il place sa **propre adresse dans le champ adresse** afin que le maître identifie de quel esclave provient cette réponse.

Ensuite, il utilise le **champ fonction pour indiquer si la réponse est sans erreur ou non**. En effet, pour une réponse normale, l'esclave reprend le même code fonction que celui du message envoyé par le maître, sinon il renvoie un code erreur correspondant au code original avec son MSB à 1.

Le **champ de donnée(s)** contient diverses informations dépendant du code fonction (voir les exemples d'échanges maître esclave de la page suivante).

Enfin, dans le cas du mode de transmission RTU, le **champ contrôle d'erreur** contient une valeur codée sur 16 bits. Cette valeur est le résultat d'un « Cyclical Redundancy Check » (CRC) calculé à partir du contenu du message.

Exemples d'échanges maître-esclave

Lecture de données
« Lecture de la fréquence de référence »

message maître

N. octet	Données		Exemple
1	Adresse : esclave		01
2	Code fonction : lecture registre		03
3	N° du registre de début de lecture	MSB	00
4		LSB	02
5	Nombre de registre de lecture	MSB	00
6		LSB	01
7	Contrôle CRC		25
8			CA

réponse esclave (sans erreur)

N. octet	Données		Exemple
1	Adresse : esclave		01
2	Code fonction : lecture registre		03
3	Nbre octets données		02
4	Données du registre 0002	MSB	02
5		LSB	58
6	Contrôle CRC		B8
7			DE

réponse esclave (avec erreur)

N. octet	Données	Exemple
1	Adresse : esclave	01
2	Code fonction : lecture avec (MSB=1)	83
3	Code d'erreur (n° registre)	03
4	Contrôle CRC	01
5		31

Ecriture de données
« Mise en marche et choix fréquence de référence »

message maître

N. octet	Données		Exemple
1	Adresse de l'esclave		01
2	Code fonction		10
3	N° du registre de début d'écriture	MSB	00
4		LSB	01
5	Nombre de registre d'écriture	MSB	00
6		LSB	02
7	Nombre d'octets de données		04
8	Données dans le registre n° 0001	MSB	00
9		LSB	01
10	Données dans le registre n° 0002	MSB	02
11		LSB	58
12	Contrôle CRC		63
13			39

réponse esclave (sans erreur)

N. octet	Données	Exemple
1	Adresse de l'esclave	01
2	Code fonction	10
3	N° du registre de début d'écriture	00
4		01
5	Nombre de registre d'écriture	00
6		02
7	Contrôle CRC	10
8		08

Message à diffusion générale
« Mise en marche de tous les esclaves »

message maître

N. octet	Données		Exemple
1	Adresse de l'esclave		00
2	Code fonction		10
3	N° du registre de début d'écriture	MSB	00
4		LSB	01
5	Nombre de registre d'écriture	MSB	00
6		LSB	01
7	Nombre d'octets de données		02
8	Données dans le registre n° 0001	MSB	00
9		LSB	01
12	Contrôle CRC		6B
13			D1

réponse esclave (sans erreur)

PAS DE REPONSE

Support de transmission

Chaque octet composant un message est transmis de la manière suivante :

LSB....MSB

En mode de transmission RTU, la séquence de bit pour transmettre un octet est la suivante :

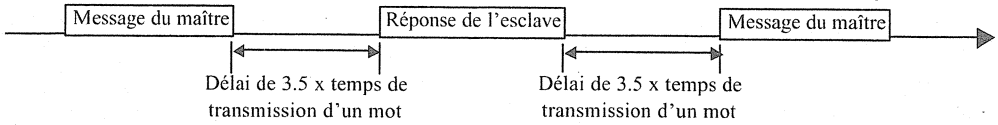
Start	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Stop

Sans contrôle de parité

Start	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Parité	Stop

Avec contrôle de parité

De plus, avant et après chaque message, il doit y avoir un silence équivalent à au moins 3,5 fois le temps de transmission d'un mot.



L'ensemble du message doit être transmis de manière continue. En effet, si un silence de plus de 1,5 fois le temps de transmission d'un mot intervient en cours de transmission, le destinataire du message considérera que la prochaine information qu'il recevra sera l'adresse de début d'un nouveau message.

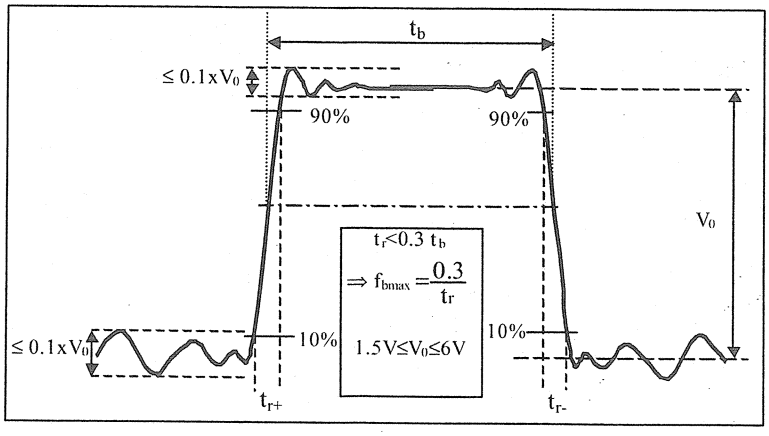
Le protocole MODBUS® ne définit que la structure des messages et leur mode d'échange. Par conséquent, on peut utiliser n'importe quel support de transmission. Les plus couramment utilisés sont les liaisons RS232, (RS422) ou RS485.

	Liaison RS 232	Liaison RS 422	Liaison RS 485
Nbre d'émetteurs et récepteurs	1 émetteur et 1récepteur	1 émetteur et 10 récepteurs	1 émetteur et 32 récepteurs
Type de transmission	unidirectionnelle	unidirectionnelle	bidirectionnelle
Vitesse de transmission	Jusqu'à 115 200 Bauds	Jusqu'à 19200 Bauds	Jusqu'à 19200 Bauds
Longueur max.	Quelques mètres	1200m	1200m

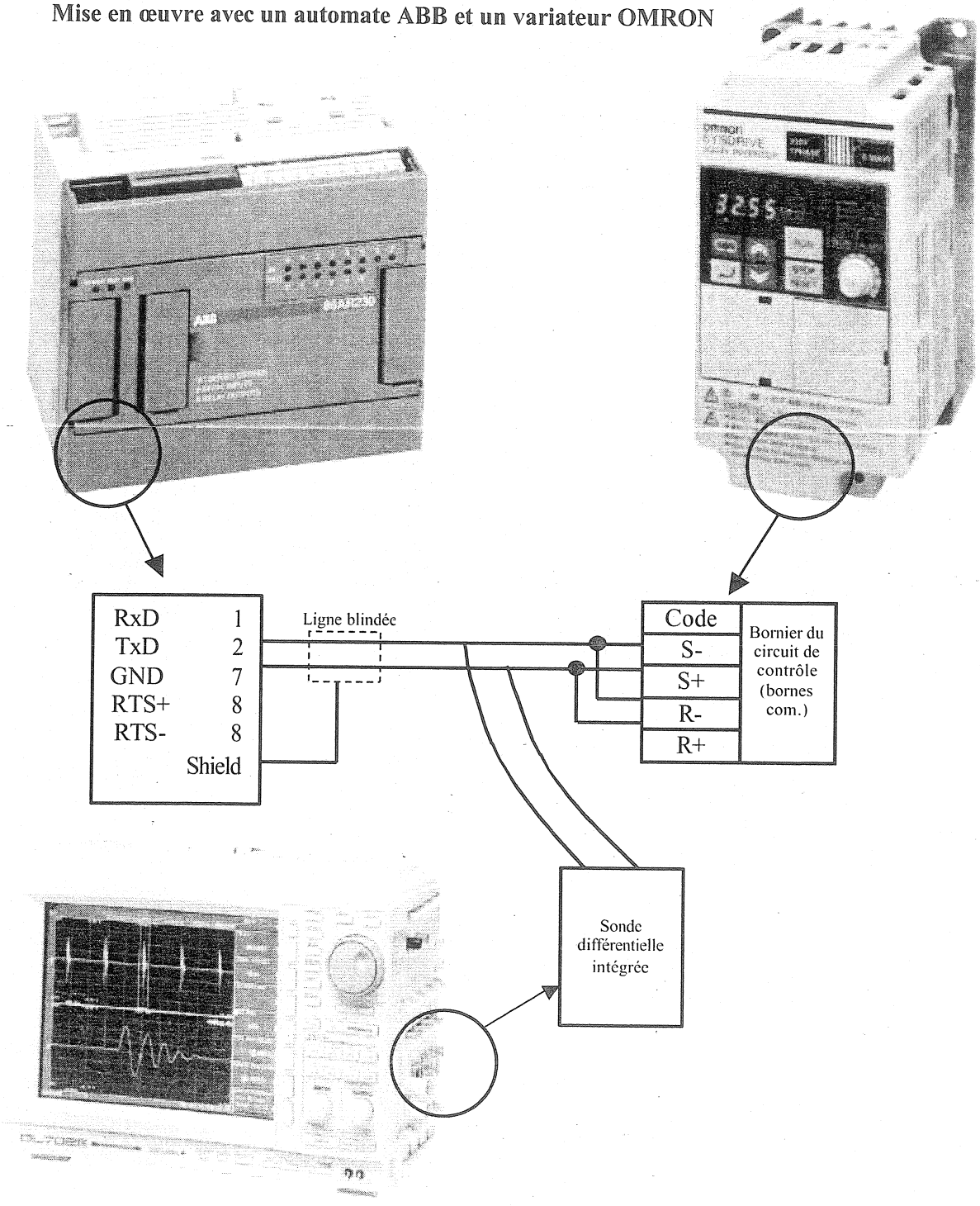
Ici nous avons mis en œuvre une liaison RS485 bidirectionnelle sur deux fils et majoritairement utilisée dans l'industrie pour des réseaux MODBUS®.

Caractéristiques de la liaison RS485 :

Tension maximum en mode commun	-7V à +12V
Résistance d'entrée du récepteur	12 kV
Charge de l'émetteur	60 Ω
Limite de courant de court-circuit	150 mA à la masse 250 mA à–7V ou +12V



Mise en œuvre avec un automate ABB et un variateur OMRON

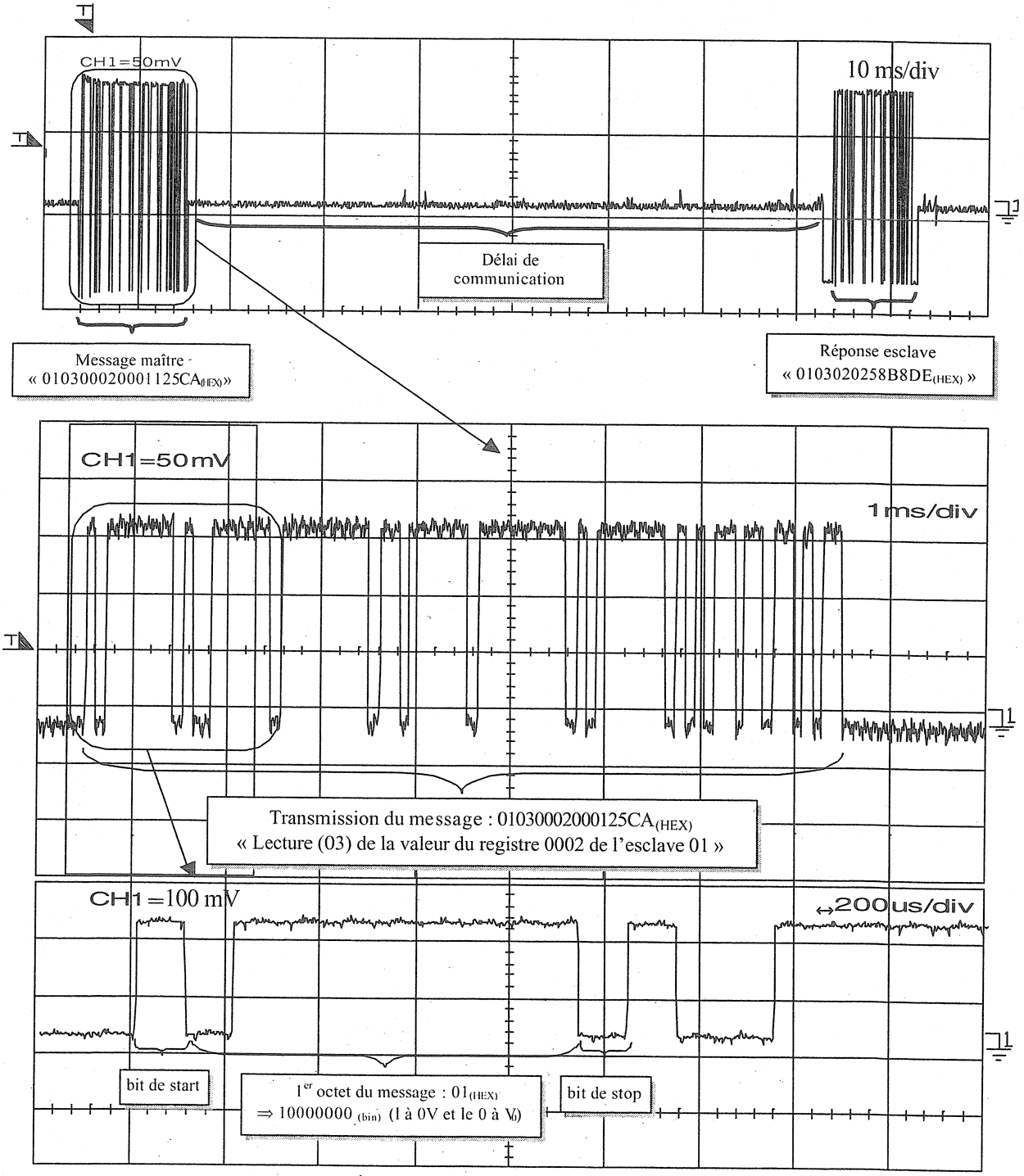


Concernant la connexion au niveau du variateur, il est possible de ne connecter que deux des bornes de communication pour n'effectuer qu'une communication unidirectionnelle. De plus, la liaison RS485 nécessite une résistance de terminaison de ligne, qui peut être intégrée au variateur ou non.

La connexion entre l'automate et le variateur est directement dépendante du sens de branchement entre leurs bornes de communication respectives, néanmoins il n'est pas destructif. Par conséquent, en cas de dysfonctionnement, il suffit d'inverser la connexion au niveau de l'un des deux.

Etude de la trame en RS485

Dans l'exemple qui suit, nous visualisons le message aller-retour entre le maître et l'esclave. Nous pouvons notamment observer sur la figure ci-dessous, le silence qui sépare l'envoi du message de celui de la réponse de l'esclave. Concernant le contenu des messages échangés, nous reprenons l'exemple de lecture de données qui a été donné dans le paragraphe concernant les exemples d'échanges maître-esclave.
Dans un deuxième, nous allons nous intéresser de plus près à la structure du message, et plus particulièrement du message émanant du maître. Concernant les états électriques correspondant aux bits transmis, il faut savoir que le bit de start est à V_0, le bit de stop à 0V, par contre, pour les bits de données, le 1 est à 0V et le 0 à V_0. De plus, comme nous l'avons déjà précisé, concernant la transmission d'un octet, le LSB est transmis en premier.



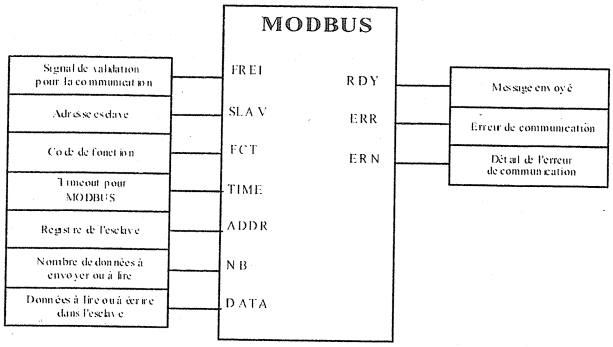
Conclusion

Dans les milieux industriels la mise en œuvre des réseaux, propriétaire ou ouvert, se fait maintenant très couramment. Les prix des composants sont beaucoup plus abordable et permettent de simplifier grandement le câblage et la maintenance des installations. Les difficultés de mise en œuvre se sont réduites et ont, elles aussi, contribué à une plus grande diffusion des compétences chez les techniciens de mise œuvre de ces matériels.

Il est donc souhaitable que nous abordions lors de nos essais de système, au moins un réseau de type industriel. Les coûts des équipements pour de tels essais deviennent plus abordables et leur mise en œuvre même si elle reste délicate autorise des erreurs de câblage sur la liaison numérique (testé sur une liaison RS 485). Nos étudiants (niveau BTS électrotechnique) ont travaillé une année durant sur la réalisation de la maquette (photo ci-contre en haut) et se sont intéressé à la liaison RS 485 au protocole MODBUS® (câblage, paramétrage et programmation de l'automate). Un outil d'analyse des mots circulant sur la ligne s'est avéré indispensable lors de la mise en œuvre de cette liaison (convertisseur RS 485 / RS 232 + logiciel pour moins de 1000 F et un PC).

Le système câblé avec les différents appareils d'investigation (oscilloscope et analyseur basique de la liaison RS 485) est présenté en situation sur la page de droite en bas. Cette maquette permet, outre l'étude de la liaison RS 485, d'aborder d'autres aspects de la mise en œuvre des systèmes automatisés (Mise en œuvre de matériels : automate programmable, variateur de fréquence et compteur autonome; raccordement d'entrées : contacts, capteurs avec sortie 2 fils et capteurs avec sortie 3 fils; raccordement des sorties : voyants, contacteurs; dialogue entre les différents matériels : liaison par contacts, liaison par sortie transistorisée, liaison par impulsion modulée en fréquence et liaison numérique; programmation ou paramétrage des différents matériels mis en œuvre).

La programmation de la liaison MODBUS® fait appel à un bloque fonction spécialisé dans lequel il suffit d'indiquer l'ensemble des informations décrite dans cet article (voir ci-dessous)



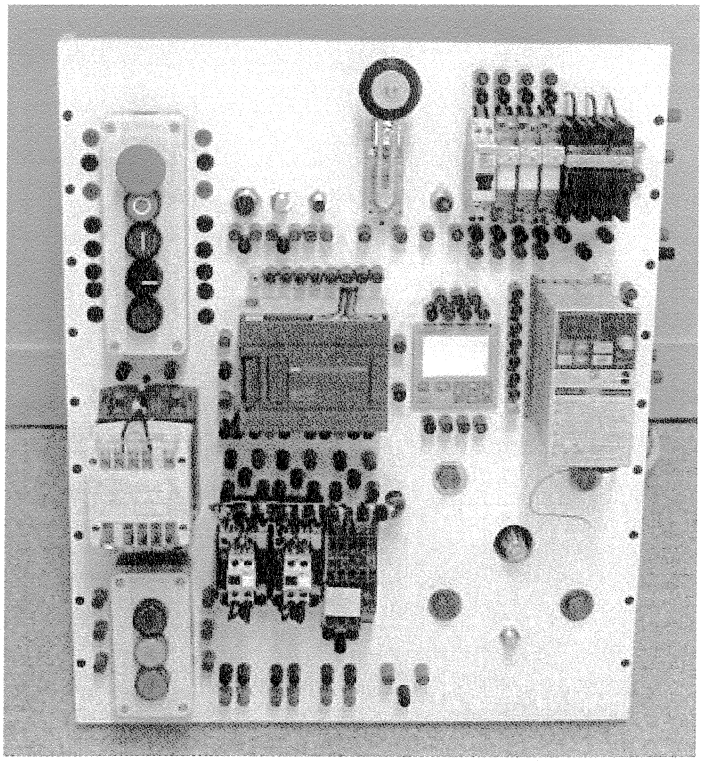
ANNEXE PROGRAMMATION MODBUS

Voici le bloque fonction MODBUS permettant le pilotage numérique du convertisseur de fréquence :
SLAV : adresse esclave
FCT : code fonction
ADDR : adresse ou il faut lire ou écrire
NB : nombre d'octet de données à lire ou écrire
DATA : données à écrire ou à lire
TIME : temps de réponse maxi de l'esclave
FREI : ordre d'envoi du message

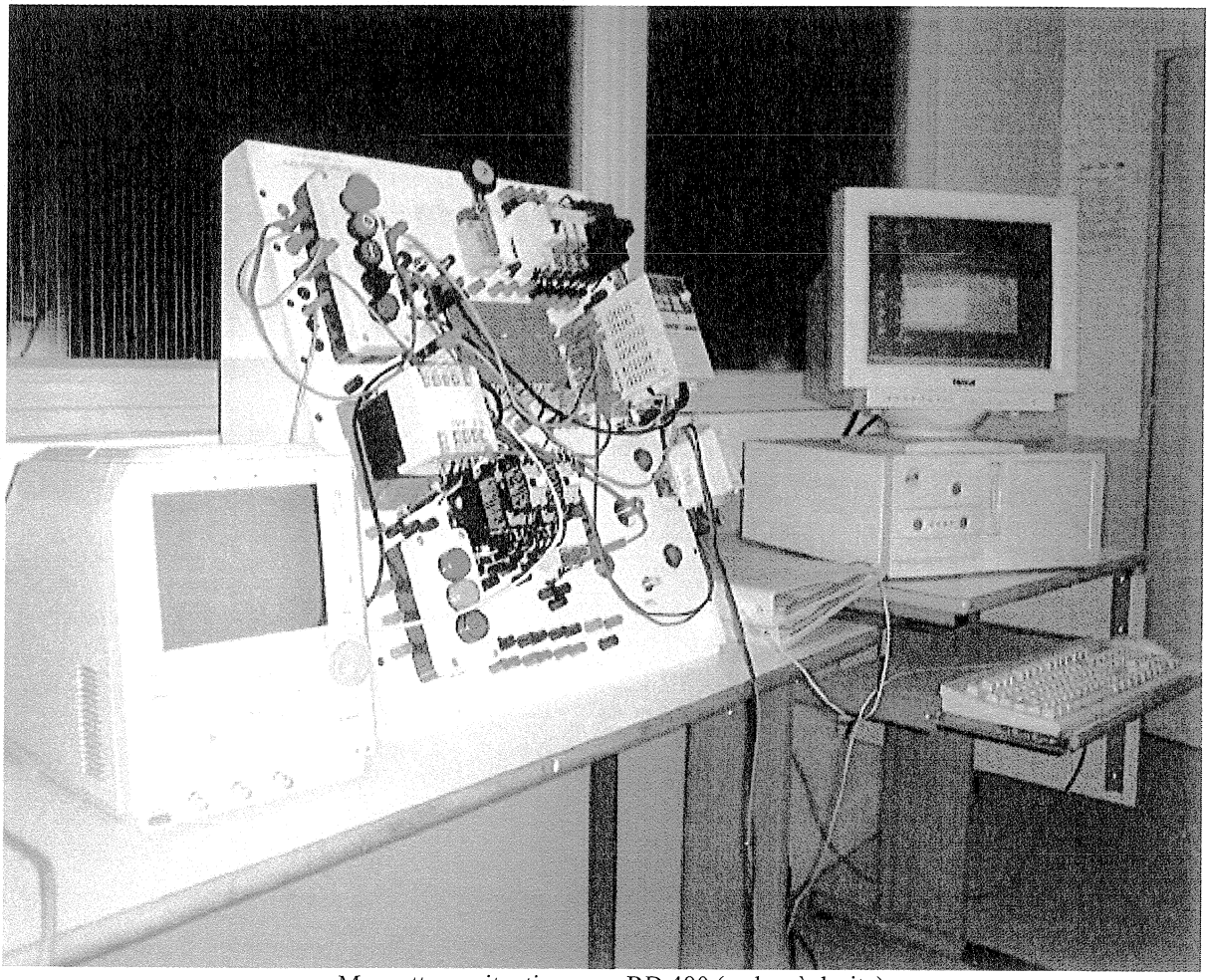
Libellé	Désignation	Valeur utilisée
FREI	C'est la validation du signal pour le lancement du message qui s'effectue sur front montant	Entrée automate
SLAV	C'est l'adresse du numéro de l'esclave	01
FCT	Lecture (3), écriture (6), lecture et écriture (16)	16
TIME	Temps de réponse maxi de l'esclave (en millisecondes)	65
ADDR	C'est l'adresse de registre à lire ou à écrire dans l'esclave	Suivant prog.
NB	Nombre de d'octet à lire ou à écrire dans l'esclave	01
DATA	C'est la valeur à lire ou à écrire dans le registre défini dans ADDR	Suivant prog.
RDY	Message envoyé	
ERR	Cette sortie indique une erreur si elle est à 1	
ERN	Détail de l'erreur	

Remerciements

Nous tenons à remercier la société OMRON www.omron.fr (MM Thierry LANDAIS et Frank SIMON plus particulièrement) pour les conseils et l'aide efficace qu'elle à pu nous offrir lors du choix et de la mise en œuvre des produits qu'elle commercialise. Le coût de l'onduleur (scalaire et vectoriel) avec RS 485 et protocole MODBUS® 3G3MV est compris entre 1500 et 3000 F pour une puissance comprise entre 500 W et 5 kW. Le coût du convertisseur RS 485 / RS 232 bidirectionnel RD 400 associé au logiciel d'émission / réception des messages est inférieur à 1000 F.



Maquette nue

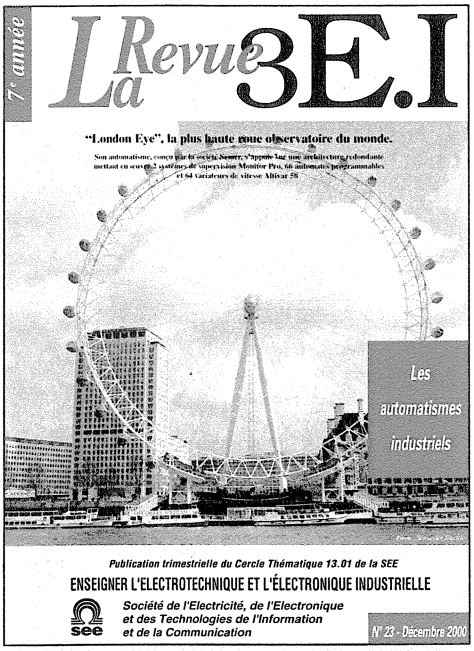


Maquette en situation avec RD 400 (en bas à droite)

3EI
Enseigner l'Electrotechnique et l'Electronique Industrielle

c'est

Une revue trimestrielle :



Un site web pour partager l'Enseignement du Génie Electrotechnique :
http://www.lesite3EI.com

Thèmes prévus pour l'année 2000-2001

- ☞ *L'éclairage - n° 24 (mars 2001)*
- ☞ *L'état de l'art en électronique de puissance - n° 25 (juin 2001)*
- ☞ *Pratiques pédagogiques et réalités industrielles n° 26 (septembre 2001)*
- ☞ *Métrologie : Mesure et estimation - n° 27 (décembre 2001)*

A venir :

- ☞ *Convertisseurs propres et filtres actifs*
- ☞ *Electronique de puissance embarquée*

Tarifs des abonnements 2000-2001 pour les 4 numéros

France et pays de la CEE
Individuel : 195 F TTC
Collectivité (bibliothèque, CDI, laboratoire, université, école d'ingénieurs, lycée, IUT, entreprise ...) : 250 F TTC

Pays hors CEE
Individuel : 260 F TTC
Collectivité (bibliothèque, CDI, laboratoire, université, école d'ingénieurs, lycée, IUT, entreprise ...) : 320 F TTC

Une seule adresse : SEE - la Revue 3EI
48, rue de la Procession - 75724 PARIS CEDEX 15