MASTER EN SCIENCES INFORMATIQUES

Gestion des commandes d'une grue à tour : de la spécification formelle au programme de l'automate

Bracaval, Frédéric

Award date:
1996

Awarding institution:
Universite de Namur

GESTION DES COMMANDES D'UNE GRUE A TOUR : DE LA SPECIFICATION FORMELLE AU PROGRAMME DE L'AUTOMATE

Frédéric Bracaval

GESTION DES COMMANDES D'UNE GRUE A TOUR : DE LA SPECIFICATION FORMELLE AU PROGRAMME DE L'AUTOMATE

Travail de fin d'études
présenté par

Frédéric Bracaval

Année académique
1995-1996

pour l'obtention du grade
de Licencié en Informatique

AVANT-PROPOS

Je tiens à remercier mon promoteur M. Eric Dubois pour le temps qu'il m'a consacré et qui, grâce à son cours, m'a donné l'ouverture d'esprit indispensable à la réalisation de ce mémoire.

Je souhaiterais également exprimer mes plus vifs remerciements à M. Jean-Marc Zeippen pour son aimable accueil et ses explications qui me furent d'une grande utilité.

Je ne voudrais pas terminer ces quelques lignes sans remercier également M. Wautelet sans qui le présent sujet n'aurait vu le jour.

Table des matières

Chapitre 3 : La technologie des automates programmables

Chapitre 4 : De la spécification à la solution

Introduction

Le présent mémoire a pour objet la spécification formelle du système de commande d'une grue à tour dans le but d'en augmenter la sûreté et ainsi réduire les erreurs pouvant mettre en cause la sécurité des personnes sur les chantiers.

Avant de préciser davantage le rôle de la spécification formelle et puisque ce mémoire s'adresse également à des personnes peu familières du génie logiciel, rappelons brièvement les différentes phases du cycle de vie d'un logiciel [SOM92] :

1) **La spécification des besoins (définition et analyse)**
   Les services, buts et contraintes du système sont établis en interrogeant les utilisateurs du système et définis de manière à ce qu'ils soient compréhensibles par les utilisateurs et l'équipe de développement.

2) **La conception du logiciel**
   Une architecture logique d'ensemble du système est établie. Les fonctions du système sont représentées de manière à être facilement transformables en programmes exécutables.

3) **L'implémentation et les tests du système**
   La conception du logiciel est implémentée en un ensemble d'unités de programmation. Ces unités sont testées individuellement afin de vérifier qu'elles correspondent à leur spécification de conception, elles sont ensuite intégrées en un système complet testé pour s'assurer que les besoins logiciels ont bien été satisfaits.

4) **La mise en oeuvre et la maintenance**
   Le système est livré au client, installé et mis en service. La maintenance concerne la correction d'éventuelles erreurs ignorées jusque là, l'adaptation et l'enrichissement des services du système lors de la découverte de nouveaux besoins.

Dans la pratique, ces phases se chevauchent et interagissent. Il est clair que le processus logiciel[1] ne peut pas être décrit par un modèle simple et linéaire : il comporte une série d'itérations sur les activités de développement mais pas trop, puisqu'il faut pouvoir définir des points de contrôle pour le planning et le suivi. Donc, après un certain nombre d'itérations, certaines parties du développement, notamment la spécification des besoins, doivent être figées afin de passer aux étapes suivantes. Suite à ce gel prématuré des besoins, les fonctionnalités du système ne correspondent pas toujours aux attentes de l'utilisateur ce qui peut entraîner des coûts de correction significatifs.

---

[1]Ensemble des activités impliquées dans la production d'un logiciel

En effet, si le système développé ne correspond pas aux souhaits du client, il faut modifier les besoins et, de ce fait, refaire la conception, l'implémentation, les tests et la validation du système. Cela augmente les coûts particulièrement si cette erreur n'est pas découverte avant l'implémentation.

Il est donc primordial de fournir un effort tout particulier au début du développement, c'est-à-dire au niveau de la définition et de la spécification des besoins.

La spécification des besoins : source principale d'erreurs

En toute généralité, une erreur logicielle est définie [LUT93] comme un écart entre une valeur - ou condition - calculée, observée ou mesurée et la valeur - ou condition - exacte, spécifiée ou théoriquement correcte.

Dans la suite de ce mémoire, le logiciel conçu pour l'automate contrôlant la grue à tour peut être considéré comme critique au point de vue de la sécurité puisqu'il surveille et contrôle des composants pouvant être impliqués dans un comportement du système jugé risqué ou dangereux pour les personnes travaillant sur le chantier. Ce logiciel doit répondre en temps-réel à la surveillance du matériel et aux commandes de l'utilisateur et inclut des problèmes de temporisation pour certaines parties du système.

Il a été démontré [LUT93] que, outre des problèmes de communication entre équipes, les causes premières des erreurs du logiciel sont plus généralement des erreurs de reconnaissance des besoins : non identifiés, mal documentés ou incompris. Les erreurs d'un système temps-réel tendent à être produites principalement [LUT93] par des écarts entre la spécification des besoins et les besoins nécessaires au fonctionnement correct du système ainsi que par des incompréhensions au niveau des interactions avec le reste du système.

Ces résultats suggèrent que les difficultés avec les besoins, autrement dit les spécifications inadéquates, sont la source première des erreurs mettant en cause la sécurité. Ces sources d'erreurs peuvent être détectées et donc supprimées durant le développement grâce à certaines procédures. Les recommandations suivantes ont émergé de l'analyse des erreurs relatives à la sécurité dans les systèmes embarqués [LUT93] :

1) **Se concentrer sur les interactions entre le logiciel et le système en analysant le domaine du problème**
   Capacités et limitations matérielles, liens de communication, environnement de fonctionnement attendu (t°, pression) doivent être reflétés dans la spécification des besoins parce qu'ils sont des sources fréquentes d'erreurs. Les dépendances de temps doivent également être incluses dans la spécification : ce sont des sources d'erreurs particulièrement difficiles puisque liées à l'exactitude fonctionnelle des systèmes.

2) **Identifier les risques le plus tôt possible dans l'analyse des besoins**
   Les besoins et le domaine du problème ne sont pas fixes, ils évoluent : l'impact des besoins inconnus sur les étapes ultérieures du développement ne doit pas être sous-estimé. L'insertion des erreurs dès le début du développement et leur découverte tardive maximisent le temps et les efforts pris pour la correction.

3) **Utiliser des techniques de spécification formelle en plus des spécifications des besoins en langage naturel**
Le manque de précision et l'incomplétude des spécifications mènent à la plupart des erreurs mettant en cause la sûreté du système. Il est primordial que la spécification soit assez détaillée pour couvrir toutes les situations possibles et que les hypothèses concernant l'environnement soient correctement documentées.

4) **Promouvoir la communication informelle entre équipes**
Une grande partie des erreurs résulte d'un malentendu ou d'une incompréhension d'un individu ou d'une équipe sur un besoin ou encore de l'ignorance d'un fait au sujet du système.

5) **Répercuter facilement les changements résultant de l'évolution des besoins**
Il est fréquent d'observer que des changements qui semblent n'impliquer qu'un seul composant du système en affectent d'autres. La "traçabilité" des besoins (concept d'annotation) est importante à ce niveau car elle offre une validation du système.

Ce mémoire traite plus particulièrement des troisième et cinquième points cités ci-dessus, à savoir l'élaboration d'un spécification formelle du système de commande d'une grue à tour suivie de son implémentation sur un automate programmable tout en gardant trace des liens pertinents entre ces deux phases du processus (traçabilité).

Définition des besoins et spécification formelle

Dans un premier temps, nous traitons du processus qui consiste à déterminer les services qu'un système logiciel doit offrir et les contraintes sous lesquelles il doit fonctionner : la capture et l'analyse des besoins. La spécification des besoins est le document résultant de cette analyse.

Comme décrit par le modèle du développement du logiciel, avant de concevoir et d'implémenter une solution informatique, il faut recueillir des informations sur le problème à résoudre, les analyser et fournir une définition complète de ce problème.

Par la suite, nous distinguerons la définition des besoins de la spécification des besoins. Ces deux documents différents s'adressent à des classes de lecteurs différentes [SOM92] :

- **La définition des besoins** est un énoncé en langue naturelle des services que le système est censé fournir et doit être compréhensible par les décideurs, les utilisateurs et les acheteurs potentiels du système (orientée client).

- **La spécification des besoins** est un document structuré qui décrit les services de manière plus détaillée et qui doit être suffisamment précis pour servir de base contractuelle entre le client et le fournisseur du logiciel. Elle doit être claire et compréhensible, elle ne doit pas être ambiguë car cela risque de mener à différentes interprétations côté client et côté acheteur. C'est à ce niveau qu'interviennent les techniques de spécification formelle basées sur des outils mathématiques.

Le cahier des charges, quant à lui, est la combinaison de la définition des besoins et de leur spécification. Les besoins doivent y être consistants (non contradictoires) et complets c'est-à-dire que toutes les fonctions du système doivent y être spécifiées. Ce document qui peut comporter des erreurs et omissions doit être structuré de manière à être facilement modifiable car il doit servir d'outil de référence aux gens qui seront chargés de la maintenance du système.

Mais revenons-en à la spécification formelle.

Contrairement à la définition des besoins, au niveau de la spécification, le langage naturel se révèle inadéquat [MEY85] [SOM92] :

- il suppose que tous utilisent les mêmes mots pour les mêmes concepts. Du fait de l'ambiguïté inhérente au langage naturel, cela peut mener à des quiproquos;
- trop flexible, il permet d'exprimer des besoins voisins dans des termes totalement différents;
- les besoins sont enchevêtrés au lieu d'être clairement cloisonnés. Afin de s'assurer que le changement d'un besoin n'a pas de conséquences au niveau des autres, c'est l'ensemble de tous les besoins qui doit être réexaminé.

Suite à l'inadéquation du langage naturel pour la rédaction de la spécification et dans l'optique d'une diminution des coûts de développement, les besoins systèmes sont exprimés à l'aide de langages de spécification formelle.

La spécification formelle d'un logiciel est une spécification exprimée dans un langage dont le vocabulaire, la syntaxe et la sémantique sont définis de manière formelle. De par ce besoin d'une sémantique formelle, les langages de spécification doivent se baser sur des langages mathématiques [SOM92].

Une spécification formelle est donc **un modèle mathématique du système**.

La détection d'erreurs ou d'omissions constitue l'argument le plus puissant en faveur de la spécification formelle, du fait que les erreurs de spécification des besoins qui ne sont pas détectées immédiatement coûtent très cher à corriger. Or, lorsqu'on fait une spécification formelle, on est obligé d'analyser le système en détail, ce qui permet souvent de révéler des erreurs et des inconsistances de la définition des besoins.

La spécification obtenue est non-ambiguë, les malentendus entre le fournisseur et le client en ce qui concerne les fonctionnalités du système sont fortement réduits.

Un autre avantage lorsqu'on utilise un tel langage, est que l'on peut utiliser des outils logiciels de vérification syntaxique et sémantique permettant de détecter des omissions ou des inconsistances.

Malheureusement, l'inconvénient majeur d'un langage formel est de ne pas être toujours compréhensible par les clients non-techniciens, du fait de l'importance des notations mathématiques à la base de ce modèle : la visibilité du processus n'est pas améliorée pour autant que l'on produise des documents formels. Il exige une certaine

formation de la part de l'utilisateur pour comprendre les notations spécialisées, aussi, certains sont réticents à ce genre de spécification. Ce problème est en partie résolu en paraphrasant la spécification formelle, en y ajoutant des commentaires en langage naturel.

Le projet CAT

Le projet CAT[2] soutenu par la Région Wallonne (DGTRE / OBJECTIF 1) a pour but le développement d'un environnement d'outils destinés à supporter la modélisation et l'analyse des besoins fonctionnels des systèmes distribués et temps-réel critiques.

Ces outils, destinés à faciliter la vérification et la validation des besoins, reposent sur le langage formel **Albert II** développé à l'Institut d'Informatique des Facultés Universitaires de Namur. Un éditeur graphique dédié fonctionnant sous Windows 95 en facilite la rédaction mais ce dernier n'est pas indispensable : malgré ses notations spécialisées, une spécification **Albert II** n'est pas un obstacle pour les traitements de texte étant donné l'existence d'une syntaxe ASCII.

Ce mémoire rentre dans le contexte de l'une des études de cas de ce projet à savoir l'étude de cas "GRUE". Il constitue un cahier des charges à la base du développement d'un prototype du système étudié. Prototype dans le sens où il vise à démontrer la faisabilité de l'application et qu'il n'inclut qu'un sous-ensemble des fonctions requises : en outre, les besoins non fonctionnels tels que le coût, la fiabilité, la portabilité, la taille, ne sont pas pris en considération.

Dans la suite de ce document, nous abordons successivement les sujets suivants.

Le premier chapitre présente le fonctionnement général d'une grue à tour, expose les problèmes inhérents à leur utilisation ainsi que les besoins du système informatique proposé. Le second chapitre a trait à la spécification formelle et présente brièvement les déclarations graphiques du système. Le troisième chapitre se rapporte au principe de fonctionnement des automates programmables en toute généralité et aborde plus précisément l'architecture et le langage de l'automate utilisé pour l'implémentation finale du système. Le quatrième et dernier chapitre, plus détaillé, est consacré à quelques considérations techniques, à l'implémentation, aux contraintes de la spécification formelle et aux explications concernant la correspondance entre ces contraintes et le programme.

[2] Conception et Analyse de cahiers des charges pour des systèmes informatiques de Télécommunication

Chapitre

1

Présentation du problème et définition des besoins

Présentation du problème

Contexte et objectif de l'étude

Ce document concerne une des études de cas réalisées dans le cadre du projet CAT : l'étude de cas "GRUE". Celle-ci a été réalisée en collaboration avec l'entreprise Wautelet, une S.P.R.L. en construction de matériel électrotechnique de la région de Charleroi. Cette entreprise gère notamment un parc d'une cinquantaine de grues à tour. Son responsable doit faire face à des problèmes de sécurité survenant lors de l'utilisation des grues par les opérateurs d'entreprises de construction.

L'objectif visé par cette étude est de diminuer les risques d'accident survenant au cours de la manipulation de grues à tour sur chantier en remplaçant le système de commande actuel par un système informatique plus sûr, robuste et incontournable, bref, écartant toute action violant les contraintes de sécurité.

Afin de mieux sécuriser les grues à tour, la solution envisagée est de :
⇨ remplacer le système de sécurité électromécanique par un automate programmable;
⇨ utiliser des capteurs "sans mouvement" (de proximité, inductifs, photoélectriques), plus robustes que les capteurs mécaniques actuellement utilisés.

Démarche de l'étude du problème

Le but de cette étude concerne donc la mise au point d'un système de contrôle assurant la sécurité sur les chantiers lors de la manutention de charges par des grues à tour.

L'étude du système s'est effectuée suivant les étapes que voici.

Dans un premier temps, acquisition des informations et estimation de la faisabilité du projet par la technologie actuelle. Les informations concernant le problème, son contexte et les moyens disponibles pour y apporter une solution ont été prélevées dans la documentation [CNAC80] et par des techniques classiques d'interviews [ZEI95a] :
- le problème : comment fonctionne une grue ?
- le contexte : comment sont-elles utilisées par les opérateurs sur les chantiers et quels sont les problèmes de sécurité et leurs causes ?
- les moyens disponibles : automates programmables industriels et capteurs.

A partir de cette première analyse et de discussions avec le client, une définition des besoins en langage naturel a été établie de façon à être lisible et compréhensible par des lecteurs non-techniciens [ZEI95b]. Les besoins y sont décrits de manière abstraite et l'accent est porté sur la lisibilité : aucune notation spécialisée n'y figure.

En parallèle avec cette définition des besoins, une spécification formelle des fonctionnalités du système à développer a été réalisée au moyen d'Albert II, langage formel utilisé pour la description des composants des systèmes composites et temps réel

et dont il est question dans le projet CAT. L'utilisation d'une méthode de spécification formelle, basée sur les mathématiques, a permis d'obtenir une spécification des besoins rigoureuse, sans ambiguïtés.

Cette spécification formelle est une description précise et détaillée constituant une base contractuelle avec le client. Au cours de cette modélisation détaillée et suite à plusieurs relectures, des erreurs de la définition des besoins sont apparues et ont été corrigées entraînant un affinement progressif des objectifs du système.

Ajoutons que, pendant la formulation des besoins, des revues régulières avec le client ont été effectuées pour vérifier la complétude des besoins. Concernant la consistance des besoins, des outils automatiques peuvent être envisagés lorsque les besoins sont exprimés dans un langage formel [SOM92].

Cette définition et cette spécification constituent le cahier des charges ainsi qu'une source de documentation pour le développement de notre prototype, la réalisation du système "grandeur nature" sortant du cadre de cette étude.

Définition des besoins

La définition des besoins est la description abstraite des services que le système est censé fournir ainsi que de ses contraintes opérationnelles [SOM92]. Cette description ne spécifie que le comportement extérieur sans se préoccuper des détails techniques. Destiné aux utilisateurs, ce document se doit d'être lisible aussi on n'y introduira aucune notation spécialisée. Elle se concentre sur les concepts, les informations plus détaillées étant reprises dans la spécification.

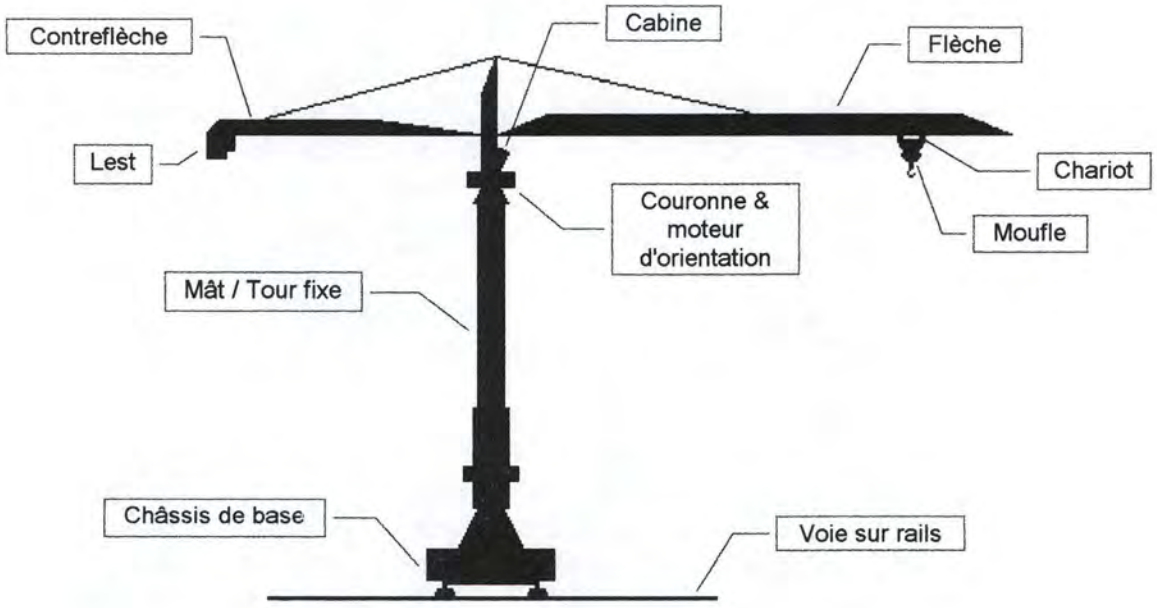
Nous commencerons par citer les différents organes d'une grue à tour en justifiant leur rôle. Ensuite, l'ensemble des objectifs généraux visant à améliorer la sûreté du système sera exposé, suivi de ceux faisant l'objet du prototype développé dans la suite de ce mémoire. Finalement, les besoins identifiés à implémenter sont passés en revue.

Architecture et fonctionnement d'une grue à tour

Dans la suite de ce document, un seul type de grues sera décrit : il s'agit des grues à tour avec contreflèche à contrepoids et couronne d'orientation située au sommet du mât. Dans la plupart des cas, ce type de grue à tour peut effectuer un mouvement de translation sur des rails mais celle-ci reste fixe au cours des manipulations. Nous ne nous attarderons pas sur l'autre type de grues, celles repliables ou "à montage rapide" avec couronne d'orientation à la base.

Définition et terminologie

Une grue est définie comme étant un appareil de levage formé d'un bras orientable monté sur un support de hauteur variable. La figure 1.1 met en évidence ses parties essentielles.



~ Figure 1.1 : Composants d'une grue à tour avec rotation au sommet du mât ~

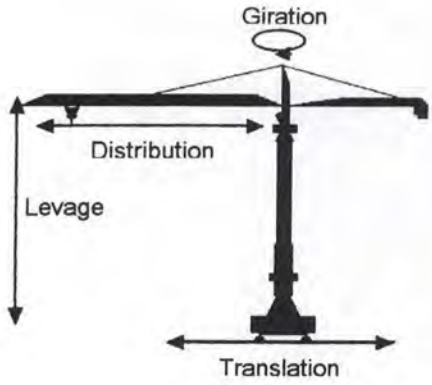
La puissance d'une grue s'exprime en tonnes-mètres. Elle est définie par le couple charge - portée comme suit :

- soit la portée maximale * la plus grande charge admissible à cette portée (ex. 15m * 1T);
- soit la charge maximale * la plus grande portée acceptant cette charge (ex. 15T * 1m).

Fonctionnement général

Les mouvements de la grue sont les suivants :

- ⇨ le levage : montée / descente du crochet;
- ⇨ la distribution : avance / recul du chariot;
- ⇨ la giration : rotation gauche / droite de la flèche;
- ⇨ la translation : avance / recul du châssis sur les rails.



Tout appareil de levage est muni de freins, cliquets d'arrêt ou autres dispositifs de sécurité, installés de telle façon qu'ils empêchent la descente inopinée des charges ou des accessoires servant à la manutention. En fait, tout dispositif d'entraînement mécanique d'un appareil de levage est équipé d'un dispositif de freinage mécanique conçu de telle façon qu'il puisse arrêter le mouvement en un temps raisonnable et le maintenir à l'arrêt [RGPT96].

Excepté pour la giration de la flèche, l'interruption de la transmission d'un mouvement provoque automatiquement la mise en action de son dispositif de freinage [RGPT96].

Outre les freins, une grue comporte les organes de sécurité suivants [CNAC80] :

- ⇨ **limiteurs de course du chariot;**
  limiteurs de course de levage (en levée et en descente)**;**
  limiteurs de mouvement de translation du châssis.

  Les limiteurs de course de levage et du chariot sont des dispositifs de sécurité assurant que les différents organes de l'engin ne viennent pas heurter les membrures de la flèche et diminuent ainsi le risque d'une rupture de câble [CNAC80]. Les limiteurs de mouvement de translation de la grue évitent de façon sûre le dépassement des positions extrêmes de la voie [RGPT96]. Remarquons finalement qu'en ce qui concerne l'angle de déplacement de la flèche, il n'y a pas de limiteurs d'orientation.

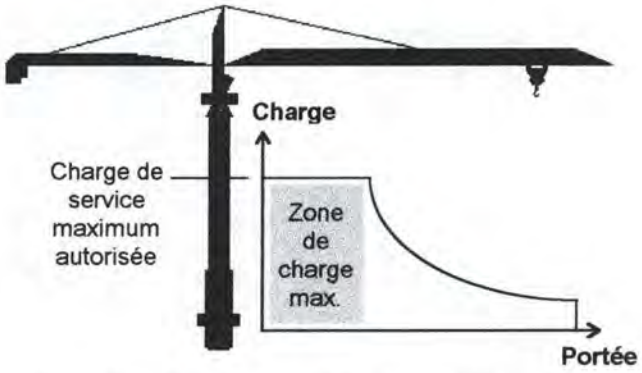
- ⇨ **limiteur de charge maximum**, il s'agit d'un dispositif dynamométrique enregistrant l'effort supporté par le câble de levage.

- ⇨ **limiteur de couple de charge (moment[1])** car la charge de service dépend de la position du chariot le long de la flèche [RGPT96] : elle décroît à mesure que le chariot s'éloigne de la cabine de l'opérateur vers l'extrémité de la flèche (fig. 1.2).

[1] Produit du poids de la charge soulevée par la longueur du bras de levier correspondant à la distance entre le mât de la grue et la position du chariot.

⇨ **anémomètre** : le préposé à la manoeuvre doit être averti en toutes circonstances si la vitesse du vent atteint ou dépasse une valeur qui peut se révéler dangereuse pour l'appareil et pour laquelle le travail avec la grue doit être arrêté ou adapté [RGPT96].

⇨ **avertisseur sonore (klaxon)** notamment pour que, lors du déplacement horizontal d'une charge, le préposé à la manoeuvre puisse, par un signal, prévenir le personnel situé à proximité du trajet de la charge [RGPT96]. Ce dispositif est également commandé par l'anémomètre pour prévenir le grutier.



~ Figure 1.2 : Charge admissible en fonction de la portée ~

En fait, les manoeuvres autorisées dépendent des facteurs suivants, gérés par le système de commande [ZEI95b]:

- la charge maximale admissible pouvant être soulevée, conditionnée par la résistance mécanique de la grue et celle du câble [CNAC80].
- le moment maximal limitant la montée de la charge et l'avance du chariot;
- la vitesse du vent : si ce dernier est trop fort, l'opérateur doit en être averti et mettre la grue hors service après avoir laissé la flèche libre de son mouvement de rotation.

En effet, dans le cas d'un vent trop puissant, l'anémomètre déclenche un avertisseur sonore - le klaxon - qui prévient le grutier du danger et de la nécessité de mettre la grue hors service : la commande de translation sur rails doit être neutralisée pour assurer la stabilité et le châssis de base doit être amarré à la voie pour interdire le moindre déplacement de la grue poussée par la force du vent.

Un mouvement de sécurité qui consiste à ramener le chariot près du mât et à descendre la charge doit toujours fonctionner, même dans les situations extrêmes.

Le moteur responsable du mouvement de levage peut fonctionner à petite, moyenne et grande vitesse tandis que les moteurs contrôlant les autres mouvements possèdent une voire deux vitesses maximum. Celles-ci doivent être engagées progressivement en respectant des contraintes de temporisation lors du passage d'une vitesse à la suivante ou à la précédente. L'inversion du sens de rotation de tout moteur doit également répondre à une contrainte de temporisation.

En période hors service [CNAC80],

- la flèche doit être laissée en girouette de façon à éviter que le vent ne s'engouffre dans la grue et ne prenne appui sur celle-ci en s'assurant au préalable que l'aire de rotation

est parfaitement dégagée et qu'elle ne risque pas d'interférer avec des fils électriques déviés aussi par le vent;
- la grue doit être placée et amarrée sur le tronçon de sécurité de la voie, le crochet doit être libre de charges et mis en position de sécurité c'est-à-dire en haut du mât;
- le chariot doit être amené près du mât et verrouillé.

Ajoutons que les organes de commande doivent être conçus de manière à ce que tout mouvement s'arrête automatiquement et soit maintenu en arrêt dès que l'opérateur cesse d'agir sur la commande [RGPT96].

Finalement, un bouton-poussoir d'arrêt d'urgence permet de couper l'alimentation sur toutes les phases de l'installation électrique de l'ensemble de la machine. Cet arrêt doit être facilement accessible par l'opérateur et permet d'immobiliser immédiatement la machine même dans des situations accidentelles totalement inattendues.

Objectifs du système proposé

L'objectif premier de l'utilisation d'un automate programmable pour le contrôle de la grue est l'augmentation de la sécurité sur les chantiers. En effet, les principaux risques d'accidents sont dus soit à l'inobservation des normes de sécurité ou des consignes données par le constructeur, soit à la déficience du matériel, à sa mauvaise installation ou encore à l'inconscience de certains utilisateurs (par exemple, coupure du système de sécurité afin d'outrepasser les limites de l'engin).

Parmi les points importants à respecter avant toute utilisation d'une grue, citons [CNAC80] :
- le choix de l'emplacement de la grue;
- l'installation de la voie de la grue;
- le montage de la grue;
- les précautions à prendre : lests, ancrage à un bâtiment, amarrage au sol, équipement électrique, nécessité d'utiliser des dispositifs tels que crochet de sécurité, limiteurs des mouvements, de charge maximum et de couple de charge.

Les systèmes de commande des grues actuellement sur les chantiers, gèrent un certain nombre de paramètres mais pas tous ceux susceptibles d'accroître la sécurité. En outre, lorsque les contraintes du système de commande deviennent trop embarrassantes, les opérateurs ont la possibilité de le "court-circuiter", ce dernier étant installé dans une armoire à portée de main.

Dans une perspective d'amélioration de la sécurité sur le chantier et suite à l'analyse détaillée du problème, il s'est révélé intéressant que le nouveau système de commande puisse prendre en considération les fonctionnalités suivantes :

- informer en permanence l'opérateur de la charge et du moment afin de lui permettre de mieux évaluer les situations à risques;

- mesurer et éviter la surintensité des moteurs et s'assurer de leur sens de rotation avant toute action (l'inversion des phases du moteur peut conduire à une fausse manoeuvre);
- pouvoir vérifier facilement l'état des différents capteurs (contrôle automatique);
- guider l'opérateur par des check-lists indiquant les opérations à exécuter afin de vérifier les règles de sécurité élémentaires à tout instant;
- éviter le basculement de la grue dû à un affaissement du sol, assurer la stabilité aussi bien au repos que dans des conditions normales de charge et de fonctionnement (par exemple, en mesurant la stabilité au moyen d'un niveau);
- empêcher un nombre de rotations de la flèche pouvant mener à la rupture des câbles d'alimentation;
- pouvoir transgresser le système de sécurité mais en gardant trace - au moyen d'une boîte noire - de l'heure, des manoeuvres réalisées et de l'identité du grutier de manière à responsabiliser ce dernier.

Concernant ce dernier aspect, il est important de faire remarquer qu'en règle générale, aucune action de l'opérateur ne doit être réellement interdite : le système de commande doit offrir la possibilité de fonctionner en mode dégradé, c'est-à-dire laisser l'opérateur libre de forcer la grue pour quelques opérations délicates. Dans une telle situation, un signal sonore persiste jusqu'à ce que la machine soit revenue dans des conditions normales de sécurité.

Délimitation d'un sous-ensemble des objectifs

Ce qui suit constitue l'ensemble des besoins fonctionnels du prototype implémenté dans l'automate chargé de contrôler la grue; prototype puisque seul un sous-ensemble des fonctionnalités requises par le futur système envisagé est repris. Les besoins non fonctionnels tels que les contraintes de fiabilité, de qualité et de disponibilité du matériel, les conditions de travail sur chantier (température, humidité, manoeuvres brusques) et les aspects économiques ne sont pas pris en considération.

Tout d'abord, nous supposerons la grue posée sur un bon sol, correctement montée et en parfait état de marche : aucun capteur défectueux, câbles en bon état, aucun problème au niveau des moteurs. Autrement dit, nous ne nous préoccuperons pas du choix de l'emplacement de la grue, de l'installation de la voie, de l'automontage de la grue, ni du contrôle automatique des capteurs, de la mesure de la surintensité des moteurs ou des perturbations de tension, c'est-à-dire tout ce qui concerne l'état physique des divers éléments et leur alimentation.

Précisons que les divers organes des appareils de levage doivent être maintenus constamment en parfait état d'entretien et de sécurité de fonctionnement [RGPT96]. A ce sujet, un organisme agréé effectue des visites tous les trimestres et l'isolement des installations électriques fait l'objet chaque semaine de vérifications par un électricien qualifié [CNAC80].

Le basculement de la grue lors de son montage ou d'un éventuel affaissement du sol, ne sera donc pas repris dans la suite de ce document, ce problème faisant appel à une technique particulière de mesure n'ayant pas encore fait l'objet d'un examen approfondi. Rappelons à ce sujet la responsabilité de l'équipe ayant monté la grue de s'être assurée de la bonne stabilité du sol en faisant appel, par un exemple, à un organisme agréé chargé de la mesure de celle-ci.

Le problème de la surveillance du nombre de rotations complètes de la flèche pouvant mener à la rupture des câbles ne sera pas pris en compte au sein du prototype, ce problème pouvant être résolu techniquement par l'intermédiaire d'une bague de contact pour l'alimentation du moteur ou par l'installation d'un système particulier de capteurs (compte-tours) afin de détecter l'accomplissement d'un tour complet de la flèche ainsi que le sens pour le compte ou le décompte du nombre total de tours effectués.

Le projet de la boîte noire destinée à enregistrer toute manoeuvre de la grue considérée comme dangereuse, sera également écarté au cours de cette étude, celui-ci requérant un élément auxiliaire à l'automate pour l'enregistrement fiable et "hermétique" des divers indices témoins. Toutefois, le principe du fonctionnement en mode dégradé est pris en considération au sein de notre prototype.

Evidemment, tous les mouvements classiques de la grue énoncés plus haut sont considérés. A ce sujet, des contraintes de temporisation empêchent l'opérateur d'inverser brutalement le sens de tout moteur. Pour notre prototype, seul le moteur de levage fera l'objet d'un changement de vitesse, les autres moteurs n'offrant pas à l'utilisateur la possibilité de modifier leur vitesse de rotation. Le moteur de levage comporte en fait trois vitesses dont la manipulation doit se plier à des contraintes de temporisation.

De plus, dans la pratique, afin d'éviter les problèmes de rebondissement, la vitesse est limitée suivant la charge soulevée : par exemple, le moteur de levage ne peut passer en troisième que si la charge au crochet vaut la moitié de la charge maximale admissible [ZEI95a]. Cet aspect ne sera pas abordé dans la suite de ce document mais soulignons son importance dans la perspective d'une réelle mise en oeuvre du système.

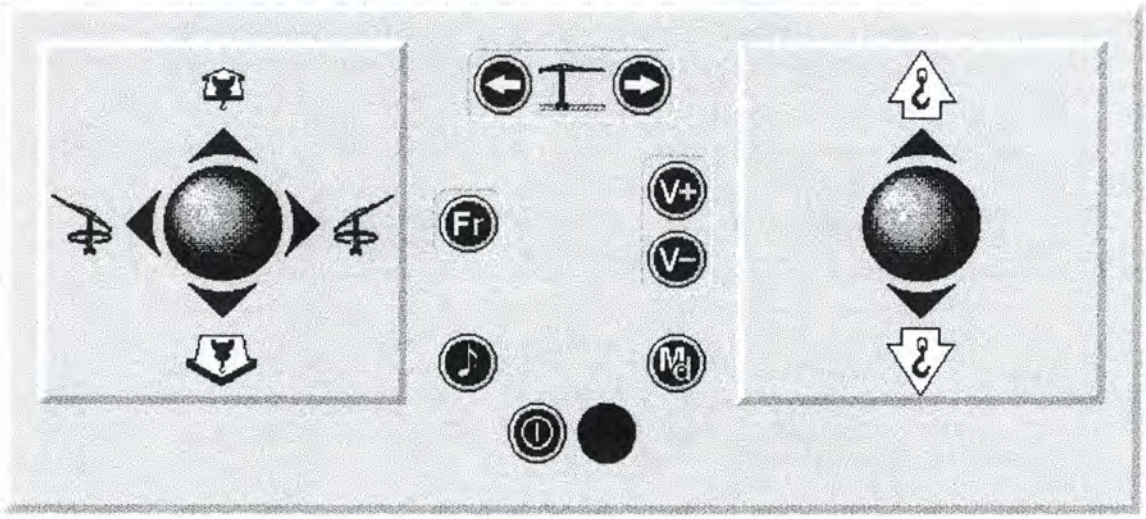
Autre détail rencontré dans la pratique, avant la mise hors service de la grue, le chariot doit être ramené dans une zone de sécurité adjacente au mât pouvant être paramétrée. Suite à l'utilisation de capteurs de fin de course "binaires", nous ne considérerons qu'une seule position de sécurité du chariot : à savoir, la position extrême du chariot contre le mât, détectée par le limiteur de course.

Synthèse des besoins fonctionnels du prototype réalisé

Ci-dessous, les différents stimuli pris en considération par le système de commande suivis des conditions de perception des commandes et de manipulation des divers organes de la machine.

Les commandes de l'opérateur

Le préposé à la manoeuvre agit sur l'engin à partir du pupitre de commande illustré par la figure 1.3.



~ Figure 1.3 : Pupitre de commande de la grue ~

Pour notre prototype, les différentes actions de l'opérateur devant être gérées par le système sont les suivantes :

⇨ Mise en/hors service de la grue.
⇨ Enclenchement de l'arrêt d'urgence.
⇨ Appui sur l'avertisseur sonore.
⇨ Passage en mode dégradé.
⇨ Enclenchement du frein de giration afin d'immobiliser la flèche.
⇨ Relâchement du frein de giration afin de libérer la flèche.

⇨ Avance du chariot (vers l'extrémité de la flèche).
⇨ Recul du chariot (vers le mât de la grue).

⇨ Avance du châssis de la grue sur les rails.
⇨ Recul du châssis de la grue sur les rails.

⇨ Rotation de la flèche vers la droite.
⇨ Rotation de la flèche vers la gauche.

⇨ Montée du crochet de levage.
⇨ Descente du crochet de levage.

⇨ Augmenter la vitesse du moteur de levage.
⇨ Diminuer la vitesse du moteur de levage.

Seul le frein du moteur de giration est commandé par l'opérateur : tout dispositif de freinage d'un autre moteur est mécaniquement activé dès que la commande de mouvement est interrompue [RGPT96].

Précisons que les orientations citées ci-dessus (avance, recul, gauche et droite) sont définies en prenant le grutier comme point de repère sauf pour le mouvement de translation du châssis.

En effet, pour le grutier, la direction varie suivant l'angle que fait la flèche avec les rails de la voie : si la flèche est parallèle à la voie, la translation se traduit pour le grutier par un mouvement vers l'avant ou vers l'arrière tandis que si la flèche est perpendiculaire à la voie, la translation se traduit pour le grutier par un mouvement vers la gauche ou la droite. Nous considérerons comme avance de la grue, le mouvement l'approchant d'une des extrémités considérée comme le début de la voie. Respectivement, le mouvement de recul est celui entraînant la grue vers l'extrémité considérée comme la fin de la voie.

Notons que les commandes des différents mouvements peuvent être actionnées dans un ordre quelconque voire simultanément par l'opérateur.

Informations délivrées par la grue

Ces informations sont fournies par les divers capteurs situés sur la grue :

⇨ Limite de déplacement avant du chariot (en bout de flèche).
⇨ Limite de déplacement arrière du chariot (position de sécurité, près du mât).
⇨ Limite de translation avant de la grue.
⇨ Limite de translation arrière de la grue.
⇨ Limite de montée du crochet.
⇨ Limite de descente du crochet.
⇨ Vitesse maximale du vent atteinte.
⇨ Charge maximale atteinte.
⇨ Moment (couple de charge) maximum atteint.

Mise en/hors service de la grue

La grue est mise en service par un appui sur la touche ON/OFF; cette action n'est perçue que si la grue est hors service. Si c'est le cas, elle provoque la mise sous tension de l'ensemble des éléments de l'engin.

Un second appui sur la même touche entraîne la coupure d'alimentation de la grue à condition que celle-ci soit en service et dans une situation telle qu'elle puisse être éteinte : en effet, le frein de giration ne peut être enclenché, le chariot doit avoir été déplacé au pied du mât, le mode dégradé ne doit pas être actif, le crochet doit avoir été relevé et ne peut supporter une charge supérieure à 20% de la charge maximale admissible de l'engin.

L'arrêt d'urgence provoque également la coupure d'alimentation du système mais sans se plier aux contraintes précédentes : cette commande est prioritaire sur toutes les autres. Evidemment, ce dispositif n'a aucun effet si la grue est déjà éteinte.

Frein de giration

Le frein de giration ne peut être enclenché par la commande correspondante que s'il est inactif, que si la grue est en service et le moteur de giration à l'arrêt. Une fois en marche, la même commande le désactive : la flèche est à nouveau libre de son mouvement de rotation. Rappelons que l'on ne sait pas éteindre la grue tant que ce frein est en service.

Passage en mode dégradé

Ce mode permet à l'opérateur de forcer exceptionnellement une manoeuvre, amenant ainsi la grue dans une situation violant les consignes de sécurité. Les contraintes telles que les limitations de la charge soulevée, du moment maximum et de la vitesse du vent sont alors oubliées lors de la mise en mouvement de certains organes de la grue. Dès que le système détecte que tout est à nouveau normal, ce mode disparaît.

Cela implique que, pour travailler en mode dégradé, l'opérateur doit maintenir la touche du mode dégradé enfoncée tout en amenant le système dans la situation violant les consignes de sécurité. Une fois une telle situation atteinte, la touche relative au mode dégradé peut être relâchée, ce mode restant actif jusqu'à ce que la grue revienne dans une situation acceptable.

En fait, ce retour en mode de fonctionnement normal s'effectue si la grue est en service, le mode dégradé est en cours, la charge au crochet et le moment ne dépassent pas leurs valeurs limites, le crochet n'a pas atteint son niveau le plus bas et le chariot ne se situe pas en bout de flèche.

De même que pour le frein de giration, la grue ne peut être éteinte tant que ce mode est actif. En d'autres termes, ce mode n'est jamais présent au démarrage de la grue.

Avertisseur sonore

Ce dernier se met en marche suite à la demande de l'opérateur, lorsqu'on travaille en mode dégradé ou encore dès que l'anémomètre perçoit une vitesse de vent supérieure à la limite assurant la stabilité de la grue. Tant qu'aucune de ces conditions n'est présente, l'avertisseur reste silencieux.

Déplacement du chariot

La commande d'avance du chariot n'est perçue que si la grue est en service, que si le chariot n'est pas en fin de course (à l'avant) et que si la charge et le moment mesurés

sont inférieurs aux valeurs maximales à moins que le mode dégradé soit actif : dans ce cas, ces deux contraintes ne sont pas prises en considération.

La commande de recul du chariot est identique à la précédente sauf que le chariot ne doit pas être contre le mât (limite arrière). Que l'on soit en mode dégradé ou non, les contraintes dues à la charge et au moment ne sont jamais prises en considération : le recul du chariot est considéré comme un mouvement de sécurité.

Ajoutons que l'on ne peut changer immédiatement le sens de déplacement du chariot : avant qu'une temporisation ne se soit écoulée, on ne peut inverser le sens de rotation du moteur. Cette temporisation est destinée à assurer que le moteur s'arrête pour freiner suffisamment avant d'alimenter le moteur en sens inverse.

Translation de la grue

Les commandes de translation du châssis de la grue sur rails ne sont perçues que si la grue est en service, que si le châssis n'est pas en fin de course et que si la charge, le moment et la vitesse du vent sont inférieurs aux valeurs maximales à moins que le mode dégradé soit actif (dans ce cas, ces trois contraintes ne sont pas prises en considération).

Ajoutons que l'on ne peut changer immédiatement le sens de translation de la grue : avant qu'une temporisation ne se soit écoulée, on ne peut inverser le sens de rotation du moteur. Cette temporisation est destinée à assurer que le moteur s'arrête pour freiner suffisamment avant d'alimenter le moteur en sens inverse.

Giration de la flèche

Les commandes de giration de la flèche ne sont perçues que si la grue est en service et que si le frein de giration n'est pas enclenché. De même que pour les moteurs précédents, on ne peut inverser le sens de rotation du moteur avant qu'une temporisation ne se soit écoulée.

Mouvement de levage

La commande de montée du crochet n'est perçue que si la grue est en service, que si le crochet n'est pas au sommet et que si la charge et le moment mesurés sont inférieurs aux valeurs maximales à moins que le mode dégradé soit actif (dans ce cas, ces deux contraintes ne sont pas prises en considération).

La commande de descente du crochet est identique à la précédente sauf que le crochet ne doit pas être à son niveau le plus bas. Les contraintes dues à la charge et au moment ne sont jamais prises en considération : la descente du crochet est considérée comme un mouvement de sécurité.

Le moteur impliqué dans ce mouvement de levage peut fonctionner à trois vitesses différentes. De même que pour l'inversion du sens de rotation, tout changement de

vitesse doit répondre à une contrainte de temporisation afin de permettre au moteur d'atteindre sa vitesse de régime avant de passer à une vitesse supérieure ou de ralentir suffisamment, celui-ci étant entraîné par son inertie.

Ajoutons que le fait de relâcher la commande de montée ou de descente du crochet réinitialise le moteur en première vitesse, ceci pour faciliter le grutier dans sa tâche de mise à niveau d'une charge. Des temporisations assurent que le redémarrage du moteur ne peut se faire que lorsque ce dernier a suffisamment freiné.

Chapitre

2

La spécification formelle du système

Destinées à tout lecteur non-initié au langage **Albert II**, les notes suivantes rappellent brièvement quelques concepts fondamentaux du langage, la structure générale d'une spécification ainsi que les conventions utilisées par la représentation graphique des déclarations. Pour un aperçu plus détaillé et une explication des diverses contraintes, se référer à l'annexe 1.

Quelques concepts du langage Albert II

Le langage **Albert II** permet de décrire le comportement admissible d'un système composite. Ses principales caractéristiques sont les suivantes :

- ⇨ il est orienté agent : forme d'orienté objet où l'on met l'accent sur la coopération entre les différents agents;
- ⇨ il supporte deux styles de spécification : le style déclaratif et le style opérationnel;
- ⇨ il permet d'exprimer des contraintes temps-réel;
- ⇨ il offre une syntaxe graphique pour les déclarations.

Une spécification **Albert II** est structurée hiérarchiquement en agents et sociétés (regroupement d'agents voire de sociétés). Elle est composée d'au moins un agent et tout agent ou société autre que la racine doit faire partie d'une et une seule société.

Le langage **Albert II** est constitué de constructions pour introduire des déclarations et exprimer des contraintes. Les déclarations introduisent le vocabulaire de l'application considérée et les contraintes (propositions logiques) identifient les comportements possibles du système composite et écartent ceux non désirés. Seules les déclarations peuvent être exprimées par la syntaxe graphique.

Une spécification **Albert II** est constituée de plusieurs types de contraintes. Elles sont classées sous onze en-têtes groupées en trois familles (fig. 2.1).

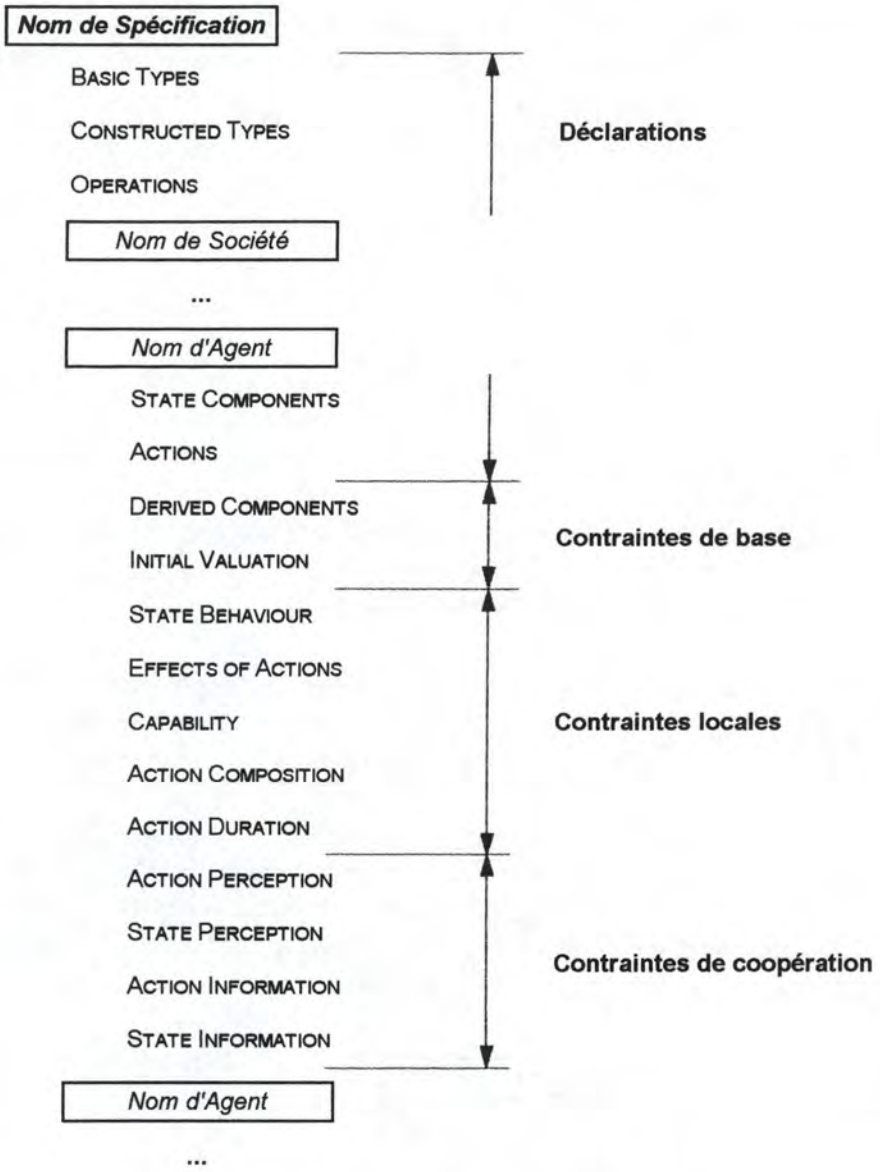
Déclarations sous forme graphique

Rappel

La déclaration graphique d'un agent décrit, au sein d'un parallélogramme, la liste de ses composants d'état et des actions qu'il peut exécuter. Les composants d'état sont représentés à l'aide de rectangles et les actions au moyen de rectangles aux coins arrondis. Les propriétés des composants d'état et les liens d'importation et d'exportation entre agents y sont également représentés.

Les composants d'état

Un composant d'état est représenté graphiquement par un rectangle, son nom et son type sont indiqués respectivement dans la partie supérieure et inférieure.



~ Figure 2.1 : Squelette d'une spécification **Albert II** ~

Les composants d'état peuvent être considérés comme des individus ou des populations. Dans notre spécification, les composants d'état de nos agents sont uniquement des individus : leur boîte est représentée en trait pointillé (cf. Annexe 1 pour les autres représentations).

Ils sont constants ou varient au cours du temps : les boîtes des composants constants ont leur contour en trait gras (exemples : ChargeMax, MomentMax, VitesseVentMax et PourcentageChargeMax de l'agent "Système", cf. infra).

La valeur d'un composant d'état peut être dérivée des valeurs d'autres composants d'état du même agent. Dans ce cas, une flèche est tracée du composant d'état vers celui qui en dépend (exemples : ChargeAtteinte, MomentAtteint, VitesseVentAtteinte de l'agent "Système", cf. infra).

Un composant d'état peut être rendu visible à un autre agent ou à une société : une flèche est tracée du composant d'état exporté vers l'extérieur du parallélogramme, le bout de la flèche pointant sur le nom de l'agent ou de la société vers lequel le composant est exporté. Cette vue graphique permet aussi de voir les composants importés visibles par l'agent : ceux-ci figurent à l'extérieur du parallélogramme de l'agent, sur les côtés indiquant qu'ils ne font pas partie de cet agent.

Les actions

Une action exécutée par un agent est représentée graphiquement par un rectangle à coins arrondis, son nom et ses arguments sont indiqués respectivement dans la partie supérieure et inférieure du rectangle.

De même que pour les composants d'état, le mécanisme statique d'exportation permet de rendre des actions visibles à d'autres agents : la représentation graphique est identique à celle utilisée pour les composants d'état.

Une action peut être restreinte : ses occurrences ne peuvent survenir que dans le contexte d'une action composée. Graphiquement, la boîte d'une action restreinte est représentée en trait gras mais si cette action est restreinte dans un autre agent que celui dans lequel elle est déclarée, sa boîte ne figure sous cette forme que du côté de l'agent dans lequel elle est restreinte. En d'autres mots, la restriction n'apparaît que du côté de l'agent concerné (cf. fig. 2.5 : Déclaration graphique de l'agent "Système").

Déclarations graphiques du système de contrôle de la grue

L'approche suivie étant celle des systèmes composites, notre spécification Albert II comporte non seulement la spécification du système de commande de la grue mais aussi celle des composants avec lesquels il interagit [ZEI95b].

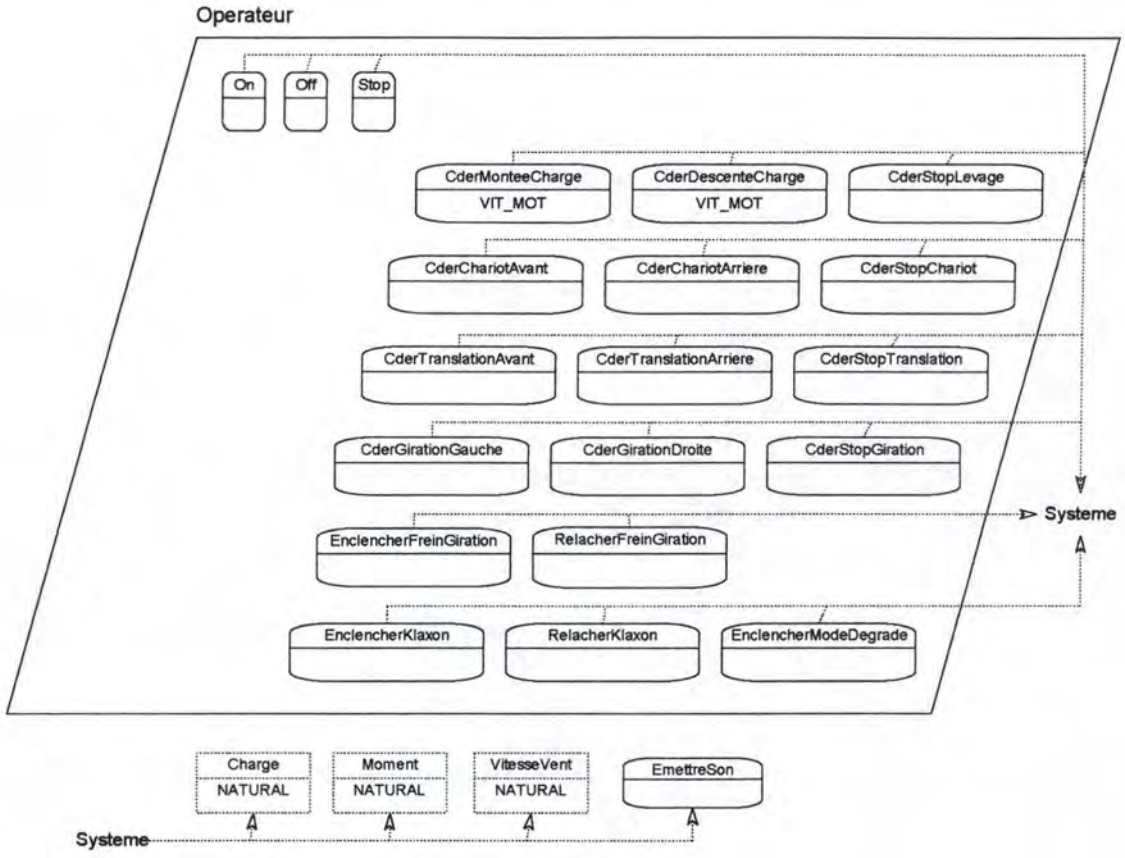
Trois agents seront considérés dans cette spécification :
- l'agent OPERATEUR, donnant des ordres et percevant des informations (fig. 2.3);
- l'agent ENVIRONNEMENT dans lequel la grue et l'opérateur évoluent (fig. 2.4).
- l'agent SYSTEME, constitué des capteurs et actionneurs (fig. 2.5);

Ces trois agents sont regroupés dans la société CHANTIER SECURISANT (fig. 2.2).

~ Figure 2.2 : Société "Chantier Sécurisant" ~

L'agent Opérateur

Cet agent (fig. 2.3) n'est constitué que d'une déclaration d'actions sans aucun composant d'état ni aucune contrainte : en effet, comme c'est souvent le cas pour des agents "humains", l'état n'est pas décrit et le comportement est laissé libre; en outre, les actions de l'utilisateur sont toutes perçues par l'agent Système.



~ Figure 2.3 : Déclaration graphique de l'agent "Opérateur" ~

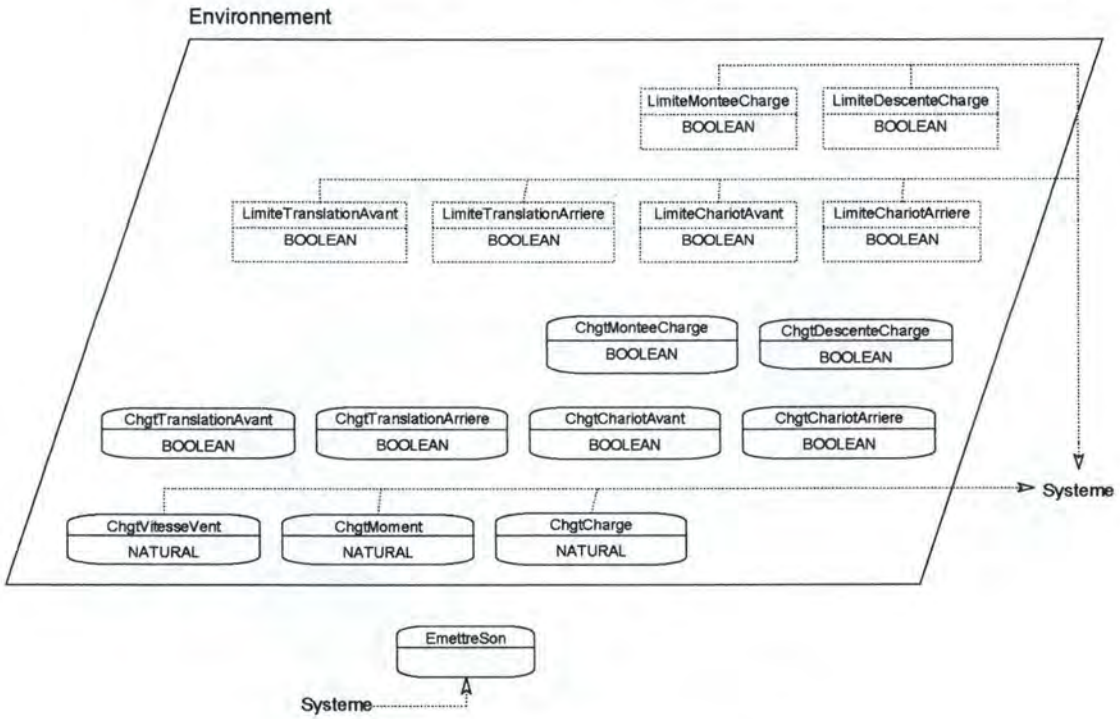
- **On**, **Off** : l'opérateur met en service la grue et l'arrête via l'interrupteur prévu à cet effet;
- **Stop** : l'opérateur a actionné l'arrêt d'urgence;
- **CderMontéeCharge(Vit_mot)**, **CderDescenteCharge(Vit_mot)** : ces deux actions surviennent lorsque le grutier commande le déplacement vertical de la charge, ces actions ont comme argument la vitesse du moteur de levage (vitesse n°1, 2 ou 3);
- **CderChariotAvant**, **CderChariotArrière** : correspondent à la commande de déplacement du chariot le long de la flèche en prenant le grutier comme point de repère ("vers l'arrière" signifie "vers le mât de la grue");
- **CderTranslationAvant**, **CderTranslationArrière** : correspondent à la commande de déplacement de la grue le long des rails, l'avant et l'arrière correspondant aux extrémités de la voie considérées comme début et fin;
- **CderGirationGauche**, **CderGirationDroite** : correspondent à la commande de rotation de la flèche;

- **CderStopLevage**, **CderStopChariot**, **CderStopTranslation**, **CderStopGiration** : chacune de ces quatre actions correspond à l'arrêt de la commande du mouvement de la grue correspondant;
- **EnclencherFreinGiration**, **RelâcherFreinGiration** : l'opérateur active/désactive le frein;
- **EnclencherKlaxon**, **RelâcherKlaxon** : l'opérateur actionne ou relâche l'avertisseur;
- **EnclencherModeDégradé** : cette action signifie que l'opérateur ordonne le passage en mode dégradé, le retour en mode normal est pris en charge par le système.

Remarque : seules les actions commandant le levage de la charge ont un argument concernant la vitesse du moteur puisque, pour notre système, les autres mouvements ne font pas l'objet d'un changement de vitesse.

L'agent Environnement

Cet agent, dont la déclaration graphique apparaît ci-dessous (fig. 2.4), renferme les événements survenant lors de l'utilisation de la grue autres que ceux occasionnés par les actions du grutier. Plus précisément, les actions suivantes surviennent lorsqu'un événement particulier est capté à partir de la grue : la vitesse du vent se modifie; le chariot, la grue ou le crochet arrive en fin de course.



~ Figure 2.4 : Déclaration graphique de l'agent "Environnement" ~

Actions

- **ChgtVitesseVent(Natural), ChgtCharge(Natural), ChgtMoment(Natural)** : ces actions sont communiquées à l'agent Système et lui permettent de connaître la valeur entière positive de chacun de ces paramètres à tout instant;
- **ChgtMontéeCharge(Boolean), ChgtDescenteCharge(Boolean)**
- **ChgtTranslationAvant(Boolean), ChgtTranslationArrière(Boolean)**
- **ChgtChariotAvant(Boolean), ChgtChariotArrière(Boolean)**

Ces six dernières actions concernant les divers mouvements de la grue ont pour effet de modifier les composants d'état internes à l'agent Environnement.

Composants d'état

- **LimiteMontéeCharge, LimiteDescenteCharge**
- **LimiteTranslationAvant, LimiteTranslationArrière**
- **LimiteChariotAvant, LimiteChariotArrière**

Ces composants booléens indiquent les positionnements en fin de course des divers organes de la machine et sont tous perçus par l'agent Système.

Notons la présence de contraintes d'information de type *XK* (*exclusive knowledge*) précisant que le système détecte toujours les limites de fin de course ainsi que les modifications de la vitesse du vent, de la charge et du moment. En quelque sorte, cela traduit l'hypothèse de la fiabilité de l'ensemble des capteurs.

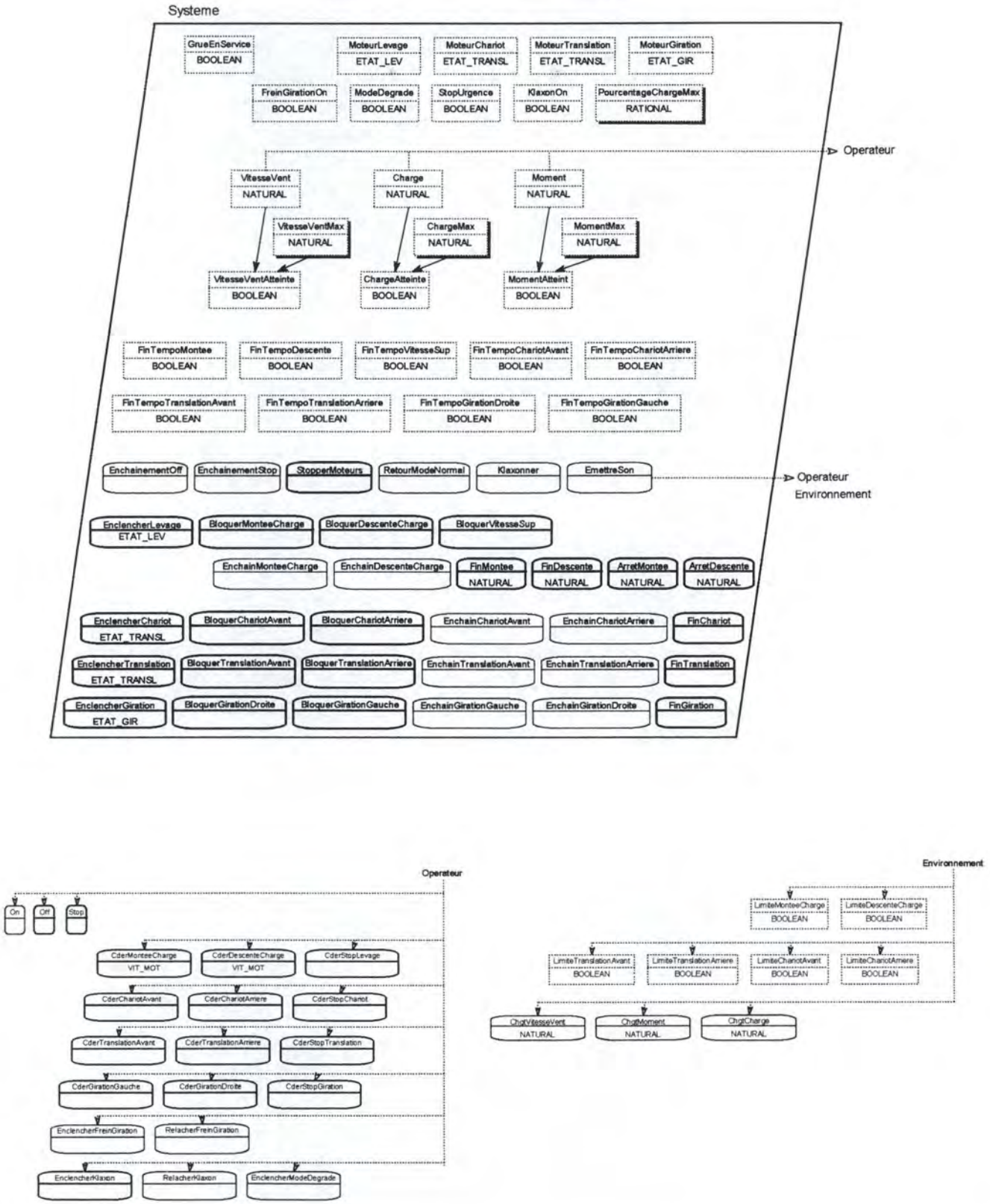
L'agent Système

Agent fondamental de notre spécification Albert, il contient de nombreux composants d'état, actions et contraintes relatifs au fonctionnement des moteurs et autres dispositifs de la grue (fig. 2.5).

Regroupons pour commencer l'ensemble des composants d'état suivant leur type en les accompagnant de leur signification.

Composants d'état

- **GrueEnService** : indicateur booléen vrai si la grue est sous tension;
- **MoteurLevage, MoteurChariot, MoteurTranslation, MoteurGiration** : ces composants indiquent l'état des moteurs c'est-à-dire leur sens de déplacement ou s'ils sont à l'arrêt ainsi que la vitesse pour le moteur de levage;
- **FreinGirationOn, ModeDégradé, StopUrgence, KlaxonOn** : booléens témoins de la présence ou de l'absence du signal correspondant;
- **VitesseVent, Charge, Moment** : valeurs entières positives mesurées et transmises par l'environnement;
- **VitesseVentMax, ChargeMax, MomentMax** : valeurs constantes des seuils limites assurant un fonctionnement de la grue sans danger;



~ Figure 2.5 : Déclaration graphique de l'agent "Système" ~

- **ChargeAtteinte**, **MomentAtteint**, **VitesseVentAtteinte** : indicateurs booléens dont les valeurs résultent de la comparaison des valeurs mesurées aux seuils limites;
- **PourcentageChargeMax** : pourcentage maximum autorisé de la charge suspendue au crochet lors de la mise hors service de la grue (nombre rationnel constant < 1);
- 9 composants **FinTempo...** : chacun de ces indicateurs booléens signale la fin d'une temporisation destinée à assurer une bonne manipulation des différents moteurs; ils empêchent le grutier d'actionner un moteur en sens inverse immédiatement après son arrêt ou d'augmenter la vitesse du moteur de levage sans qu'il ait atteint sa vitesse de régime précédente.

Actions

- **EnchaînementOff**, **EnchaînementStop** : ces deux actions composées agissent lorsque l'opérateur a commandé l'arrêt de la grue que ce soit par la commande Off ou par un arrêt d'urgence; l'action restreinte **StopperMoteurs** est alors activée.
- **RetourModeNormal** : cette action est exécutée automatiquement dès que le système détecte que le mode dégradé n'est plus nécessaire, les contraintes de sécurité étant à nouveau vérifiées.
- **Klaxonner** : lorsque le grutier maintient le klaxon, cette action composée a pour mission d'activer l'action **EmettreSon**. Cette dernière action est également exécutée par le système en cas de fonctionnement en mode dégradé ou lorsque le vent atteint sa vitesse maximale.
- **EnclencherLevage(Etat_Lev)**, **BloquerMontéeCharge**, **BloquerDescenteCharge**, **BloquerVitesseSup**, **EnchaînMontéeCharge**, **EnchaînDescenteCharge**, **FinMontée(Natural)**, **FinDescente(Natural)**, **ArrêtMontée(Natural)**, **ArrêtDescente(Natural)**
- **EnclencherChariot(Etat_Transl)**, **BloquerChariotAvant**, **BloquerChariotArrière**, **EnchaînChariotAvant**, **EnchaînChariotArrière**, **FinChariot**
- **EnclencherTranslation(Etat_Transl)**, **BloquerTranslationAvant**, **BloquerTranslationArrière**, **EnchaînTranslationAvant**, **EnchaînTranslationArrière**, **FinTranslation**
- **EnclencherGiration(Etat_Gir)**, **BloquerGirationDroite**, **BloquerGirationGauche**, **EnchaînGirationGauche**, **EnchaînGirationDroite**, **FinGiration**

La plupart des actions citées ci-dessus assurent le fonctionnement correct des moteurs de la grue. Les actions "Enclencher..." sont responsables de la mise en marche des moteurs suite à la commande de l'opérateur. Les actions "Bloquer..." sont destinées à empêcher l'utilisateur d'inverser le sens de rotation de tout moteur tant que celui-ci n'a pas suffisamment freiné. Les action composées "Enchaîn..." et "Fin..." gèrent le tout, elles contrôlent la succession des occurrences d'actions.

Pour de plus amples explications concernant les différents types de contraintes et leur contenu se référer au chapitre 4 et à l'annexe 2.

Chapitre

3

La technologie des automates programmables

Généralités sur les automates programmables industriels

Définition

Les A.P.I. (Automate Programmable Industriel)[1] sont apparus aux Etats-Unis vers 1969 où ils répondaient aux désirs des industries automobiles de développer des chaînes de fabrication automatisées qui pourraient suivre l'évolution des techniques et des modèles fabriqués [MLE79]. L'A.P.I. s'est ainsi substitué aux armoires à relais en raison de sa souplesse mais également parce que, dans les automatismes complexes, les coûts de câblage et de mise au point devenaient trop élevés.

L'A.P.I. est une machine programmable destinée à automatiser les tâches les plus nombreuses de l'industrie, permettant l'automatisation des processus logiques ("tout ou rien") mais également analogiques. Contrairement à la solution informatique, l'A.P.I. ne requiert pas de formation spécialisée en informatique : les personnes habituées aux logiques à relais s'adaptent rapidement et sans difficulté à celui-ci.

> *"Un automate programmable industriel est une machine électronique, programmable par un personnel non informaticien et destinée à piloter, en ambiance industrielle et en temps réel, des procédés logiques séquentiels."* [MLE79]

Concrètement, les automates programmables se placent à l'interface de l'automatique et de l'informatique. Ce concept hybride d'"automate" au sens de l'automatique, "programmable" selon la terminologie informatique, est l'aboutissement d'un cheminement intellectuel convergent de ces deux disciplines. Leur caractéristique essentielle est leur fonctionnement cyclique, très rapide, qui leur confère une approche originale des traitements en "temps-réel" d'où l'évidence de leur utilisation en tant qu'outil de commande de procédés [MLE79].

A ce niveau, ils sont plus aisément mis en oeuvre que bien d'autres moyens, moyens supérieurs tels que l'informatique, moyens inférieurs tels la micro-informatique et la logique câblée. En effet, ils viennent combler une lacune pour une gamme d'automatismes considérés comme trop importants pour relever des techniques de la logique câblée classique et pour lesquels une solution informatique s'avère trop onéreuse.

Sa nature programmable lui confère une souplesse exceptionnelle tandis que sa conception le rend parfaitement adapté aux contraintes, parfois extrêmement sévères, de l'environnement industriel perturbé dont les types d'agression sont les suivants :

- l'environnement physique et mécanique : température, humidité, vibrations, chocs;
- la pollution chimique : gaz corrosifs, vapeurs d'hydrocarbures, poussières métalliques (aciéries) ou minérales (cimenteries);
- les perturbations électriques : les parasites d'origine électrostatique et les interférences électromagnétiques qui imposent un isolement efficace des entrées/sorties.

Finalement, notons que, de nos jours, l'A.P.I. intervient non seulement dans la commande individuelle des machines et des processus mais participe également à la supervision, à la coordination des tâches ainsi qu'à la gestion de la production grâce à ses capacités de dialogue.

---

[1] En anglais : *P.L.C. (Programmable Logic Controller)*

Objectif et structure d'un automatisme

L'automatisation d'un procédé (machine ou ensemble de machines) consiste à en assurer la conduite par un dispositif technologique. L'intervention d'un opérateur est souvent nécessaire pour assurer un pilotage global du procédé, pour surveiller les installations et reprendre en commande manuelle tout ou partie du système (fig. 3.1).

Mécanismes d'entrée - Capteurs

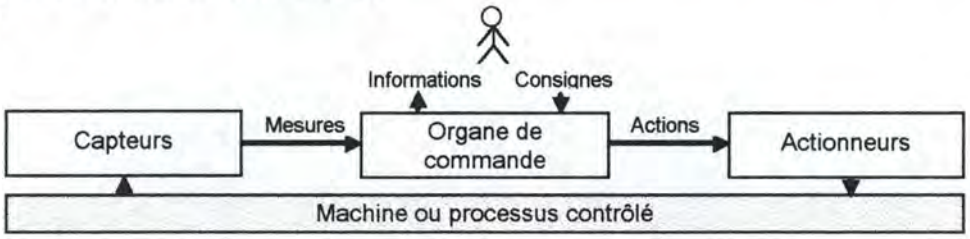
Actionnés manuellement par l'opérateur (boutons-poussoirs), commandés automatiquement par le processus (capteurs de fin de course) ou mesurant des paramètres du procédé, ils prélèvent l'information et généralement en transforment la nature physique.

Organe de commande

Il élabore les actions à partir des mesures et des consignes selon la loi de commande de l'automatisme. Il peut s'agir, par exemple, d'un automate programmable contenant diverses instructions pour activer ou désactiver les mécanismes de sortie suivant l'état des mécanismes d'entrée, et en accord avec la logique du programme.

Mécanismes de sortie - Actionneurs

Encore appelés actuateurs ou effecteurs, ils transmettent les ordres de commande au procédé, transmission généralement accompagnée d'une transformation de la nature physique de l'information et d'un apport de puissance (alimentation des moteurs électriques du procédé, entre autres).



~ Figure 3.1 : Structure d'un automatisme ~

Degrés et niveaux d'automatisation

On distingue trois **degrés d'automatisation** d'un système [MLE79] :

1. la surveillance de grandeurs : l'organe de contrôle acquiert les informations, les analyse et produit des bilans (fonction passive à l'égard du procédé);
2. le mode guide opérateur : traitements plus élaborés, automatisme dit "en boucle ouverte" car il ne réagit pas directement sur le procédé mais propose aux responsables du site, des actions pour le conduire selon un critère donné (ces derniers ferment la boucle);
3. la commande proprement dite, en boucle fermée : elle correspond à l'automatisation complète de certaines fonctions (acquisition des informations, traitement de celles-ci puis action sur le procédé). L'homme, exclu de la conduite, surveille le système et intervient en cas d'incident pour reprendre le pilotage manuel du procédé.

Les fonctions précédentes sont simples ou complexes selon le procédé auquel elles sont assignées, aussi introduisons à présent les **niveaux d'automatisation** [MLE79] :

1. le niveau élémentaire concerne une machine simple (ou partie de machine) : par exemple, fonctions de sécurité, d'enchaînement de tâches, de surveillance des temps morts, de positionnement de pièces, etc.
2. le second niveau concerne un ensemble de machines simples ou une machine plus complexe : c'est le domaine classique de l'automatisme industriel.
3. le troisième niveau concerne un procédé ou un atelier complet : il s'agit d'un problème complexe pouvant englober aussi bien des paramètres techniques que des variables économiques.

Solutions technologiques à l'automatisation

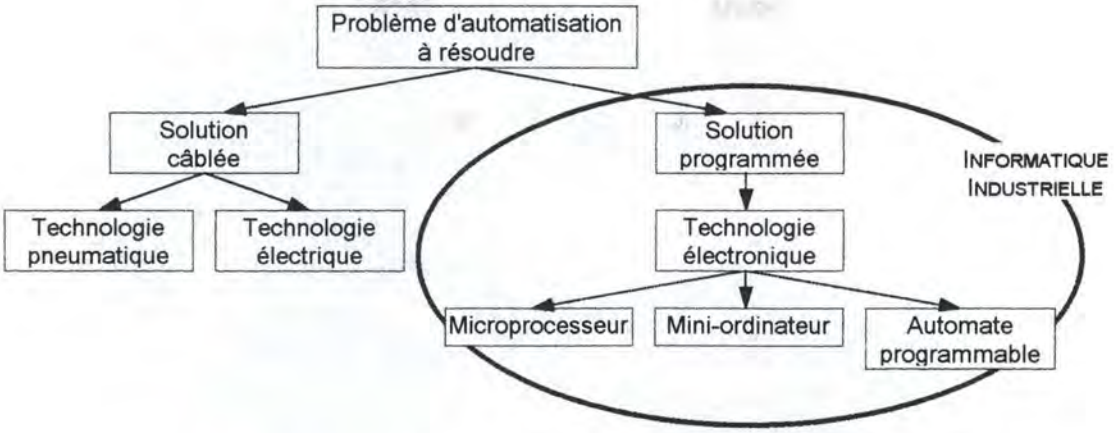
L'organe de commande d'un système peut être réalisé à l'aide de plusieurs outils technologiques regroupés en deux principales catégories : les solutions câblées et les solutions programmées (fig. 3.2) [MLE79].

En ce qui concerne les solutions câblées, chaque équation booléenne constituant la loi de commande des procédés logiques est réalisée physiquement par un circuit intégré. La principale caractéristique de cette logique câblée est qu'elle permet de traiter en parallèle les équations représentant la loi de commande en sollicitant simultanément l'ensemble des opérateurs logiques offerts par le circuit.

Les logiques programmables sont, quant à elles, des outils informatiques. L'informatique industrielle résout des problèmes de nature industrielle en conjuguant les théories de l'automatique et les moyens de l'informatique. Contrairement aux logiques câblées, les logiques programmables offrent une approche séquentielle en précisant à un seul processeur, à l'aide d'instructions appropriées, quelle fonction il doit effectuer à un instant donné et sur quels signaux.

A chaque instant, le processeur exécute une et une seule instruction ce qui implique que les données doivent être mémorisées pour être disponibles lorsqu'elles devront être traitées. Les ordres commandant les fonctions du processeur sont également placés en mémoire et présentés successivement à ce dernier. Après exécution d'une instruction, le résultat contenu dans le registre accumulateur est transféré lui aussi vers la mémoire. Il y restera en tant que résultat intermédiaire ou sera transféré ultérieurement à l'extérieur de la machine s'il concerne une sortie.

Une horloge interne synchronise l'enchaînement des instructions, sa grande rapidité permet d'aborder toutes les fonctions logiques en temps-réel.



~ Figure 3.2 : Solutions technologiques à un problème d'automatisation ~

Principe de fonctionnement

Une caractéristique originale des automates programmables est le fonctionnement cyclique de l'unité centrale.

Durant chaque cycle, l'automate scrute l'état des mécanismes d'entrée, exécute son programme sous le contrôle d'une horloge et modifie l'état des sorties en conséquence. Les échanges sont implicites : les entrées/sorties ne nécessitent pas de programmation de la part de l'utilisateur et ne lui apparaissent pas, ce qui constitue une autre caractéristique originale des A.P.I. La durée de cycle[2] dépend du contenu, de la longueur du programme et du nombre d'entrées/sorties connectées ainsi que de la vitesse intrinsèque de l'automate, mais si elle vient à dépasser une valeur limite, le "chien de garde" (*watchdog timer*) provoque l'arrêt du processeur.

En prenant l'hypothèse d'un traitement purement logique, avec pour seul saut le retour à l'instruction n° 1, on rencontre encore différents types de cycles suivant les A.P.I. Cependant pour illustrer notre propos, nous ne présentons que le cycle élémentaire composé des deux phases suivantes, complètement séquentielles (fig. 3.3) :

~ Figure 3.3 : Cycle d'exécution des programmes d'un A.P.I. ~

1) Phase d'analyse des entrées/sorties.
   Pendant cette phase, les données associées aux sorties externes sont transférées de la table de données aux bornes de raccordement des câbles de sortie. Les conducteurs des entrées sont examinés et les bits d'état correspondants sont mis à jour.

---

[2]Durée du cycle = temps d'accès total aux entrées/sorties + temps d'exécution du programme.
Le temps de cycle ou période d'un A.P.I. est le temps d'exécution de 1K-instructions logiques.

2) Phase de traitement du programme.
   Au cours de cette partie du cycle, les états des mécanismes d'entrée, mis à jour, sont appliqués au programme. Le processeur exécute l'ensemble des instructions dans l'ordre dans lequel elles apparaissent. Les bits d'état sont mis à jour selon les règles de la continuité logique, au fur et à mesure de la succession des instructions.

Insistons sur le fait que ces deux phases sont séquentielles : pendant la phase d'exécution du programme, tout changement d'état survenant extérieurement au niveau des capteurs n'est pris en considération que lors de la phase suivante d'analyse des entrées/sorties. De même, tout changement d'état des actionneurs provoqué au cours de l'exécution du programme est mémorisé et n'est appliqué que lors de la phase suivante d'analyse des entrées/sorties.

L'automate programmable SLC 150 d'Allen-Bradley

Notion de gamme

Gamme quantitative

Le critère retenu pour classer les automates dans la gamme quantitative est le nombre maximum d'entrées et de sorties supporté par ces derniers [MLE79]. On distingue les appareils de :

- **bas de gamme**, $20 \leq E/S < 100$;
- **milieu de gamme**, $100 \leq E/S < 500$;
- **haut de gamme**, $500 \leq E/S$.

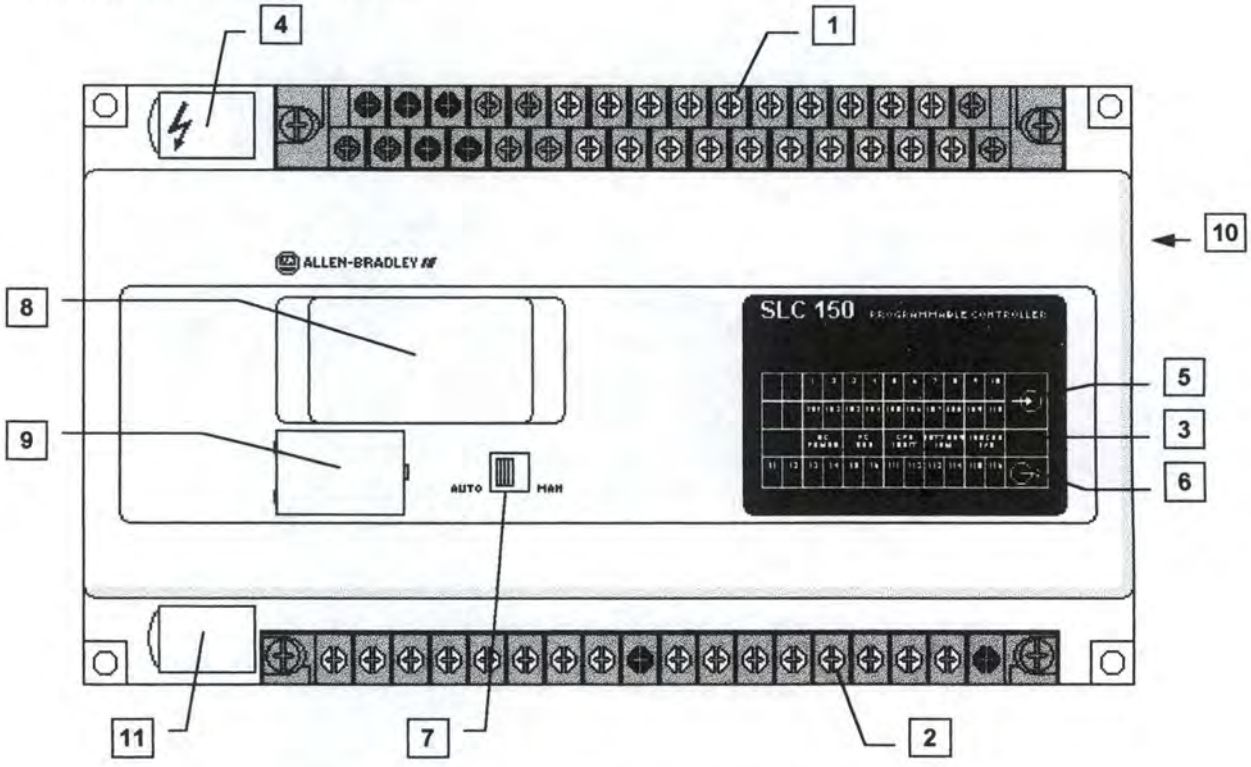
Gamme qualitative

Le classement dans la gamme qualitative s'effectue en fonction du traitement numérique [MLE79]. On distingue les appareils de :

- **classe 1** s'ils n'offrent pas de traitements numériques;
- **classe 2** s'ils offrent quelques traitements numériques simples;
- **classe 3** s'ils possèdent des facultés de traitements évolués.

Par rapport à cette classification, sachant que le SLC 150 possède 32 entrées/sorties "tout ou rien" et n'offre pas la possibilité d'effectuer des traitements numériques, celui-ci s'avère être un automate de bas de gamme, classe 1.

Architecture du SLC 150



~ Figure 3.4 : Automate programmable SLC 150 d'Allen-Bradley ~

1. Bornier des conducteurs d'alimentation et des 20 entrées numériques.
   Les bornes à vis facilitent l'insertion des conducteurs et garantissent la sécurité des connexions. Ce bloc d'entrée peut être aisément retiré, sans décâblage, pour faciliter un éventuel remplacement de l'unité centrale.

2. Bornier des 12 sorties numériques (idem).

3. Cinq indicateurs de diagnostic (diodes électroluminescentes LED) :

   DC POWER indique que l'unité centrale est sous tension;

   PC RUN indique que l'unité centrale est en mode *Run*;

   CPU FAULT indique que le processeur a détecté une erreur au niveau de l'unité centrale, des unités d'expansion ou de la mémoire. Les sorties sont désactivées et le programme automatiquement stoppé;

   BATTERY LOW informe que la tension aux bornes de la pile a chuté en dessous d'un seuil limite. Cette pile - facultative - fournit l'énergie nécessaire à la mémoire RAM en cas de coupure d'alimentation;

   FORCED I/O indique qu'une ou plusieurs adresses d'entrée ou de sortie ont été forcées (dans l'état *On* ou *Off*) afin de tester le programme.

4. Compartiment du fusible. Si la tension d'alimentation est présente mais que l'indicateur DC POWER n'est pas allumé, le fusible a probablement sauté.

5. Vingt indicateurs d'état des signaux d'entrée, identifiés par les numéros d'adresse de 1 à 10 et de 101 à 110, correspondant aux numéros inscrits sous les connexions des entrées. Lorsqu'un mécanisme d'entrée est activé, l'indicateur d'état correspondant s'illumine.

6. Douze indicateurs d'état des signaux de sortie, identifiés par les numéros d'adresse de 11 à 16 et de 111 à 116, correspondant aux numéros inscrits au-dessus des connexions des sorties. Lorsque l'instruction conditionnelle responsable de l'activation d'une sortie est vérifiée, l'indicateur d'état correspondant à cette sortie s'illumine et le circuit concerné est activé.

7. L'interrupteur AUTO/MANUAL commandant le redémarrage de l'unité centrale après une chute de tension ou la correction d'une erreur de CPU. Lors du redémarrage, en position AUTO, l'automate exécute ses tests de diagnostic et entre automatiquement en mode *Run* (s'il s'y trouvait avant la dernière interruption); en position MANUAL, l'automate exécute également ses tests de diagnostic mais n'entre pas en mode *Run*. Pour ce faire, il faut placer l'interrupteur en mode AUTO ou utiliser soit la console de programmation (*Pocket Programmer*) soit un ordinateur connecté à l'automate.

8. Compartiment du module de mémoire EEPROM. Ce module optionnel peut être enfiché dans le processeur. Le programmeur de poche ou le logiciel de l'ordinateur

relié à l'automate par l'interface, permet de sauvegarder le programme de la mémoire RAM (de l'unité centrale) dans l'EEPROM, ainsi que de charger un programme de l'EEPROM dans la mémoire RAM, bien évidemment.

9. Port de communication. Le programmeur de poche, l'interface (convertisseur RS-232/RS-422) ou encore le TCAT (*Timer Counter Access Terminal*) s'y connecte.

10. Prise pour la connexion d'une unité d'expansion ou d'un module HSI (*High Speed Input*).

11. Compartiment de la pile. Cette dernière - non indispensable - fournit la puissance nécessaire au maintien du contenu de la mémoire RAM durant 2 à 3 ans.

Composants auxiliaires du SLC 150

~ Figure 3.5 : Composants utilisables avec l'automate SLC 150 ~

L'automate SLC 150 peut être utilisé conjointement avec les composants auxiliaires suivants (fig. 3.5) :

- Appareils de programmation et de contrôle comprenant la console de programmation, le TCAT (*Timer Counter Access Terminal*) et l'interface pour la liaison PC-automate. Ces appareils se branchent dans le port de communication de l'automate.

- Unités d'expansion d'entrées/sorties, connectées à l'unité centrale par câble plat. Diverses dispositions peuvent être mises en oeuvre afin d'obtenir la configuration d'entrées/sorties désirée.

- Module HSI (*High Speed Input*) s'impose, dans certaines applications, lorsque la vitesse d'acquisition est très élevée au niveau de certaines entrées (jusqu'à une fréquence de 5 Khz).

- Le module de mémoire EEPROM qui se place directement au coeur de l'unité centrale.

Programmation du SLC 150

Celle-ci peut être réalisée à l'aide d'un des deux appareils suivants, à la fois similaires et distincts, communiquant avec l'automate via le port de communication :

- La console de programmation (*Pocket Programmer*) pour programmer, éditer et contrôler les opérations de l'automate;

- Le kit de l'interface PC-SLC comprenant le logiciel, les modes d'emploi et une interface (convertisseur RS-232/RS-422) requise pour la communication entre le PC et l'automate. Grâce au logiciel, un traditionnel PC (IBM ou compatible) peut être utilisé à la place de la console. Ce logiciel amplifie la convivialité et la souplesse et offre, outre les fonctions offertes par la console, la compétence additionnelle de programmer en mode *Off-line* (sans liaison avec l'automate) ainsi que la disponibilité d'une bibliothèque de programmes, l'affichage des schémas à contacts, le test et la mise au point de programmes de façon aisée.

Précisons qu'à chaque instant, l'A.P.I. est soit en mode exécution, soit en mode production de programme mais il ne peut exécuter simultanément ces deux types de travaux.

Le langage de programmation utilisé par l'automate SLC 150 est le langage à relais LD (*Ladder Diagram*) faisant l'objet de la suite de ce chapitre.

LD, le langage à relais

Sans commune mesure avec les langages que l'informatique a l'habitude d'utiliser, le langage des automates programmables est simple de nature, simplicité née de la logique.

Le LD (*Ladder Diagram* : diagramme en échelle), langage graphique dit "à relais" ou encore "à contacts", fut le premier langage des automates : en effet, il s'imposait naturellement car, aux Etats-Unis, les problèmes logiques étaient résolus par des armoires à relais d'où l'habitude d'utiliser les schémas à relais. Au contraire, l'Europe a développé la logique statique avant d'arriver aux logiques programmables d'où la pratique de l'écriture des équations booléennes [MLE79].

Il va de soi que le principal intérêt du langage à relais est d'être familier à l'opérateur découvrant les logiques programmables, d'être bien adapté au dépannage et à certaines visualisations du fonctionnement tandis que les langages de calcul, plus puissants, conviennent bien aux études et aux développements.

Le schéma à contacts est la représentation graphique des problèmes d'automatismes avec les symboles des circuits électriques. Dans un circuit électrique, la continuité électrique est nécessaire pour alimenter la sortie tandis que dans un diagramme à contacts, la continuité logique est nécessaire pour que la sortie soit actionnée.

Les cinq constituants suivants sont présents dans tous les jeux d'instructions des automates utilisant les schémas à relais :

- le relais normalement ouvert ⊣|⊢ ;
- le relais normalement fermé ⊣|/|⊢ ;
- l'ouverture de branche parallèle ⊤ ;
- la fermeture de branche parallèle ⊥ ;
- le symbole d'affectation du résultat à une variable intermédiaire ou à une sortie —( )—.

Une fonction logique (ET/OU) est obtenue par un assemblage convenable de ces constituants (fig. 3.6). Une ligne est une concaténation de ces constituants et comprend au moins un symbole d'affectation.

OPERATEUR BOOLEEN	ET	OU	COMPL.
SYMBOLIQUE MATHEMATIQUE	$\wedge$	$\vee$	$\neg$
CONVENTION AMERICAINE	⊣|⊢⊣|⊢	⊣⊢ ⊣⊢	⊣|/|⊢

~ Figure 3.6 : Représentation des opérateurs booléens ~

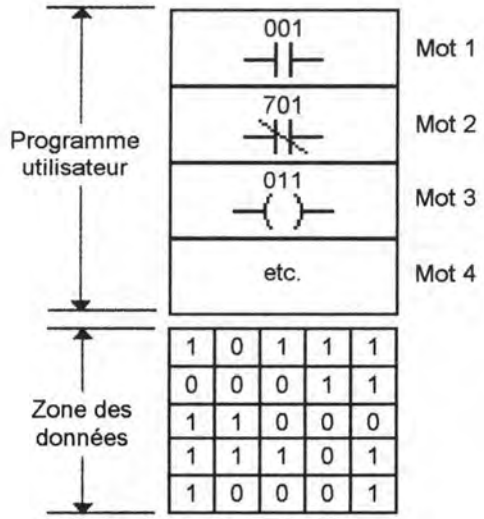
Dans un programme écrit en LD, les lignes "d'instructions" sont exécutées l'une après l'autre à chaque cycle, dans l'ordre de leur apparition. Une ligne est elle-même exécutée de gauche à droite. Ces diagrammes en échelle ont la particularité d'être lisibles et, en conséquence, leur maintenance est simple à assurer. C'est le cas notamment du SLC qui, grâce à sa programmation facile, permet de le maîtriser en un temps très court.

Principes de la programmation du SLC 150

Structure de la mémoire

La figure 3.7 illustre de manière simplifiée la mémoire centrale de l'automate : celle-ci est spécialisée et non banalisée, c'est-à-dire découpée en zones, une première réservée au programme de l'utilisateur et une seconde, destinée à contenir les données.

La zone des données contient à la fois les valeurs des entrées acquises par la machine, des variables intermédiaires et des sorties à transmettre. Dans le cas du SLC 150, cette zone des données est structurée en bits, celui-ci ne travaillant qu'avec des valeurs booléennes et des fonctions logiques. Remarquons qu'il existe une correspondance rigoureuse entre la position des bits correspondant aux entrées/sorties et les bornes de raccordement des câbles à l'extérieur de la machine.



~ Figure 3.7 : Représentation simplifiée de la mémoire centrale du SLC 150 ~

Cette spécialisation de la mémoire a comme conséquence intéressante de décharger complètement l'utilisateur des procédures d'entrées/sorties industrielles. En effet, **les instructions d'entrées/sorties sont implicites sur les automates** [MLE79], ces dernières sont gérées par la machine : lorsque la machine rencontre une variable d'entrée dans une équation, elle se charge d'acquérir la valeur correspondante; s'il s'agit d'une variable intermédiaire, elle la recherche en mémoire pour la lire ou y enregistrer une valeur selon l'instruction en cours; pour une variable correspondant à une sortie, il en est de même : elle observe sa valeur ou y affecte un résultat d'équation.

Adressage des instructions

Le long d'une ligne de programme (échelon ou *rung* en anglais), chaque symbole représente une instruction surmontée généralement du numéro de l'adresse à laquelle s'applique cette instruction. Ce numéro identifie la fonction de l'instruction et la lie à un bit d'état de la "zone de données" de la mémoire de l'automate. Ce bit d'état peut être soit

à ON (état logique 1), soit à OFF (état logique 0) produisant ainsi l'état VRAI/FAUX des instructions auxquelles il est associé.

Par exemple, si le bit correspondant à l'adresse 001 = OFF alors
l'instruction *Examine On* est fausse et l'instruction *Examine Off* est vraie.

```
   001                          001
--] [--                      --]\[--
```

Toutes les instructions sont stockées dans la zone "Programme utilisateur" de la mémoire centrale de l'automate, dans le même ordre que celui dans lequel elles furent encodées. Pendant l'exécution du programme, le processeur de l'automate examine les bits d'état et les met à jour selon les instructions successives et les règles de la continuité logique.

Les adresses d'E/S et les adresses internes

Les adresses d'E/S sont utilisées dans le programme pour désigner les mécanismes d'E/S connectés à l'automate. **Chaque adresse d'E/S correspond rigoureusement à une borne de raccordement de l'automate.** D'autres adresses, dites "internes", concernent des variables intermédiaires qui permettent d'établir les conditions exactes sous lesquelles l'automate va activer ou désactiver les sorties en réponse à l'état des entrées.

La configuration maximale offerte par le SLC 150 est de 112 adresses d'E/S (70 entrées et 42 sorties) réparties en 7 blocs : un bloc comprend 10 entrées et 6 sorties. Le SLC 150 possède déjà 20 entrées et 12 sorties c'est-à-dire que l'on dispose d'office des blocs 0 et 1 [ABC87]. Pour accroître le nombre d'E/S, il faut connecter à l'automate des unités d'expansion.

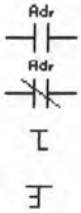
TYPE DE DONNEE	ADRESSES SLC 150 avec $2 \leq x \leq 6$ (bloc supplémentaire)
ENTREES	**{001, ..., 010}** (Bloc 0) $\cup$ **{101, ..., 110}** (Bloc 1) $\cup$ {x01, ..., x10} (Bloc x)
SORTIES	**{011, ..., 016}** (Bloc 0) $\cup$ **{111, ..., 116}** (Bloc 1) $\cup$ {x11, ..., x16} (Bloc x)
VARIABLES INTERMEDIAIRES	**{701, ..., 863}** $\cup$ **{017, 018}** (Bloc 0) $\cup$ **{117, 118}** (Bloc 1) $\cup$ {x17, x18} (Bloc x)
TEMPORISATIONS	**{901, ..., 932}** $\cup$ **{951, ..., 982}**

Les instructions du SLC 150

Parmi le jeu d'instructions du SLC 150, ne sont reprises ci-dessous que celles apparaissant au sein du programme développé dans la suite de cet ouvrage. Pour de plus amples informations concernant les autres instructions disponibles, se référer au "manuel utilisateur" de l'automate [ABC87].

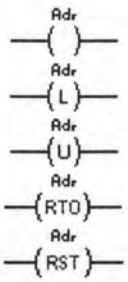
Les instructions dites "conditionnelles" :

- l'instruction ***Examine On***
- l'instruction ***Examine Off***
- l'ouverture de branche parallèle ***Branch Open***
- la fermeture de branche parallèle ***Branch Close***



Les instructions dites "de sortie" :

- l'instruction ***Output Energize***
- l'instruction ***Latch*** (verrouiller)
- l'instruction ***Unlatch*** (déverrouiller)
- l'instruction de temporisation *Retentive Timer On-Delay* ***RTO***
- l'instruction de réinitialisation ***Reset***



Excepté les instructions d'ouverture et de fermeture de branche parallèle, toutes les instructions précitées sont accompagnées d'une adresse. En outre, les instructions de temporisation et le *Reset* nécessitent un paramètre additionnel détaillé dans la suite de ce chapitre.

Les instructions dites "de type relais"

Par ces termes, nous désignons l'ensemble des instructions citées ci-dessus hormis celles de temporisation et de réinitialisation.

Bien que nous l'ayons déjà expliqué ci-dessus, rappelons que chaque instruction est liée, par son adresse, à un bit d'état de la "zone de données" de la mémoire de l'automate. Ce bit, positionné à ON ou OFF, détermine l'état de l'instruction.

En effet, l'instruction ***Examine On*** demande au processeur d'examiner si le bit - correspondant à l'adresse jointe à l'instruction - est à l'état ON. Si c'est le cas, l'instruction est vraie, sinon elle est fausse. Au contraire, l'instruction ***Examine Off*** est

vraie si le bit relatif à l'adresse associée est en position OFF et est fausse si ce dernier est actif (ON).

Lorsque les instructions conditionnelles qui la précèdent sont vraies, l'instruction ***Output Energize*** ordonne à l'automate d'activer (mise à ON) le bit d'état adressé par cette instruction.

Dans l'exemple ci-joint,

lorsqu'à la fois, l'instruction *Examine On 001* et l'instruction *Examine Off 002* sont vraies (c'est-à-dire 001 ON et 002 OFF), le bit d'état associé à l'adresse de sortie 011 est mis en position ON et la sortie correspondante est donc alimentée.

Remarque importante : En termes de continuité, lorsqu'apparaît, dans une ligne de programme, un enchaînement continu d'instructions conditionnelles vraies, la continuité logique existe; en conséquence, l'instruction de sortie est vraie et son bit d'état est activé en position ON. Si une seule des instructions conditionnelles de la suite devient fausse, la continuité logique est perdue; l'instruction de sortie est alors fausse et son bit d'état changé en position OFF [ABC87].

Les instructions de branchement ***Branch Open*** et ***Branch Close*** sont utilisées pour créer des chemins parallèles d'instructions conditionnelles; en d'autres mots, ils contribuent à la création du "OU" logique. L'exemple suivant illustre une telle situation : pour que la sortie 011 soit alimentée, il faut que l'entrée d'adresse 001 soit à l'état ON **ou** que l'entrée d'adresse 002 soit à l'état OFF.

Lorsque les instructions conditionnelles précédant une instruction ***Latch*** sont vérifiées, le bit d'état associé à l'adresse de cette instruction est activé en position ON et reste verrouillé dans cet état même si la continuité logique est perdue par la suite. Seule une instruction ***Unlatch*** appliquée à la même adresse permet de déverrouiller le bit d'état.

Illustrons ce principe de rémanence par un exemple. Soit le diagramme en échelle suivant :

Imaginons les adresses d'entrées 001 et 002 symbolisant respectivement deux boutons-poussoirs A et B. L'adresse 011 est celle d'une sortie de l'automate connectée à une lampe. Un appui sur le bouton-poussoir A allume la lampe. L'instruction de sortie exécutée étant une instruction *Latch*, le bit associé à l'adresse 011 est maintenu à ON, en d'autres termes, la lampe reste allumée et cela même si le bouton A est relâché. Un appui sur la touche B s'avère nécessaire pour déverrouiller le bit associé à l'adresse 011 et éteindre la lampe en conséquence.

N.B. En cas de rupture d'alimentation ou du passage du mode *Run* en un autre mode, le bit d'état lié à une instruction *Latch* ou *Unlatch* est mémorisé (ceci à condition évidemment que la mémoire soit alimentée par la pile prévue à cet effet). Donc, lors du retour de l'alimentation ou du mode *Run*, tout mécanisme de sortie associé à un bit d'état précédemment actif (à ON) redémarre aussitôt. Par contre, dans le cas d'une erreur au niveau du processeur (*CPU Fault*), les bits d'état liés aux instructions *Latch* et *Unlatch* sont automatiquement réinitialisés [ABC87].

Les instructions de temporisation et de réinitialisation

Ces deux instructions sont, rappelons-le :

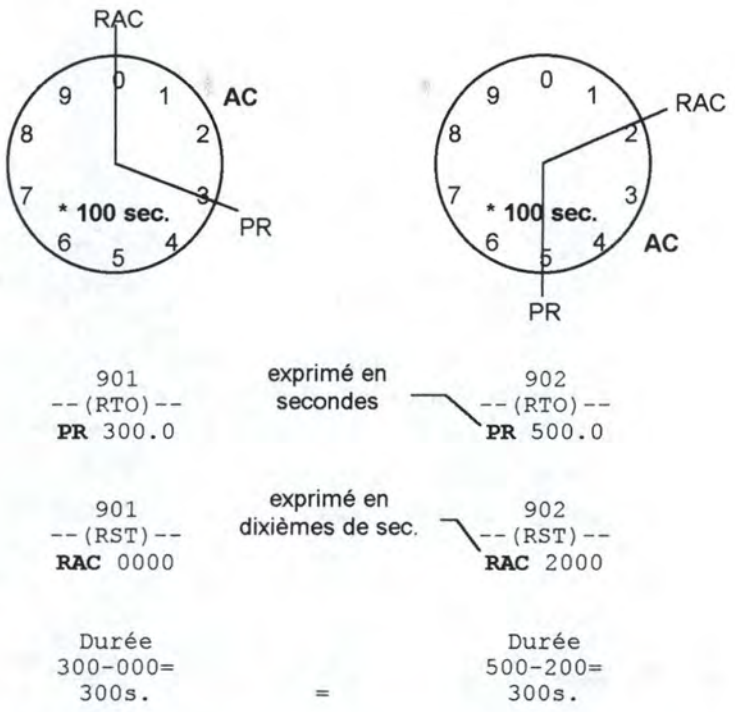
- la temporisation cumulative à l'enclenchement ***RTO*** (*Retentive Timer On-Delay*) et
- la réinitialisation ***Reset*** requise par la précédente.

Une temporisation ***RTO*** fonctionne comme une horloge interne, comptant des intervalles de 0,1 seconde durant tout le temps qu'elle est vérifiée. Le nombre d'intervalles comptés est appelé la valeur cumulée AC (*ACcumulated value*). Le principe est donc le suivant : tant que l'instruction est vraie, la valeur AC s'incrémente; dès qu'elle devient fausse, le comptage est interrompu et la valeur AC mémorisée [ABC87]. Le comptage reprend à cette valeur ainsi retenue aussitôt que l'instruction est de nouveau activée; c'est la raison pour laquelle cette temporisation est dite "cumulative".

Il va de soi que l'intérêt d'une temporisation est de délivrer un signal lorsque celle-ci atteint une durée prescrite. La constante de durée de la temporisation est fixée en programmant une valeur préréglée PR (*PReset value*), paramètre supplémentaire de la temporisation. Lorsque la valeur AC égale cette constante PR, le bit d'état associé à l'adresse de la temporisation est mis en position ON.

Une constante de réinitialisation RAC (*Reset ACcumulated value*) accompagnant l'instruction ***Reset*** est affectée à la variable AC dès que l'instruction *Reset* est exécutée. La constante RAC est égale à 0 dans la plupart des cas, PR représente alors la durée de la temporisation. Dans les autres cas (RAC≠0), la durée de la temporisation est alors égale à PR-RAC.

Sur chaque horloge de la figure 3.8, RAC et PR délimitent l'ensemble des valeurs que doit parcourir AC pour accomplir la temporisation. Notons que la durée de temporisation est la même dans ces deux cas distincts.



~ Figure 3.8 : Représentation graphique de deux temporisations *RTO* équivalentes ~

Bits d'état et de débordement

Le bit d'état - informant que la temporisation est arrivée à son terme - a la même adresse que l'instruction de temporisation. Le bit d'état d'une instruction *RTO* est mis à ON lorsque la valeur AC atteint la constante PR présélectionnée. De là, cette dénomination de temporisation cumulative "à l'enclenchement" pour *RTO*.

```
    901
--(RTO)--
PR xxx.x

Lorsque AC<PR,
    901
  --] [--
est fausse.

Lorsque AC=PR,
    901
  --] [--
est vraie.
```

Un bit de débordement indique si la valeur AC de la temporisation passe de 999,9 à 0. L'adresse de ce bit particulier est celle de la temporisation + 50. Le bit d'état n'est pas affecté par cette modification.

```
    901
--(RTO)--
PR xxx.x

Lorsque AC>999,9
    951
  --] [--
est vraie.
```

L'instruction ***Reset***

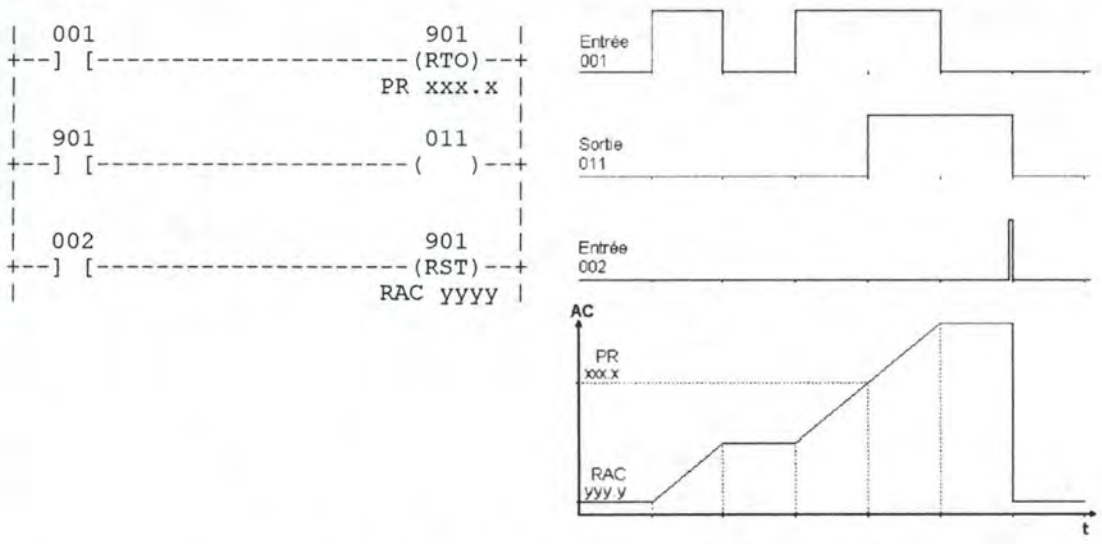
```
901
--(RST)--
RAC yyyy
```

Cette commande réinitialise la temporisation dont l'adresse est précisée. Cette action a pour effet de désactiver (mise à OFF) les bits d'état et de débordement de la temporisation et d'assigner la constante RAC à la valeur AC. Ce *Reset* doit être "relâché" pour que la temporisation puisse par la suite être exécutée de nouveau.

N.B. En cas de rupture d'alimentation ou du passage du mode *Run* en un autre mode, les valeurs AC ainsi que les bits d'état et de débordement de chaque temporisation sont mémorisés et restaurés lors du rétablissement de l'alimentation ou du retour en mode *Run*. Au contraire, dans le cas d'une erreur au niveau du processeur (*CPU Fault*), les valeurs AC, les bits d'état et de débordement sont automatiquement réinitialisés.

Illustration d'une temporisation cumulative à l'enclenchement ***RTO***

Le chronogramme suivant (fig. 3.9) illustre la nature rémanente de la temporisation. En effet, la temporisation comptabilise les périodes pendant lesquelles la continuité logique est présente sur sa ligne. Pendant ces périodes, à moins qu'une instruction *Reset* soit appliquée à cette temporisation, la valeur AC s'incrémente et, lorsque la ligne contenant la commande *RTO* devient fausse, la valeur AC cesse de croître mais est néanmoins mémorisée à sa valeur en cours. L'instruction *Reset* réinitialise AC à la valeur RAC (0 dans la plupart des cas).



~ Figure 3.9 : Principe de fonctionnement d'une temporisation *RTO* ~

Prise en compte d'une transition : le créneau unitaire

La construction présentée ci-dessous apparaît à maintes reprises au cœur du programme situé en annexe. En outre, elle constitue une fonction de base de la méthodologie de réalisation des automatismes [MLE79].

Pour illustrer son utilité, prenons l'exemple simple d'une lampe commandée par un seul bouton-poussoir en guise d'interrupteur. Un premier appui sur ce dernier doit allumer la lampe, celle-ci restant alimentée même si l'utilisateur relâche la touche; un deuxième appui est donc nécessaire pour éteindre la lampe. Si nous programmons la ligne suivante, l'effet obtenu n'est pas celui précédemment spécifié.

En effet, cette ligne ne provoque pas l'effet escompté : la sortie 011 n'est activée que si l'entrée 001 est en position ON; en d'autres mots, la lampe n'est allumée que lorsque l'utilisateur maintient la pression sur la touche; sitôt qu'elle est relâchée, la continuité logique est rompue et la sortie n'est plus activée.

Autre façon d'aborder cette application :

Dans ce cas-ci, dès que l'automate détecte que l'utilisateur actionne le bouton-poussoir (phase d'analyse des E/S), le bit d'état 001 en mis position ON; ensuite, l'exécution successive de chaque ligne du programme provoque l'allumage de la lampe (*Latch*) immédiatement suivi de son extinction (*Unlatch*) : celle-ci "clignote" à une fréquence dépendante du temps de cycle de l'automate et ce, tant que l'utilisateur maintient la touche enfoncée.

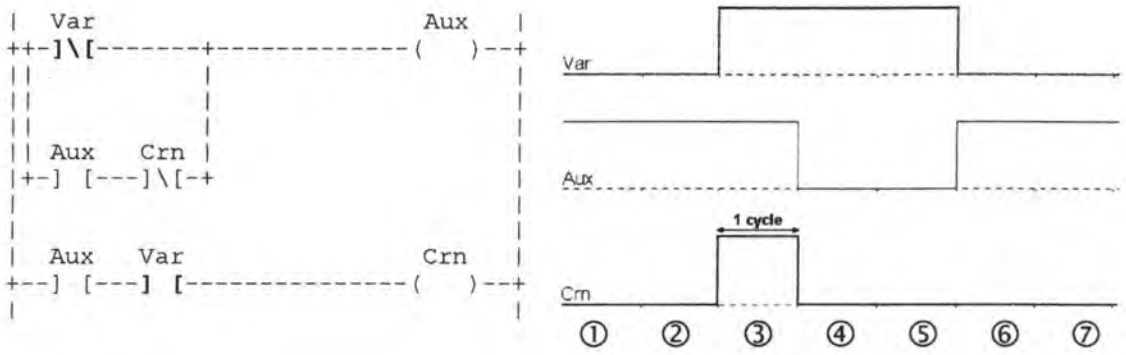
Cette application souligne la distinction entre le fonctionnement **par niveau** et le fonctionnement **par transition** [MLE79]. Le type d'automate utilisé ici fonctionne par niveau; rappelons le fonctionnement de la temporisation : si le signal passe à zéro, la temporisation s'arrête en mémorisant le temps écoulé et dès que le signal passe de 0 à 1, la temporisation s'incrémente à nouveau d'une unité à chaque cycle.

Or, de nombreuses applications reposent sur les transitions de signaux et non sur leur niveau. En fait, la différence de vitesse entre le processus industriel et l'automate impose que le front montant (0→1) ou descendant (1→0) de la transition soit détecté et traité durant un seul cycle alors que le phénomène physique peut se prolonger plus longtemps [MLE79].

D'où la nécessité de construire un signal logique qui, déclenché par un front montant ou descendant, prend la valeur 1 pendant un cycle et un seul : le créneau unitaire [MLE79].

Créneau unitaire associé à un front montant

Si l'on souhaite se préoccuper uniquement de la transition 0→1 d'un signal, il faut créer une variable intermédiaire qui prend la forme d'un créneau de durée "1 cycle". Celle-ci peut être réalisée de la manière présentée à la figure 3.10.



~ Figure 3.10 : Créneau unitaire associé à un front montant ~

Crn est le créneau unitaire associé au front montant (transition 0→1) d'une entrée *Var*. Pour le construire, nous utilisons une variable auxiliaire *Aux*. Le tableau suivant reprend les valeurs prises par les différentes variables au cours des cycles successifs.

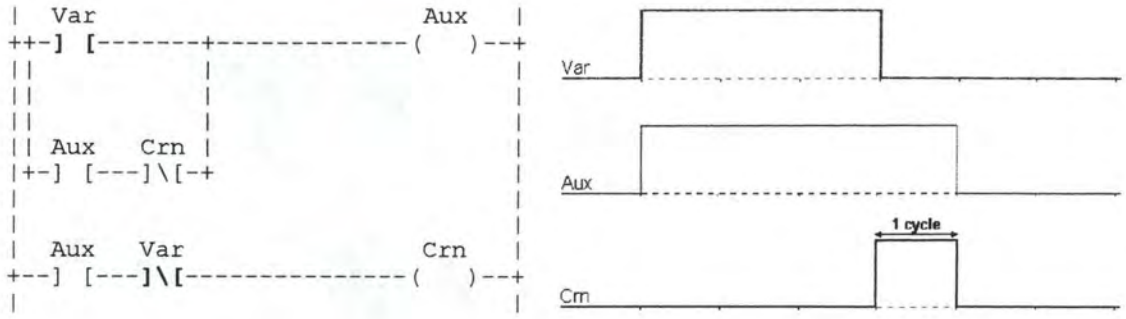
Cycle =	Analyse E/S	+ Exécution du programme	
1	Var = 0	Aux = ¬Var = 1	Crn = 0
2	"	"	"
3	Var = **1**	Aux = $Aux_{-1} \wedge \neg Crn_{-1}$ = **1**	Crn = **1**
4	Var = 1	Aux = 0	Crn = 0
5	"	"	"
6	Var = 0	Aux = ¬Var = 1	Crn = 0
7	"	"	"

Notre application peut à présent être réalisée : il suffit d'y ajouter les lignes suivantes après avoir construit le créneau en faisant correspondre le bouton-poussoir à l'entrée *Var*. Lorsqu'une transition 0→1 du signal de la touche est perçue, si la lampe est éteinte, la première ligne active la variable 011 (ce qui va entraîner l'allumage de la lampe à la phase suivante d'analyse des entrées/sorties) tandis que si la lampe est allumée, la deuxième ligne désactive cette variable de sortie.

```
|  Crn  Lampe                Lampe  |
|  702   011                  011   |
+--] [---]\[---------------( L )--+
|                                   |
|  Crn  Lampe                Lampe  |
|  702   011                  011   |
+--] [---] [---------------( U )--+
|                                   |
```

Créneau unitaire associé à un front descendant

Le schéma à relais et le chronogramme de la figure 3.11 illustrent le principe du créneau unitaire correspondant à un front descendant (transition 1→0). La seule différence avec le créneau associé au front montant réside dans le complément des instructions relatives à l'entrée *Var* dont on veux prélever le front descendant.



~ Figure 3.11 : Créneau unitaire associé à un front descendant ~

Chapitre

4

De la spécification à la solution

Architecture du système de commande

Le système destiné à commander la grue est constitué d'un automate programmable SLC 150 auxquels sont connectés les dispositifs suivants :

- le pupitre reprenant l'ensemble des commandes à la disposition de l'opérateur (fig.4.1);
- l'ensemble des capteurs renseignant sur la position des divers éléments de la grue, sur la charge et le moment supportés par celle-ci et sur la vitesse du vent (anémomètre);
- l'ensemble des actionneurs : les différents moteurs actionnant la grue sans oublier l'avertisseur sonore.

En ce qui concerne le tableau de commande, nous l'avons imaginé de manière à ce que l'utilisateur puisse agir facilement, en regroupant les actions concernant un même mouvement.

La grande majorité des capteurs utilisés est constituée de capteurs "tout ou rien", essentiellement des capteurs de fin de course. Cependant, cette application nécessite l'utilisation de capteurs analogiques notamment pour la mesure de la charge soutenue par le crochet, pour la mesure du moment ou encore pour la mesure de la vitesse du vent pouvant mettre en péril la stabilité de la grue.

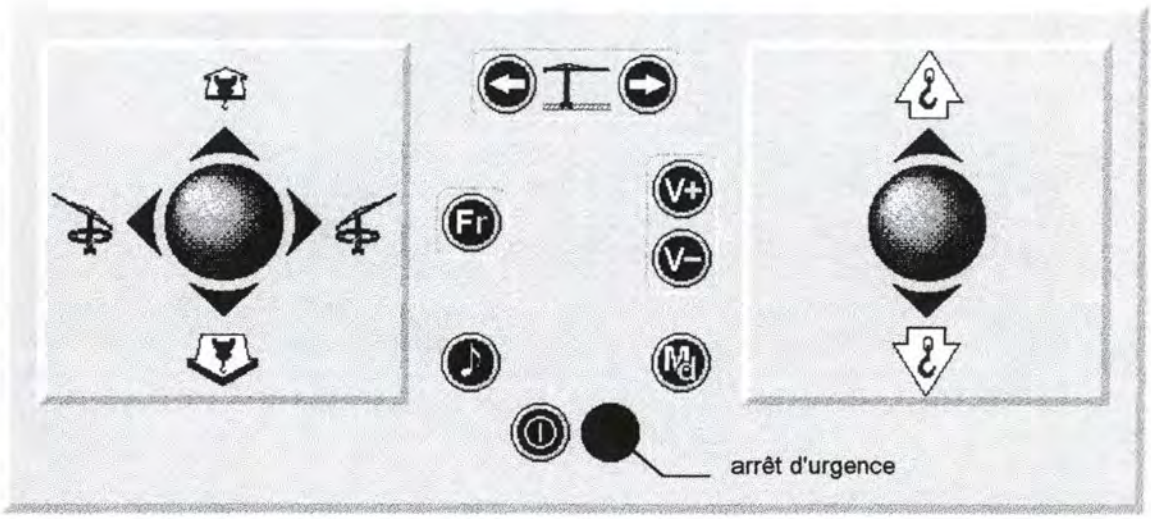
Puisque l'automate choisi est un automate de classe 1 n'offrant pas de traitements numériques (cf. Chapitre 3, Automate programmable SLC 150 d'Allen-Bradley) et parce que la mise en oeuvre matérielle du système sort du cadre de ce mémoire, nous avons pris l'hypothèse que chacun de ces capteurs soit "à détection de seuil", c'est-à-dire que le signal analogique correspondant à la mesure est comparé à un seuil et l'information renvoyée par le capteur est 1 si le signal égale ou dépasse ce seuil fixé ou 0 dans le cas contraire.

Examinons à présent l'ensemble des éléments connectés aux entrées et sorties de l'automate en précisant leur adresse et leur libellé les identifiant au cœur du programme. Il va de soi que ces éléments découlent directement de la spécification : les boutons-poussoirs du pupitre correspondent aux actions de l'agent "Opérateur" de la spécification, les composants d'état de l'agent "Environnement" représentent les capteurs de fin de course; enfin, les sorties de l'automate actionnant les différents moteurs, le frein de giration, l'avertisseur sonore ou encore la mise sous-tension de la grue apparaissent en tant que composants d'état dans l'agent "Système".

La suite de ce chapitre aborde plus en détail les liens existant entre la spécification formelle et le programme en LD (respectivement en annexes 2 et 3). Avant cela, le concept de traçabilité est brièvement présenté.

Dispositifs connectés aux entrées

Le pupitre de commande



~ Figure 4.1 : Pupitre de commande de la grue ~

Mise en/hors service de la grue		TOUCH ONOFF	001
Montée du crochet		TOUCH CR /\	002
Descente du crochet		TOUCH CR \/	003
Avance du chariot		TOUCH CH ->	004
Recul du chariot		TOUCH <- CH	005
Translation de la grue vers l'avant		TOUCH GR ->	006
Translation de la grue vers l'arrière		TOUCH <- GR	007
Giration vers la droite		TOUCH FL >>	008
Giration vers la gauche		TOUCH << FL	009
En/Dé-clenchement du frein de giration	Fr	TOUCH FREIN	010
Passage en mode dégradé	Md	TOUCH DEGRA	101
Augmentation de la vitesse de levage	V+	TOUCH VIT +	108
Diminution de la vitesse de levage	V-	TOUCH VIT -	109
Avertisseur sonore		TOUCH KLAX	110

En règle générale, à chaque action de l'agent "Opérateur" correspond une entrée de l'automate programmable. Seules les actions **CderStop...** n'apparaissent pas explicitement comme entrées de l'automate : en fait, un moteur n'est plus actionné dès

que l'entrée de commande du mouvement n'est plus captée. Faisant référence à l'arrêt d'urgence, l'action **Stop** ne figure pas non plus dans le programme pour des raisons de priorité. Le rôle de l'arrêt d'urgence est de couper l'alimentation de toutes les phases de l'installation électrique et ce, dans toutes les situations même les plus inattendues pouvant entraîner un dysfonctionnement de l'automate (ex. problème au niveau du processeur ou de la mémoire). Pour assurer cette sécurité dans tous les cas, il ne fait pas l'objet d'une entrée de l'automate : il est prioritaire par rapport au système de commande.

Notons également que dans notre implémentation, les couples d'actions **On/Off** et **Enclencher/RelâcherFreinGiration** sont commandées chacun par une seule touche du pupitre : des créneaux unitaires ont été construits pour détecter l'appui sur ces touches et agir correctement en fonction de l'état du système.

Actions de l'agent Opérateur	Implémentation sur automate
On	front montant de l'appui sur ⏻ et grue hors service
Off	front montant de l'appui sur ⏻ et grue en service
EnclencherFreinGiration	front montant de l'appui sur Fr et frein non enclenché
RelâcherFreinGiration	front montant de l'appui sur Fr et frein enclenché

6 limiteurs de course

Ces capteurs de fin de course correspondent aux composants d'état de l'agent "Environnement" de la spécification.

Limite de montée de la charge	CROCH HAUT	102
Limite de descente de la charge	CROCH BAS	103
Chariot en bout de flèche	CHAR EXTRM	104
Chariot au pied de la grue	CHAR SECU	105
Limite de translation avant de la grue	GRUE LIMAV	106
Limite de translation arrière de la grue	GRUE LIMAR	107

Limiteurs de charge, de couple de charge et anémomètre

Les 20 entrées du SLC 150 ayant été utilisées par l'ensemble des commandes et des limiteurs de course énumérés ci-dessus, les capteurs suivants nécessitent la connexion d'une unité d'expansion[1].

Limiteur de charge signalant que la charge > max	CHARG > MAX	201
Limiteur de charge signalant que la charge < 20%	CHARG < 20%	202
Limiteur de couple de charge, moment > max	MOMEN > MAX	203
Anémomètre, signal de vitesse du vent > max	ANEMO V>MAX	204

[1] Une unité d'expansion de SLC 100 suffit : 10 entrées (201-210) et 6 sorties (211-216) supplémentaires.

En fait, l'apparition de ces quatre capteurs à détection de seuil est liée à l'utilisation d'un automate de bas de gamme ne pouvant traiter de valeurs numériques. Pour un automate de gamme supérieure offrant la possibilité d'effectuer des traitements analogiques, ces capteurs ne figureraient pas comme entrées de l'automate.

Dispositifs connectés aux sorties

Les 12 sorties du SLC 150

Grue en service	GR EN SERVI	011
Activation du moteur de levage, première vitesse	-MOT- VIT 1	012
Activation du moteur de levage, deuxième vitesse	-MOT- VIT 2	013
Activation du moteur de levage, troisième vitesse	-MOT- VIT 3	014
Sens du moteur de levage, 1 = vers le haut	SENS LEVAG	015
Activation du moteur de distribution	-MOT- DISTR	016
Sens du moteur de distribution, 1 = vers l'avant	SENS DISTR	111
Activation du moteur de translation	-MOT- TRANS	112
Sens du moteur de translation, 1 = vers l'avant	SENS TRANS	113
Activation du moteur de giration	-MOT- GIRAT	114
Sens du moteur de giration, 1 = vers la droite	SENS GIRAT	115
Activation du frein de giration	FREIN GIRAT	116

Sortie supplémentaire (unité d'expansion)

Avertisseur sonore (Klaxon)	AVERT SONOR	211

Chacune de ces sorties apparaît dans la spécification en tant que composant d'état de l'agent "Système". Pour chaque moteur, alors qu'un seul composant d'état est présent au niveau de la spécification, plusieurs sorties sont nécessaires techniquement : pour les moteurs à vitesse unique, la première sortie actionne le relais responsable de son alimentation électrique, la seconde fournit le sens de rotation; le moteur de levage comportant trois vitesses, quatre sorties différentes lui sont donc associées (trois correspondant à chaque "enroulement" du moteur asynchrone et une pour le sens).

Définition et avantages de la traçabilité

La spécification des besoins est un document qui décrit les objectifs qu'un système doit satisfaire et le comportement observable (interactions avec l'environnement) qu'il doit avoir de manière à réaliser ces objectifs [WIE95].

En fait, la spécification des besoins joue un rôle important de communication entre les personnes impliquées dans le processus de développement et qui expriment quelquefois des besoins contradictoires (clients, développeurs, équipe de maintenance). Ces spécifications doivent être continuellement mises à jour afin de suivre l'évolution des besoins de l'utilisateur [WIE95] car :

- les développeurs et les clients l'utilisent pour se mettre d'accord sur les fonctionnalités que le produit doit avoir;
- les personnes chargées de la maintenance s'en servent comme d'une norme d'après laquelle ils corrigent les erreurs du produit;
- les clients et l'équipe de maintenance l'utilisent pour se mettre d'accord sur les modifications à apporter au produit afin de mieux correspondre à l'évolution des besoins des utilisateurs.

A la fin du développement, la spécification, la conception et l'implémentation doivent concorder et pour toutes les modifications ultérieures, tous les documents doivent rester en accord avec les changements des besoins de l'utilisateur. Pour réaliser cela, toutes les parties de tous les documents devant changer ensemble doivent être liées.

La "traçabilité" est la caractéristique que chaque partie de chaque document peut mener à toute autre partie de tout autre document de sorte que les parties devant être modifiées ensemble soient liées les unes aux autres [WIE95].

Le problème du maintien de la traçabilité est celui d'enregistrer les liens pertinents entre les documents développés au cours des différentes phases du processus. Ces documents liés concernent donc aussi bien la spécification, la conception et l'implémentation du système. Une spécification des besoins d'un logiciel possède cette qualité de "traçabilité" si l'origine de chaque besoin est claire et si elle référence facilement chaque besoin dans le développement futur ou la documentation [WIE95].

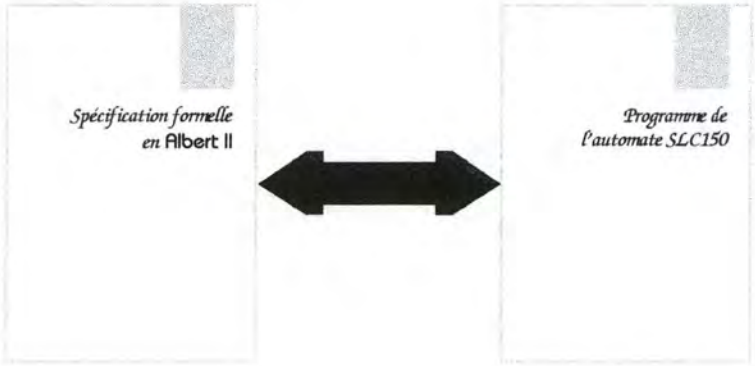
On peut trouver également dans la littérature deux concepts de traçabilité : vers l'avant (*forward traceability*) et vers l'arrière (*backward traceability*) applicables aux différents types de documents [WIE95].

La traçabilité vers l'avant des besoins est la capacité de pouvoir se déplacer des composants d'une spécification vers les composants de l'implémentation tandis que la traçabilité vers l'arrière des besoins est la capacité de retrouver la source d'un besoin (personne ayant demandé que ce besoin soit présent).

Quelques avantages de la traçabilité [WIE95] :

- l'impact d'une modification des besoins peut être estimé;
- les conflits entre les besoins peuvent être découverts plus rapidement (retards évités);
- les besoins n'ayant pas encore été implémentés peuvent être rassemblés et le travail restant à effectuer peut être estimé;
- la qualité du produit par rapport aux besoins de l'utilisateur peut être évaluée;
- la faculté d'estimer la possibilité de changer l'implémentation en restant en accord avec les besoins inchangés.

Puisqu'il existe divers types de liens entre différents documents, précisons, qu'en ce qui nous concerne, nous nous intéresserons à la traçabilité vers l'avant en particulier : nous établirons en quelque sorte une liste de références entre la spécification formelle décrite au moyen du langage **Albert II** et le programme en langage à relais de l'automate utilisé pour l'implémentation du système.



De la spécification formelle vers l'implémentation

La fin de ce dernier chapitre est consacrée à l'examen plus approfondi des différents liens entre les contraintes de la spécification Albert et l'implémentation du système sur l'automate. En effet, comme nous l'avons fait remarquer précédemment, il est important d'assurer une certaine "traçabilité" entre ces deux phases du développement : cette propriété est essentielle dans la mesure où elle facilite la maintenance du programme en cas de modification des spécifications.

En effet, ce qui nous importe est de savoir comment les différentes contraintes de la spécification se regroupent et sont implémentées dans l'automate; les références entre ces deux types de document permettent de mieux évaluer la répercussion de tout changement des spécifications sur le programme, de mieux cibler la présence des contraintes spécifiées dans la liste d'instructions de l'automate.

Dans un premier temps, rappelons que la spécification de notre application décrit trois agents en interaction : Opérateur, Environnement et Système. L'automate et le dispositif d'arrêt d'urgence (externe à l'automate) sont assimilables à l'agent "Système". En effet, tout comme l'agent perçoit les actions de l'opérateur et les modifications de l'environnement (vitesse du vent, moment, charge, fins de course), l'automate reçoit en entrée les informations correspondant aux commandes du pupitre et aux valeurs des différents capteurs.

Insistons également sur le fait que cette division en agents au niveau de la spécification est différente au niveau du programme : un agent peut être implémenté de diverses façons étant donné les ressources disponibles [ZEI95b].

A l'inverse de la spécification formelle structurée en agents, le programme de l'automate est constitué de lignes d'instructions exécutées les unes à la suite des autres et dont la disposition n'a pas de réelle importance. Néanmoins, les lignes se référant à une même partie de la grue ont été regroupées pour des raisons évidentes de lisibilité (cf. annexe 3); soit les "modules" suivants :

- **Commandes diverses** (Mise en service, frein de giration, mode dégradé, avertisseur)
- **Moteur de distribution**
- **Moteur de translation**
- **Moteur de giration**
- **Moteur de levage**

Avant de regrouper les contraintes de la spécification formelle et d'établir la correspondance avec les instructions du programme, nous avons jugé utile d'expliquer brièvement le contenu de chaque groupe de contraintes de l'agent "Système" de la spécification Albert.

Résumé des contraintes de l'agent "Système"

DERIVED COMPONENTS

Les trois indicateurs booléens ChargeAtteinte, MomentAtteint et VitesseVent Atteinte sont le résultat de la comparaison des valeurs mesurées (Charge, Moment, VitesseVent) avec les valeurs limites (ChargeMax, MomentMax, VitesseVentMax). Ces contraintes n'apparaissent pas explicitement dans le programme de l'automate suite à l'utilisation de capteurs à détection de seuil pour ces différentes mesures : le capteur compare la mesure à un seuil limite et renvoie une valeur booléenne à l'automate (seul type de donnée connu du SLC 150).

Rappelons que, dans l'hypothèse de l'utilisation d'un automate de plus haute gamme offrant des traitements numériques simples, ces relations mathématiques pourraient être facilement mises en œuvre.

Remarquons que les composants Charge, Moment et VitesseVent auraient pu figurer côté agent "Environnement" comme c'est le cas des indicateurs booléens de fins de course mais une spécification Albert impose que tout composant dérivé ne peut dépendre que de composants du même agent.

INITIAL VALUATION

L'ensemble de ces contraintes précise que les composants du premier état de la vie de l'agent "Système" sont initialisés à "faux" : ceci traduit notamment l'arrêt des moteurs et le fait que la grue est hors service, l'arrêt d'urgence non enclenché, le mode dégradé inactif, le frein relâché et l'avertisseur sonore éteint.

STATE BEHAVIOUR

Une partie de ces contraintes décrit ce qu'implique la mise hors service de la grue : frein de giration inactif, charge inférieure à un certain pourcentage du maximum autorisé, crochet relevé au sommet, chariot au pied du mât et mode de fonctionnement normal (non dégradé). Au niveau de l'automate, ceci se retrouve évidemment à la ligne du programme chargée de la mise hors service de la grue.

Les deux autres contraintes précisent que le moteur de levage a trois vitesses et qu'un arrêt du moteur correspond nécessairement à la vitesse "zéro".

EFFECTS OF ACTIONS

Sont reprises sous cet en-tête les actions apportant une modification des valeurs des états de l'agent "Système" : entre autres, celles engendrées par le grutier - l'agent "Opérateur" selon la terminologie de la spécification - et les trois actions de l'environnement renseignant sur les changements de vitesse du vent, de charge et de moment.

Les autres actions - internes à l'agent "Système" - concernent le retour en mode normal, la mise hors service de la grue et la modification des composants d'état relatifs aux différents moteurs ainsi que celle de leurs indicateurs de fin de temporisation.

CAPABILITY

Cet ensemble de contraintes exprime l'arrêt automatique des moteurs de distribution, de levage et de translation lors de l'arrivée en fin de course du chariot, du crochet ou du châssis de la grue ou encore lorsque, le mode dégradé n'étant pas actif, les contraintes relatives à la charge et au moment (voire la vitesse du vent pour la translation) ne sont plus vérifiées.

Notons également l'occurrence de l'action EmettreSon lorsque la grue étant en service, le mode dégradé est actif ou la vitesse du vent est supérieure au maximum admissible. Conditions que l'on retrouve, il va de soi, dans l'implémentation au niveau de la ligne activant l'avertisseur sonore.

L'ensemble des conditions nécessaires à l'arrêt automatique du mode dégradé (action RetourModeNormal) apparaît également au sein de ce type de contraintes.

ACTION COMPOSITION

Mis à part la mise en marche manuelle de l'avertisseur sonore, toutes les actions composées figurant sous cet en-tête concernent la succession des actions assurant le bon fonctionnement des moteurs. Ces contraintes sont utilisées en étroite relation avec les contraintes de durée d'action et de perception décrites ci-dessous.

ACTION DURATION

Comme son nom l'indique, cet ensemble de contraintes détermine la durée des occurrences d'actions internes à l'agent "Système". Elles sont importantes car elles précisent notamment la durée des temporisations.

Par exemple, l'action BloquerGirationDroite a pour effet (cf. Effects of actions) de remettre le composant d'état FinTempoGirationDroite à TRUE mais cet effet n'a lieu que lorsque l'occurrence de l'action est terminée, c'est à dire au bout de 3 secondes.

ACTION PERCEPTION

Ce groupe de contraintes est sans aucun doute le plus important de tous puisqu'il exprime les conditions sous lesquelles une action du grutier est perçue ou non; en d'autres mots, c'est à ce niveau que se trouve la gestion proprement dite des commandes de la grue.

On retrouve notamment la contrainte spécifiant que le grutier ne peut pas enclencher une vitesse du moteur de levage autre que celle immédiatement supérieure ou inférieure à celle en cours de fonctionnement. De même, la perception d'une commande de mise en mouvement d'un moteur ne peut se faire que si ce dernier est à l'arrêt et que la temporisation de blocage due à un mouvement précédent en sens inverse est terminée.

Notons que le système perçoit toujours un changement de charge, de moment ou de vitesse du vent en provenance de l'environnement.

STATE PERCEPTION, ACTION INFORMATION & STATE INFORMATION

Ces dernières contraintes stipulent que le système ne perçoit les informations en provenance de l'environnement que lorsqu'il est en service. De même, le grutier ne perçoit la charge, le moment, la vitesse du vent et le signal sonore que lorsque la grue est en service.

Liens entre la spécification formelle et le programme

Nous subdivisons ce dernier point selon la structure observée par le programme en citant, pour chaque partie du système, les types de contraintes rencontrés dans la spécification Albert et leur équivalence au niveau du programme.

Mise en/hors service

Les contraintes définies sous l'intitulé "ACTION PERCEPTION" définissent les conditions sous lesquelles l'opérateur peut exécuter la mise en marche et la mise hors service du système. Ces conditions apparaissent directement dans les lignes d'instructions commandant le démarrage ou l'arrêt de la grue.

$\mathcal{XK}$(Operateur.On / ¬ GrueEnService)

```
| Rung: 003 Mise en service de la grue (premier appui sur [0])                |
|                                                                             |
| APPUI GR EN                                                        GR EN    |
| ONOFF SERVI                                                        SERVI    |
|  802   011                                                          011     |
+--] [---]\[-------------------------------------------------------( L )----+
```

$\mathcal{XK}$(Operateur.Off / GrueEnService ∧ ¬ FreinGirationOn
∧ Environnement.LimiteMonteeCharge ∧ Environnement.LimiteChariotArriere
∧ ¬ ModeDegrade ∧ (Charge < PourcentageChargeMax * ChargeMax))

```
| Rung: 004 Mise hors service de la grue (second appui sur [0])               |
|                                                                             |
| APPUI FREIN CHAR  CROCH MODE  CHARG GR EN                          GR EN    |
| ONOFF GIRAT SECU  HAUT  DEGRA < 20% SERVI                          SERVI    |
|  802   116   105   102   800   202   011                            011     |
+--] [---]\[---] [---] [---]\[---] [---] [-----------------------------( U )----+
```

Les contraintes "STATE BEHAVIOUR" expriment ce qu'implique la mise hors service de la grue et ne font que confirmer les conditions précédentes. Les contraintes sous l'en-tête "EFFECTS OF ACTIONS" modifient les composants d'état de l'agent "Système" suite aux actions de l'opérateur; il en est de même pour le programme : les effets de l'appui sur ON/OFF sont respectivement d'activer et de désactiver la sortie 011 (GR EN SERVI).

Les contraintes de l'ensemble "ACTION COMPOSITION" précisent qu'une demande d'arrêt de la machine est immédiatement suivie d'un arrêt de tous les moteurs. Au niveau du programme, cette dernière spécification est répercutée dans les lignes d'instructions responsables de l'activation de chaque moteur : un moteur ne peut être actionné que si la grue est en service.

Frein de giration

Deux types de contraintes spécifient le fonctionnement de ce dispositif de freinage : les contraintes "EFFECTS OF ACTIONS" modifiant le composant indiquant l'état actif ou inactif du frein en fonction des actions du grutier, et les contraintes "ACTION PERCEPTION" désignant les conditions devant être réunies pour que le grutier puisse engager ou relâcher ce frein et qui sont également clairement reprises dans l'implémentation.

$$\mathcal{XK}(\text{Operateur.EnclencherFreinGiration} / \text{GrueEnService} \wedge \neg\, \text{FreinGirationOn} \wedge \text{MoteurGiration} = \text{Stop})$$

```
| Rung: 007 Enclenchement du frein de giration (premier appui sur [F])          |
|                                                                               |
| APPUI FREIN -MOT- GR EN                                              FREIN    |
| FREIN GIRAT GIRAT SERVI                                              GIRAT    |
|  804   116   114   011                                                116     |
+--] [---]\[---]\[---] [-----------------------------------------------( L )----+
```

$$\mathcal{XK}(\text{Operateur.RelacherFreinGiration} / \text{GrueEnService} \wedge \text{FreinGirationOn})$$

```
| Rung: 008 Declenchement du frein de giration (second appui sur [F])           |
|                                                                               |
| APPUI FREIN GR EN                                                    FREIN    |
| FREIN GIRAT SERVI                                                    GIRAT    |
|  804   116   011                                                      116     |
+--] [---] [---] [-----------------------------------------------------( U )----+
```

Mode dégradé

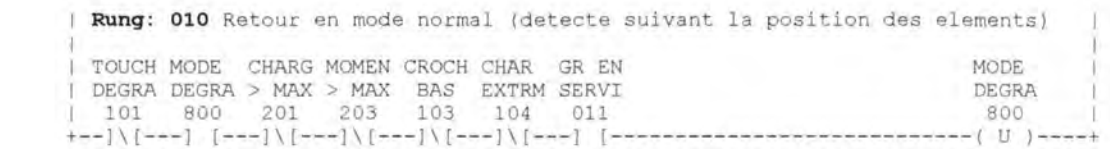
Pour le passage en mode dégradé, on retrouve une fois de plus la traduction en langage à relais de la contrainte "ACTION PERCEPTION".

$$\mathcal{XK}(\text{Operateur.EnclencherModeDegrade} / \text{GrueEnService} \wedge \neg\, \text{ModeDegrade})$$

```
| Rung: 009 Passage en mode degrade (appui sur [Md])                            |
|                                                                               |
| TOUCH MODE  GR EN                                                    MODE     |
| DEGRA DEGRA SERVI                                                    DEGRA    |
|  101   800   011                                                      800     |
+--] [---]\[---] [-----------------------------------------------------( L )----+
```

Pour le retour en mode normal, la contrainte de "CAPABILITY" exprime les conditions de désactivation automatique du mode dégradé. On retrouve ces conditions dans le programme comme les contraintes de perception d'action vues jusqu'à présent.

$$\mathcal{XO}\ (\text{RetourModeNormal} / \text{ModeDegrade} \wedge \neg\ \text{ChargeAtteinte} \wedge \neg\ \text{MomentAtteint} \wedge \neg\text{Environnement.LimiteDescenteCharge} \wedge \neg\text{Environnement.LimiteChariotAvant} \wedge \text{GrueEnService})$$



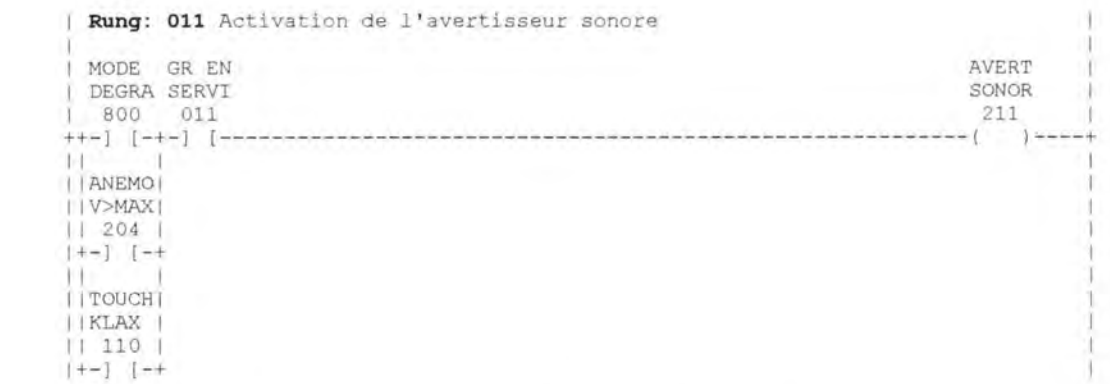
Une fois de plus, remarquons que les effets des actions sont implicites dans le programme : si la continuité logique est vérifiée sur la ligne, l'instruction "de sortie" est automatiquement exécutée. Dans la spécification, on retrouve explicitement cette modification d'état dans le groupe de contraintes intitulé "EFFECTS OF ACTIONS".

Avertisseur sonore

L'avertisseur sonore est modélisé différemment dans la spécification et le programme. Une action composée "Klaxonner" décrite dans "ACTION COMPOSITION" énonce que l'action de mise en marche manuelle de l'avertisseur est suivie de l'émission du son, elle-même obligatoirement suivie de l'action d'arrêt.

Une contrainte de "CAPABILITY" exprime quant à elle les autres situations occasionnant la mise en marche de l'avertisseur sonore.

$$\mathcal{O}\ (\text{EmettreSon} / (\text{ModeDegrade} \vee \text{VitesseVentAtteinte}) \wedge \text{GrueEnService} \wedge \neg\ \text{KlaxonOn})$$



Cette ligne de programme exprime les mêmes idées : l'avertisseur est activé durant tout le temps que la touche du klaxon est enfoncée ou que le mode dégradé est actif ou encore que la vitesse du vent dépasse sa valeur limite.

Au sujet des contraintes "EFFECTS OF ACTIONS" et "ACTION PERCEPTION", les remarques sont les mêmes que celles énoncées aux points précédents.

Commandes des moteurs

De nouveau, les contraintes "EFFECTS OF ACTIONS" ne transparaissent pas dans l'implémentation; elles évoquent les valeurs que prennent les composants d'état de l'agent "Système" lors de la survenance des différentes actions relatives au fonctionnement des moteurs.

Les contraintes de "CAPABILITY" énoncent les conditions d'arrêt automatique des moteurs : lorsque le chariot, le crochet ou le châssis de la grue arrive en fin de course; lorsque la charge, le moment ou la vitesse du vent dépasse sa valeur limite à condition que le mode dégradé ne soit pas actif. Ces événements sont repris directement dans les lignes responsables de l'activation des moteurs : si une des instructions devient fausse, la continuité logique est rompue et la sortie n'est plus alimentée.

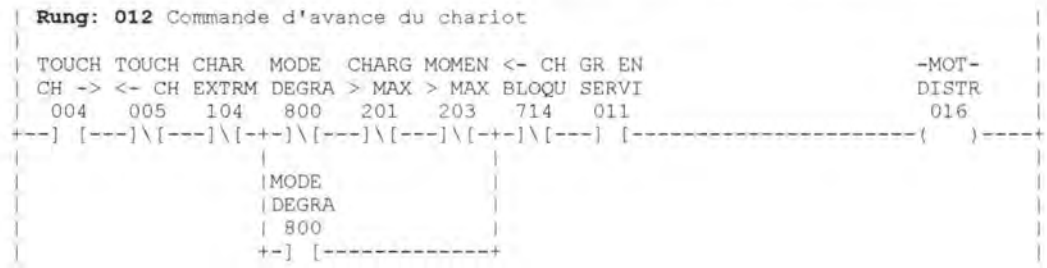
Pour illustrer nos propos, nous ne reprenons ci-dessous que les contraintes et les parties de code se rapportant à la commande de déplacement du chariot, le principe étant identique pour les commandes de translation et de giration. Le levage offre quant à lui quelques particularités supplémentaires étant donné la possibilité de manoeuvrer selon trois vitesses différentes.

$$O\,(\,\text{EnclencherChariot (Stop)} / \text{MoteurChariot} \neq \text{Stop} \wedge \text{GrueEnService}$$
$$\wedge\,(\,\text{Environnement.LimiteChariotAvant} \vee \text{Environnement.LimiteChariotArriere}$$
$$\vee\,(\text{MoteurChariot} = \text{Avant} \wedge \neg\,\text{ModeDegrade}$$
$$\wedge\,(\text{ChargeAtteinte} \vee \text{MomentAtteint}))\,)\,)$$

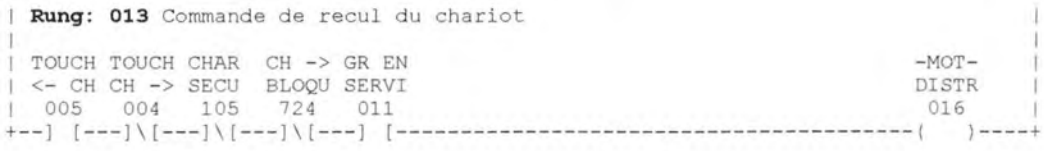
Sous le titre "ACTION DURATION", les contraintes de durée des actions "Bloquer..." succédant à l'arrêt d'un moteur, ont pour effet d'empêcher le grutier d'actionner le moteur en sens inverse pendant une courte période de temps.

En effet, lors de l'enclenchement d'un moteur, une variable booléenne "FinTempo..." est rendue fausse, ce qui a pour conséquence d'interdire l'inversion du sens de ce moteur jusqu'à la fin de l'occurrence de l'action "Bloquer..." correspondante (cf. Effects of actions). Cette interdiction apparaît dans les contraintes "ACTION PERCEPTION" qui figurent une fois de plus en toutes lettres dans le programme. Des variables de blocage de mouvement en sens inverse sont construites sur base du front descendant de la sortie activant le moteur.

$$\mathcal{XK}\,(\text{Operateur.CderChariotAvant} / \text{GrueEnService} \wedge$$
$$\neg\,\text{Environnement.LimiteChariotAvant} \wedge (\,\text{ModeDegrade} \vee (\neg\,\text{MomentAtteint} \wedge$$
$$\neg\,\text{ChargeAtteinte})\,) \wedge \text{MoteurChariot} = \text{Stop} \wedge \text{FinTempoChariotArriere}\,)$$



$\mathcal{XK}$(Operateur.CderChariotArriere / GrueEnService ∧
¬ Environnement.LimiteChariotArriere ∧ MoteurChariot = Stop
∧ FinTempoChariotAvant)



Les contraintes "ACTION COMPOSITION" décrivent l'enchaînement des actions inhérent à l'utilisation de tout moteur : la commande du grutier enclenche le moteur jusqu'à ce que celui-ci ait cessé d'agir sur la commande, qu'un arrêt d'urgence soit apparu ou que le système ait provoqué automatiquement l'arrêt (arrivée en fin de course, maximum atteint pour la charge, le moment ou la vitesse). Tout arrêt du moteur est suivi d'un blocage du démarrage en sens inverse.

Terminons en rappelant l'existence d'une temporisation supplémentaire pour le moteur de levage, destinée à interdire de passer trop rapidement d'une vitesse à une autre. On retrouve également sous l'en-tête "ACTION PERCEPTION" l'obligation du grutier de passer à la vitesse immédiatement supérieure ou inférieure à la vitesse en cours.

Conclusion

Arrivé au terme de ce mémoire, il apparaît clairement que la rédaction d'une spécification formelle d'un système n'est pas une tâche négligeable. Elle implique l'observation minutieuse du comportement du système dans toutes les situations.

Remarquons qu'il serait faux de déclarer que le logiciel résultant de l'utilisation de méthodes formelles est parfait : une spécification formelle est un modèle du monde réel, elle peut donc contenir des contresens, des erreurs ou des omissions !

Toutefois, cette approche est efficace parce qu'elle rend les erreurs de spécification plus faciles à détecter et fournit une base non ambiguë pour la conception du système.

En résumé, les avantages de la spécification formelle sont les suivants :

- elle permet de mieux comprendre les besoins du logiciel et fournit une base non-ambiguë pour une conception élégante;
- il est possible de la faire analyser par des méthodes mathématiques et de prouver sa complétude, sa consistance et qu'une implémentation lui est conforme;
- on dispose d'outils logiciels pour assister son développement et sa mise au point;
- elle est d'autant plus facile à maintenir qu'elle est structurée en agents indépendants, comme l'est toute spécification décrite à l'aide du langage Albert II.

Mais elle présente aussi quelques inconvénients :

- la spécification formelle augmente les coûts de développement et devrait réduire plus tard les coûts de maintenance : cette rentabilité n'est pas évidente et beaucoup se montrent réticents à l'utilisation de la spécification formelle car l'effort nécessaire pour l'écrire ne semble pas justifier les avantages qu'ils pourraient en tirer;
- elle nécessite une certaine familiarité avec les mathématiques et la logique ainsi qu'un effort d'apprentissage des symboles spécialisés de la notation.

J'ose supposer que des personnes intéressées par la réalisation technique d'un tel système trouveront ici une aide précieuse en raison de l'approche formelle et des explications données.

Le premier chapitre avait pour objet de présenter le fonctionnement général des grues à tour et, suite aux dangers encourus lors de leur manipulation, délimiter les objectifs devant être réalisés par un système informatique destiné à gérer les commandes de ces machines.

Le deuxième chapitre abordait les déclarations graphiques de la spécification formelle Albert de ce système.

Le troisième chapitre de ce mémoire était consacré aux automates programmables et plus particulièrement à la description d'un de leur langage de programmation.

Enfin, au cours du dernier chapitre, nous nous sommes attardés sur la question essentielle de la traçabilité entre la spécification formelle et l'implémentation.

Je ne voudrais pas terminer cette conclusion sans exprimer quelques recommandations dans l'éventualité d'une réelle mise en œuvre du système.

D'une part, l'automate utilisé pour le contrôle de la grue doit pouvoir effectuer des mesures analogiques et des comparaisons numériques pour la charge, le moment et la vitesse du vent.

D'autre part, je tiens à faire remarquer que le système décrit tout au long de ce mémoire ne constitue qu'un prototype dans le sens où il ne répond qu'à un sous-ensemble des problèmes à résoudre en vue d'améliorer la sécurité des hommes travaillant parmi ces machines. Entre autres, il serait intéressant d'étudier de manière plus approfondie les aspects suivants :

- informer en permanence l'opérateur de la charge et du moment afin de lui permettre de mieux évaluer les situations à risques;
- mesurer et éviter la surintensité des moteurs;
- pouvoir contrôler automatiquement l'état des différents capteurs;
- éviter le basculement de la grue aussi bien au repos que dans des conditions normales de charge et de fonctionnement en mesurant la stabilité;
- empêcher un nombre de rotations de la flèche pouvant mener à la rupture des câbles d'alimentation;
- enregistrer dans une boîte noire, l'heure, la succession des manœuvres réalisées et l'identité du grutier à chaque fonctionnement en mode dégradé.

Annexes

Annexe

1

Le langage Albert II

Concepts utilisés dans la syntaxe concrète d'Albert II

Modèles d'une spécification

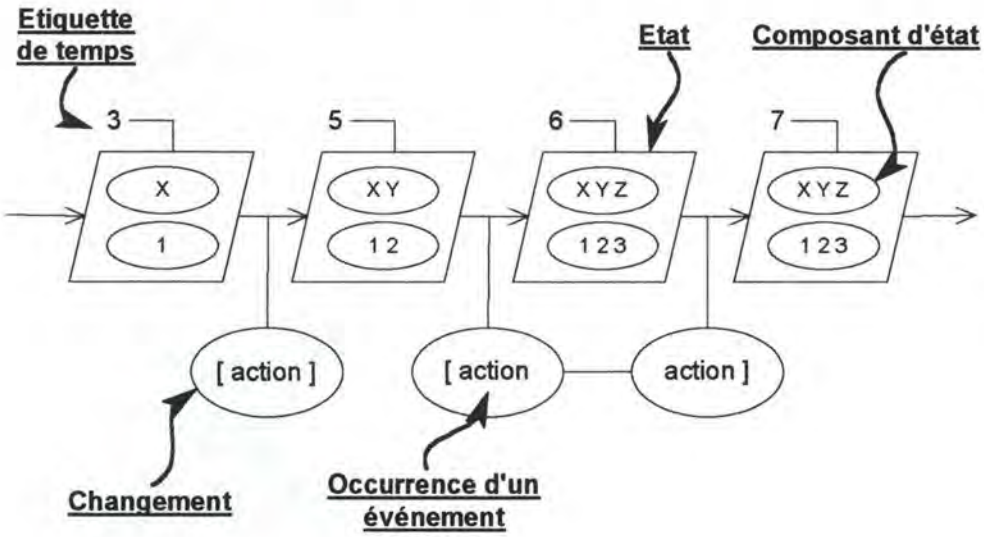
Le but de ce langage de spécification des besoins est de décrire le comportement admissible d'un système composite (composé d'éléments hétérogènes, des composants software et/ou hardware jusqu'aux composants humains et/ou mécaniques). Cette description est appelée une spécification du système. Une spécification Albert II est structurée en termes d'agents (unité de base), les agents étant groupés en sociétés.

Si cette spécification décrit fidèlement le système, les comportements admissibles du système sont des modèles de la spécification.

De manière à maîtriser leur complexité, les modèles sont considérés à 2 niveaux :

- au niveau de l'agent : un ensemble de comportements possibles est associé à chaque agent indépendamment du comportement des autres agents;
- au niveau de la société : les interactions entre les agents sont prises en compte et entraînent des restrictions supplémentaires sur le comportement de chaque agent.

La spécification décrit un agent en définissant un ensemble de "vies admissibles" modélisant tous ses comportements possibles (chaque vie admissible est un modèle de la spécification). Une vie est une succession de changements et d'états, chaque état étant étiqueté d'une valeur augmentant au cours de la durée de vie de l'agent (fig. A.1). La suite d'états faisant partie d'une vie admissible de l'agent est appelée "trace", la suite de changements constitue quant à elle "l'historique". La valeur d'un état à un instant donné peut toujours être dérivée de l'historique des changements survenus jusqu'à ce moment.



~ Figure A.1 : Vie admissible d'un agent ~

Un état est structuré en termes de composants d'état.

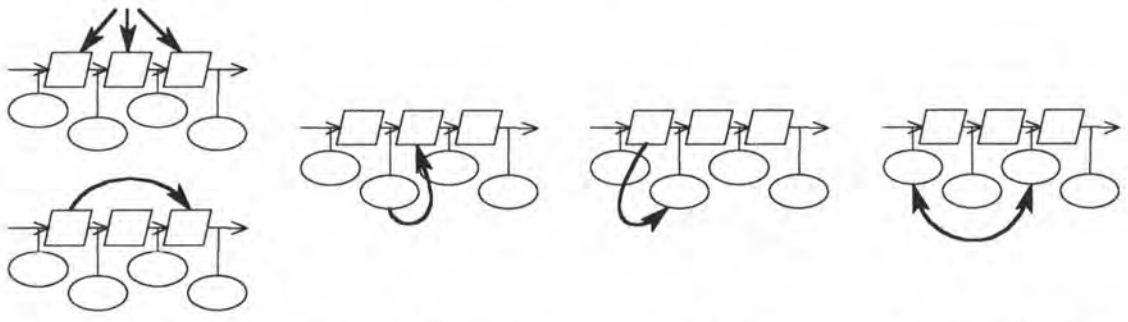
Un changement est composé de zéro ou plusieurs événements simultanés (deux événements appartiennent à un même changement s'ils apparaissent durant l'intervalle de temps délimité par la valeur de l'état précédent et celle de l'état suivant ce changement et ne sont donc pas forcément simultanés au sens propre).

Un événement correspond soit
- à l'apparition d'une action instantanée, notée]action[;
- au début de l'occurrence d'une action, notée]action;
- à la fin de l'occurrence d'une action, notée action[;

Une action peut avoir une durée; l'effet sur l'état a lieu, dans ce cas, dans le premier état suivant sa terminaison. Le mot "action" désigne à la fois les événements ayant un effet sur l'état suivant la terminaison de l'action et ceux sans influence directe sur l'état adjacent. Remarquons que deux occurrences d'actions agissant dans un même intervalle de temps ne peuvent être concurrentes, c'est-à-dire ne peuvent affecter le même composant d'état.

Le langage Albert II est constitué de constructions pour introduire des déclarations et exprimer des contraintes. Un premier ensemble de vies possibles associées à un agent peut être dérivé des déclarations, cet ensemble est ensuite réduit par les contraintes en un nouveau sous-ensemble de vies admissibles.

Les contraintes réduisent l'ensemble des vies admissibles de quatre manières différentes (fig. A.2), en déclarant une relation
1) devant être vérifiée par tous les états ou entre les états de la trace d'un agent;
2) entre la fin de l'occurrence d'une action et l'état résultant adjacent;
3) entre l'état d'un agent et le début des occurrences d'action du changement adjacent;
4) entre les changements de l'historique d'un agent.



~ Figure A.2 : Typologie des contraintes ~

Contraintes basées sur l'état / basées sur la vie

Une des propriétés importantes du langage Albert II est sa capacité de traiter ces deux types de contraintes, c'est-à-dire de supporter les styles de spécification opérationnel et déclaratif.

Les contraintes basées sur l'état (style opérationnel) expriment une relation entre deux états consécutifs, entre un état et le changement survenant dans cet état ou entre un changement et l'état résultant.

Les contraintes basées sur la vie (style déclaratif) se rapportent aux changements ou états étendus sur toute la vie d'un agent. Elles peuvent exprimer le fait que certains états peuvent induire d'autres états dans le futur et/ou dans le passé de la vie de l'agent ou décrire certaines propriétés qui restent vraies durant la vie de l'agent.

Syntaxe concrète

Une spécification Albert II est constituée de plusieurs types de contraintes qui supportent l'analyste dans l'écriture de formules complexes et cohérentes. Ces patrons ont été introduits pour des raisons méthodologiques afin de guider l'analyste dans sa manière de spécifier la dynamique des systèmes complexes.

Ecrire une spécification Albert II implique deux activités : l'écriture des déclarations introduisant le vocabulaire de l'application considérée et l'écriture des contraintes, propositions logiques qui identifient les comportements possibles du système composite et écartent ceux non désirés. Seules les déclarations peuvent être exprimées par la syntaxe graphique.

Chaque type de contraintes est classé dans une des quatre catégories que voici :

- les déclarations;
- les contraintes de base;
- les contraintes locales;
- les contraintes de coopération.

A la figure A.3, une liste exhaustive de tous les patrons possibles, classés par ordre d'apparition dans une spécification.

Déclarations

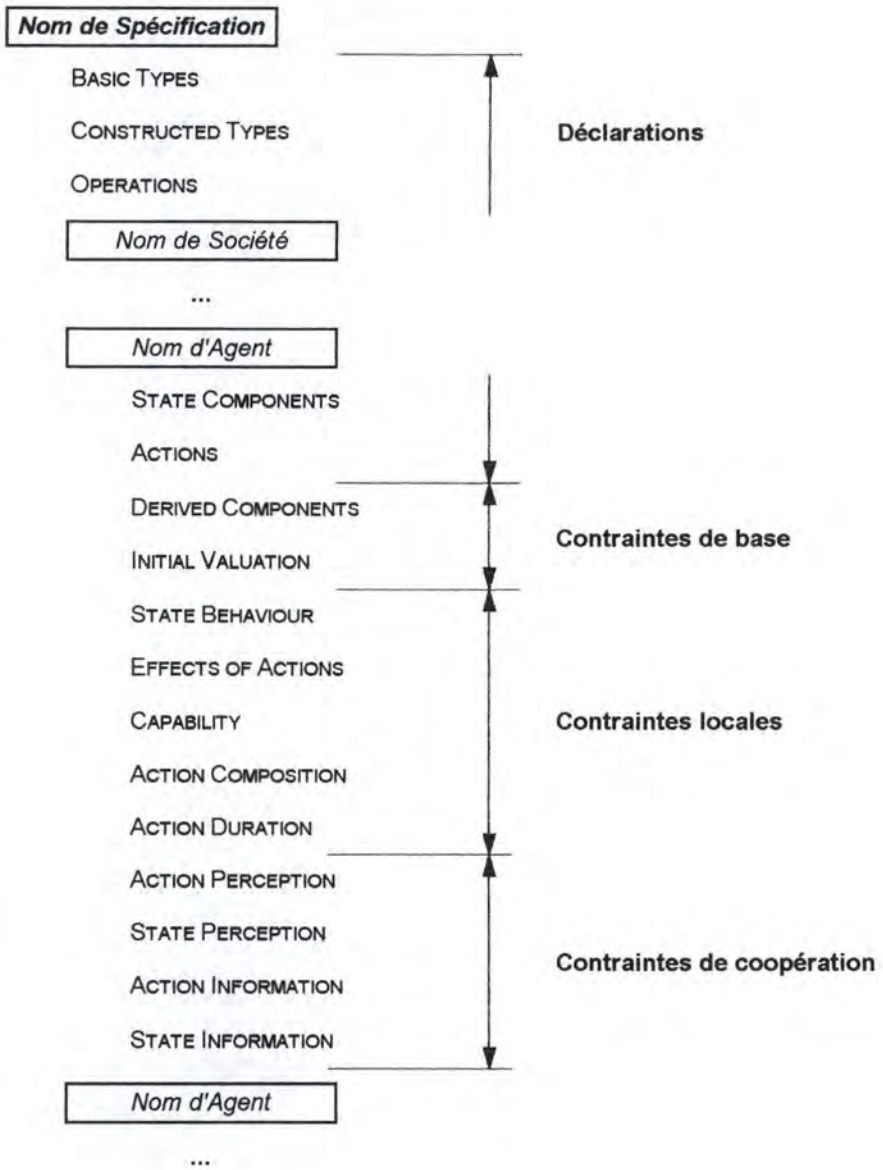
Les types de données et leurs opérations

Albert II est un langage fortement typé : l'état des agents et les paramètres des actions sont structurés grâce à des types abstraits. Ces types de données peuvent être :

- des types élémentaires prédéfinis : BOOLEAN, INTEGER, RATIONAL, CHAR et STRING;
- des types élémentaires définis par l'analyste sous l'intitulé "BASIC TYPES";
- des types composés sous l'intitulé "CONSTRUCTED TYPES" : ces types complexes peuvent être créés par l'analyste en utilisant des constructeurs de type prédéfinis tels que le produit cartésien (cp), l'ensemble (set), la suite (seq), l'amas (bag), la table, l'union et l'énumération (enum), ces constructeurs pouvant être appliqués récursivement sur les types élémentaires et composés. L'ensemble des éléments d'un type composé peut être restreint grâce à une clause "with" liant ses composants;
- des types correspondants aux identificateurs d'agents et de sociétés : un type est automatiquement associé à chaque classe d'agents et de sociétés, un nom de type est dérivé du nom de l'agent ou de la société en remplaçant toutes les lettres minuscules par des majuscules.

Pour tous les types, les opérateurs = et ≠ sont automatiquement définis mais l'utilisateur peut en définir d'autres dans une spécification, regroupés sous l'en-tête "OPERATIONS". Les opérations peuvent être définies récursivement et leur ordre de déclaration n'a pas d'importance (une opération peut se référer à une autre déclarée précédemment).

Les opérations sur les types de données ne doivent pas être confondues avec les actions des agents : elles ne constituent que des fonctions mathématiques utilisées pour simplifier les expressions des contraintes mais ne peuvent modéliser le comportement dynamique des agents.

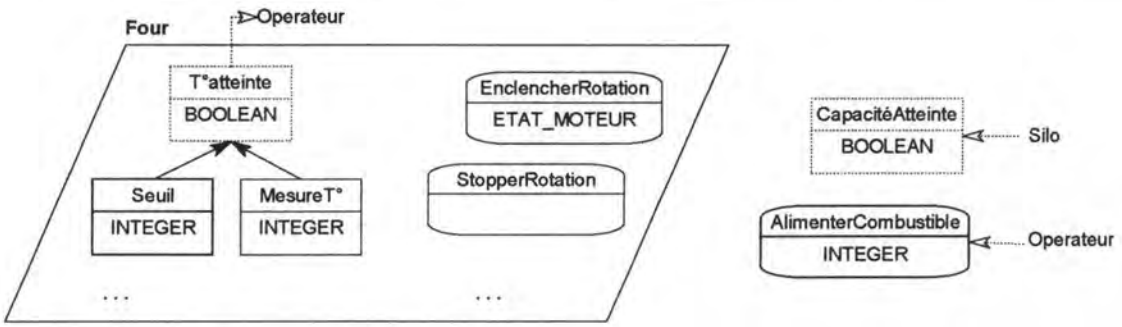


~ Figure A.3 : Squelette d'une spécification Albert II ~

Les agents (structure)

Ils peuvent être individuels (une et une seule instance dans le système à modéliser) ou membres de classes (au moins une instance mais il peut y en avoir d'avantage). La partie déclaration d'un agent consiste en la description de sa structure d'états et la liste des actions dont peut être fait son historique.

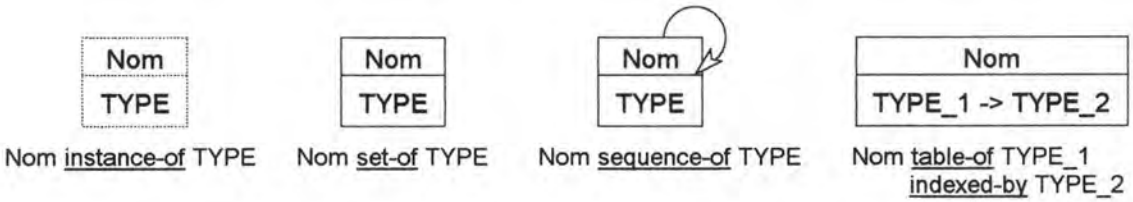
Comme illustré à la figure A.4, les composants d'état sont représentés à l'aide de rectangles et les actions au moyen de rectangles aux coins arrondis, le tout regroupé au sein d'un parallélogramme. Les propriétés des composants d'état et les liens d'importation et d'exportation entre agents y sont également représentés.



~ Figure A.4 : Exemple de déclaration graphique d'un agent ~

Il est possible de transposer les déclarations graphiques sous forme textuelle mais cela en diminue leur lisibilité puisqu'elles sont éclatées en plusieurs parties, reprises sous divers patrons; qui plus est, seules les exportations sont évoquées afin de limiter la redondance.

La structure de l'état d'un agent est définie en termes de zéro ou plusieurs **composants d'état** déclarés textuellement sous l'en-tête "STATE COMPONENTS". Ces composants d'état peuvent être considérés comme des individus ou des populations (ensemble, suite ou table d'individus). Leur notation est représentée à la figure A.5.



~ Figure A.5 : Notations graphique et textuelle associées aux composants d'état ~

Les composants d'état sont constants ou varient au cours du temps : graphiquement, les boîtes des composants constants apparaissent en trait gras, textuellement, leur nom est précédé d'une étoile "*".

La valeur d'un composant d'état peut être dérivée des valeurs d'autres composants d'état du même agent. Cependant, un composant ne peut être dérivé de lui-même ni

directement ni indirectement via d'autres composants dérivés et, un composant constant ne peut être dérivé que de composants constants. Graphiquement, une flèche est tracée du composant d'état vers celui qui en dépend. Textuellement, le nom du composant d'état dérivé est suivi du mot "derived-from" et des noms des composants d'état dont il dépend.

Un mécanisme statique d'exportation permet de rendre les valeurs des composants d'état visibles à d'autres agents. Un composant rendu visible à une société est visible par tous les agents et sociétés la composant. Un composant peut être exporté vers le même agent au sein duquel il est déclaré : dans le cas d'un agent individuel, cela n'a aucun effet mais si cet agent est membre d'une classe, les autres instances de cette classe le perçoivent.

Graphiquement, une flèche est tracée du composant d'état exporté vers l'extérieur du parallélogramme, le bout de la flèche pointant sur le nom de l'agent ou de la société vers lequel le composant est exporté. La vue graphique permet aussi de voir les composants importés visibles par l'agent : ceux-ci figurent à l'extérieur de la boîte de déclaration de l'agent, sur les côtés pour informer qu'ils ne font pas partie de cet agent. Chaque composant importé doit être exporté de l'agent exportateur et chaque composant exporté doit être importé vers l'agent receveur.

Textuellement, le nom du composant d'état exporté est suivi d'une flèche "→" et de la liste des noms d'agents vers lesquels il est exporté (chaque nom séparé d'une virgule).

Les actions exécutées par l'agent et représentées graphiquement par les rectangles aux coins arrondis ou énoncées textuellement au sein du patron "ACTIONS", peuvent, quant à elles, avoir des arguments dont l'ordre a de l'importance. De même que pour les composants d'état, le mécanisme statique d'exportation permet de rendre des actions visibles à d'autres agents (les remarques à ce sujet sont identiques à celles formulées pour les composants d'état).

On peut restreindre une action, c'est-à-dire forcer ses occurrences à ne survenir que dans le contexte d'une action composée (cf. composition d'action). On peut spécifier dans quels agents cette propriété est observée, y compris l'agent dans lequel l'action est déclarée. Si une action est restreinte au sein d'un agent, les vies de cet agent dans lesquelles cette action apparaît seule ne font partie d'aucun modèle de la spécification.

Graphiquement, la boîte d'une action restreinte est représentée en trait gras mais si cette action est restreinte dans un autre agent que celui dans lequel elle est déclarée, sa boîte ne figure sous cette forme que du côté de l'agent dans lequel elle est restreinte. En d'autres mots, la restriction n'apparaît que du côté de l'agent concerné.

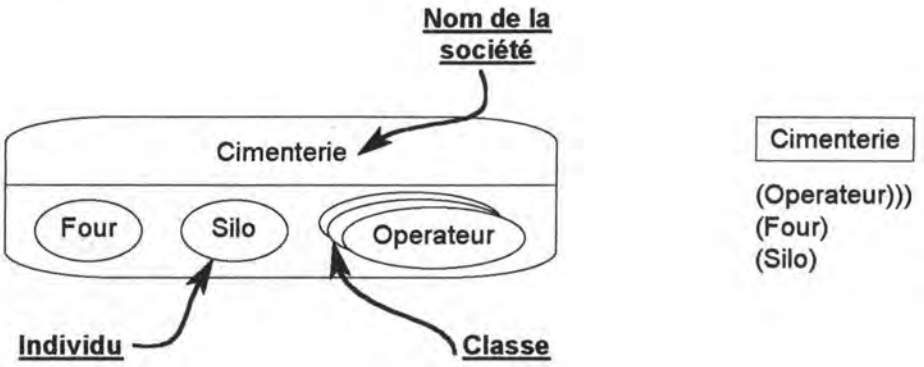
Dans la représentation textuelle, les actions restreintes dans l'agent dans lequel elles sont déclarées ont leur nom précédé d'une étoile (*ACTION) tandis que, pour celles restreintes dans d'autres agents, c'est le nom de l'agent qui est suivi d'une étoile (ACTION→ AGENT*). La restriction n'apparaît que du côté de l'agent dans lequel l'action est déclarée.

Les sociétés

Comme les agents, elles peuvent être individuelles ou membres de classes. Une société correspond en fait à un regroupement d'agents voire de sociétés afin de construire une société plus grande.

Une spécification peut donc être considérée comme une structure hiérarchique en arbre où les feuilles sont des agents et les branches, des sociétés. Une spécification Albert II est composée d'au moins un agent et tout agent ou société autre que la racine doit faire partie d'une et une seule société.

Ci-joint (fig. A.6), une société et ses composants représentés graphiquement et textuellement.



~ Figure A.6 : Exemple de déclarations graphique et textuelle d'une société ~

Soulignons le fait qu'on ne peut connaître le type d'un agent ou d'une société (individu ou classe) qu'en observant la description de la société qui l'englobe : en effet, la définition d'un agent ou d'une société n'indique pas s'il s'agit d'une classe ou non. De plus, une spécification composée d'un seul agent n'explicite pas si cet agent est de type classe; par convention, ce dernier sera considéré en tant qu'individu.

Remarquons également que la structure d'une société n'indique pas si ses composants sont des agents ou des sociétés.

Identificateurs, conflits de noms et commentaires

Suite aux déclarations qui viennent d'être présentées, rappelons brièvement les quelques règles pour la création des identificateurs :

- noms de TYPE : en majuscules, commençant obligatoirement par deux lettres;
- noms de variables : en minuscules;
- noms d'Opérations, de Champs de produits cartésiens, de Constantes, d'Agents, de Sociétés, d'Actions et de Composants d'état : mixtes, commençant par une majuscule et contenant au moins une minuscule.

Tous peuvent contenir des chiffres ainsi que le caractère de soulignement "_".

Qu'ils soient prédéfinis, construits par l'utilisateur ou dérivés des noms des agents et sociétés, tous les types doivent posséder un nom unique. De plus, étant donné que des noms de types en sont automatiquement dérivés, il faut veiller à ce que les noms d'agents et de sociétés soient différents des noms de types déjà existant : par exemple, on ne peut nommer un agent "Integer".

Les noms des opérations, des champs des produits cartésiens, des constantes définies par l'utilisateur, des agents et des sociétés doivent tous être uniques entre eux.

Le nom d'un composant d'état (ou d'une action) peut être identique à celui utilisé pour une opération, un champ de produit cartésien, une constante définie par l'utilisateur, un agent ou une société. Si c'est le cas, notons qu'une utilisation d'un tel nom dans une expression est supposé faire référence au composant d'état.

Cependant, les noms des composants d'état (resp. actions) doivent être uniques au sein de l'ensemble des composants d'état (resp. actions) du même agent mais des composants d'état (resp. actions) de même nom peuvent exister dans différents agents. D'autre part, les composants d'état et les actions d'un même agent peuvent porter le même nom sans ambiguïté.

Remarque sur les commentaires : la syntaxe Albert II autorise l'utilisation de commentaires précédés d'une barre verticale "|". Ceux-ci n'ont évidemment aucun impact sur l'ensemble des modèles d'une spécification.

Contraintes sur les agents

Elles sont utilisées pour diminuer l'ensemble infini de vies possibles d'un agent, ensemble obtenu suite à la déclaration de sa structure.

Albert II n'a pas de sémantique opérationnelle. Une vie doit être considérée dans son entièreté avant de pouvoir être considérée comme admissible et l'ajout de nouveaux états et changements à la fin d'une vie admissible ne conduit pas nécessairement à une vie admissible (la vie d'une spécification est admissible si et seulement si toutes les contraintes sont vérifiées ensemble dans l'entièreté du système).

Comme énoncé antérieurement, ces contraintes sont classées sous onze en-têtes groupées en trois familles (fig. A.3) afin de fournir un certain guidage méthodologique à l'analyste.

Contraintes de base

Les composants dérivés

sous l'intitulé "DERIVED COMPONENTS"
But : donner les règles de dérivation des composants dérivés de l'agent.

Ces contraintes sont une manière syntaxique de simplifier des expressions : une expression complexe peut être remplacée par un nom. Elles définissent une relation mathématique entre les composants dérivés et les autres composants d'état de l'agent.

Chaque composant déclaré précédemment comme dérivé doit avoir une règle de dérivation sous cet en-tête. La règle doit utiliser tous les composants cités dans la déclaration du composant dérivé. Réciproquement, chaque composant ayant une règle de

dérivation doit avoir été déclaré comme dérivé et dépendant de tous les composants figurant dans la règle.

Un composant dérivé ne peut dépendre que de composants d'état du même agent (valeurs de composants importés interdites). Un composant constant ne peut être dérivé que de composants constants. Un composant ne peut pas être dérivé de lui-même que ce soit directement ou indirectement via d'autres composants.

L'évaluation initiale

sous l'intitulé "INITIAL VALUATION"
But : décrire l'état initial de l'agent.

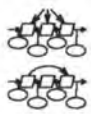
Ces contraintes donnent les valeurs des composants d'état pour le premier état de la trace de l'agent. Chaque composant initialisé ne peut l'être plus d'une fois et doit avoir été déclaré sous la rubrique "STATE COMPONENTS". Aucun composant dérivé ne peut être initialisé et réciproquement, la valeur utilisée pour l'initialisation des composants classiques ne peut être calculée à partir des valeurs des autres composants d'état qu'ils soient de l'agent ou importés.

L'initialisation n'est pas obligatoire : si un composant n'est pas initialisé, les modèles de la spécification peuvent avoir des traces avec n'importe quelle valeur possible pour ce composant.

Contraintes locales

Les contraintes locales se rapportent au comportement interne de l'agent et sont réparties en cinq groupes.

Le comportement d'état



sous l'intitulé "STATE BEHAVIOUR"
But : exprimer un invariant sur l'état de l'agent ou une contrainte sur l'évolution de son état.

Ces contraintes permettent d'exprimer les propriétés des états ou les propriétés liant les états de la trace d'une vie admissible d'un agent. Ces propriétés sont vérifiées dans tout modèle. Elles ne peuvent contraindre que les valeurs de composants d'état préalablement déclarés et appartenant au même agent. Les références aux composants d'état importés sont autorisées tant que leurs valeurs ne sont pas contraintes. On distingue notamment :

- les contraintes vraies pour tous les états des traces possibles d'un agent (invariants) : elles sont écrites selon les règles habituelles de la logique du premier ordre au moyen des connectives logiques $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\forall$, $\exists$. Toute variable hors de la portée d'un quantificateur est considérée comme quantifiée universellement.

- les contraintes sur l'évolution du système, se rapportant à des états à différents moments : des connectives temporelles additionnelles permettent de se référer à plus d'un état à la fois; f, f1 et f2 sont des déclarations :

♦f est vraie ssi f vraie un moment dans le passé (y compris le présent);
◊f est vraie ssi f vraie un moment dans le futur (y compris le présent);
■f est vraie ssi f toujours vraie dans le passé (y compris le présent);
□f est vraie ssi f toujours vraie dans le futur (y compris le présent);
f1 $\mathcal{U}$ f2 est vraie ssi f1 est vraie (à l'instant présent) jusqu'à ce que f2 soit vraie;
f1 $\mathcal{S}$ f2 est vraie ssi f1 est vraie (à l'instant présent) depuis que f2 est vraie.

- les contraintes concernant l'expression des propriétés temps-réel : pour décrire les retards et les temporisations écoulées (*time-outs*), exprimées en affectant une période de temps aux connectives présentées ci-dessus. Cette période est précisée au moyen d'une unité de temps (de la nanoseconde à l'année, fig. A.8) et peut exprimer une durée minimale, exacte ou maximale :

  $\blacklozenge_{</=/>X}$ f est vraie ssi f est vraie
  un moment dans le passé il y a **moins de / exactement / plus de** X unités de temps;

  $\Diamond_{</=/>X}$ f est vraie ssi f est vraie
  un moment dans le futur **dans les / dans exactement / après** X unités de temps;

  $\blacksquare_{</=/>X}$ f est vraie ssi f est vraie
  depuis **moins de / exactement / plus de** X unités de temps;

  $\square_{</=/>X}$ f est vraie ssi f vraie
  pendant **moins de / exactement / plus de** X unités de temps.

Les effets des actions

sous l'intitulé "EFFECTS OF ACTIONS"
But : décrire les effets des occurrences d'actions internes ou importées sur l'état de l'agent.

Seules les actions apportant un changement sur les états sont décrites sous cette rubrique (il peut exister des actions sans aucun effet sur les composants d'état).

Ce patron est le seul décrivant les modifications des valeurs des composants d'état.

Deux actions simultanées ne peuvent avoir d'effets contradictoires c'est-à-dire ne peuvent agir sur les mêmes composants d'état ou, de manière équivalente, deux actions ayant de tels effets ne peuvent être simultanées dans un modèle (et donc deux occurrences de la même action ayant certains effets sur des composants d'état ne peuvent survenir simultanément).

L'effet d'une action est exprimé par une propriété caractérisant l'état suivant la fin de l'occurrence de l'action. La valeur d'un composant dans l'état résultant est définie par une relation entre les arguments de l'action, l'agent responsable de cette action et l'état de la trace précédant le commencement de l'occurrence de l'action.

La déclaration exprimant l'effet d'une action est composée de deux parties : la première (membre de gauche) indique quelle occurrence de quelle action est considérée, la seconde (membre de droite) est une liste d'affectations d'expressions aux composants d'état internes. A l'apparition de l'occurrence de l'action, toutes les expressions du membre de droite sont évaluées et dès la terminaison de l'occurrence de l'action, chaque composant de la liste est affecté du résultat correspondant à son expression précédemment évaluée.

En fait, du démarrage de l'occurrence d'une action jusqu'à sa terminaison, les composants d'état modifiés par cette occurrence gardent leurs valeurs inchangées : les effets se produisent à la fin de l'occurrence de l'action. Par exemple, si au commencement de l'occurrence A = 0 et B = 3, puis l'effet [A = B+1; B = A+5] se réalise, alors, à la fin de l'occurrence A = 4 et B = 5.

Les capacités

sous l'intitulé "CAPABILITY"
But : décrire les conditions de survenance des actions internes.

Ces contraintes décrivent les relations explicites entre l'accomplissement d'une condition au niveau de la trace et le commencement de l'occurrence d'une action au niveau de l'historique.

Trois opérateurs permettent d'exprimer des obligations, des interdictions et des obligations exclusives :

- l'obligation "O (action interne / situation)" exprime le fait que l'action doit survenir si les conditions exprimées dans la situation sont vérifiées;
- l'interdiction "$\mathcal{F}$ (action interne / situation)" exprime le fait qu'une occurrence de l'action ne peut pas démarrer si les conditions exprimées par la situation sont vérifiées;
- l'obligation exclusive "χO (action interne / situation)" est équivalente à la combinaison "O (action interne / situation)" et "$\mathcal{F}$ (action interne / $\neg$ situation)". L'action se déclenche si et seulement si la situation est vérifiée.

La règle par défaut est que toutes les actions sont permises quelle que soit la situation : elles peuvent apparaître à certains moments et non à d'autres. Le tableau suivant (fig. A.7) illustre ces opérateurs : l'action est "prendre la voiture", la condition requise est "le temps ensoleillé".

	par défaut	O(🚗 /☼)	$\mathcal{F}$(🚗 /☼)	χO(🚗 /☼)
☼	🚗 🚫	🚗	🚫	🚗
💧	🚗 🚫	🚗 🚫	🚗 🚫	🚫

~ Figure A.7 : Signification des connectives ~

La composition d'action

sous l'intitulé "ACTION COMPOSITION"
But : définir comment une action interne se décompose en un ensemble de sous-actions internes et/ou importées.

Une contrainte de composition d'action consiste en un nom d'action suivi d'une double flèche et d'une expression où des actions sont combinées ensemble grâce à des combinateurs exprimant :

- la composition séquentielle "A1 ; A2 ; ..." : une occurrence A1 qui, une fois terminée, est suivie d'une occurrence A2, elle-même suivie, etc. Cette suite n'est pas nécessairement continue (temps morts entre occurrences);

- la répétition "{ A }^{n}" : n occurrences de A, chacune démarrant après la terminaison de la précédente éventuellement après un certain laps de temps;

- la composition simultanée "A1 ⊗ A2 ⊗ ..." : une occurrence A1 et une occurrence A2 et une occurrence..., toutes démarrant et se terminant aux mêmes moments;

- la composition parallèle "A1 || A2 || ..." : une occurrence A1 et une occurrence A2 et une occurrence... dans n'importe quel ordre, pouvant se manifester au même moment;

- l'alternative "A1 ⊕ A2 ⊕ ..." : une occurrence A1 ou une occurrence A2 ou une occurrence..., le "ou" étant exclusif. L'identificateur d'action DAC (*dummy action*) peut être utilisé pour exprimer le fait qu'une action composée peut apparaître suite au déclenchement d'une autre action ou de sa propre initiative (A1 ↔ A2 ⊕ ... ⊕ DAC).

Les actions composées peuvent elles-mêmes être les composantes d'actions composées de plus haut niveau mais il est interdit de décrire des compositions contenant des cycles, c'est-à-dire des actions composées d'elles-mêmes que ce soit directement ou indirectement via d'autres compositions d'actions : par exemple, A1 est composée de A2 où A2 est également une action composée... de A1 !

Certaines actions peuvent être restreintes, une telle action ne peut survenir hors du contexte d'une action composée (que si son occurrence fait partie de l'occurrence d'une action composée). Une action localement restreinte est une action ne pouvant pas se manifester hors du contexte d'une composition dans son agent, mais elle peut survenir de sa propre initiative dans d'autres agents sans faire partie d'une occurrence d'action composée. Une action restreinte dans un agent donné ne peut pas du tout, quant à elle, voir ses effets se réaliser sans ceux de l'action composée dont elle fait partie.

La durée d'action

sous l'intitulé "ACTION DURATION"
But : contraindre la durée des occurrences d'actions internes.

La contrainte peut être :

- une durée précise | action | = durée (si la durée est nulle, l'action est instantanée);
- une limite inférieure | action | > durée;
- une limite supérieure | action | < durée;
- l'union des deux contraintes précédentes.

Ce type de contraintes diminue fortement le nombre de modèles valides. Si aucune contrainte de ce genre n'est donnée sur la durée d'une occurrence d'action alors toute durée possible doit être prise en compte. Remarquons de plus que contraindre la durée d'une action composée revient indirectement à contraindre la durée de ses composants.

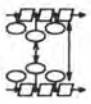
Il faut cependant être prudent car trop de contraintes sur la durée des actions peuvent inhiber toutes les occurrences d'actions.

La limite de durée d'une action est déterminée grâce aux unités de temps prédéfinies énoncées ci-dessous (fig. A.8).

Unité	Syntaxe	Unité	Syntaxe
nanoseconde (10^{-9})	ns	minute	m, min, '
microseconde (10^{-6})	us, µs	heure	h, hour, hours
milliseconde	ms	jour	d, day, days
centième de seconde	cs	semaine	w, week, weeks
dixième de seconde	ds	mois	month, months
seconde	s, sec, secs,"	année	y, year, years

~ Figure A.8 : Unités de temps ~

Contraintes de coopération



Cette famille de contraintes spécifie comment un agent interagit avec son environnement : comment il informe les autres agents des actions qu'il exécute, comment il montre ses états aux autres agents, comment il perçoit les actions exécutées par les autres agents et comment il peut voir leurs états.

Ces contraintes sont exprimées au moyen des trois opérateurs, analogues à ceux rencontrés dans les capacités : la connaissance $\mathcal{K}$, l'ignorance I et la connaissance exclusive $\mathcal{XK}$.

La perception d'action

sous l'intitulé "ACTION PERCEPTION"
But : déterminer quand les actions des autres agents sont perçues par l'agent.

Ces contraintes définissent la sensibilité de l'agent aux changements apparaissant dans son environnement et dus aux autres agents :

- la connaissance "$\mathcal{K}$ (agent . action externe / situation)" exprime le fait que, si la situation est vérifiée, le comportement de l'agent courant est influencé par l'action émise par l'agent externe, c'est-à-dire que toute occurrence de cette action est perçue;

- l'ignorance "I (agent . action externe / situation)" exprime le fait que, si la situation est vérifiée, le comportement de l'agent n'est pas influencé par l'action de l'agent externe, une occurrence de cette action n'est donc pas perçue;

- la connaissance exclusive "$\mathcal{XK}$ (agent . action externe / situation)" est équivalente à la combinaison de "$\mathcal{K}$ (agent . action externe / situation)" et de "I (agent . action externe / $\neg$ situation)"; l'occurrence de l'action est perçue si et seulement si la situation est vérifiée.

La règle par défaut est que toutes les actions importées peuvent être perçues quelle que soit la situation. Chaque possibilité (action perçue ou non) doit être prise en considération à tout instant possible afin de déduire les modèles valides.

La perception d'état

sous l'intitulé "STATE PERCEPTION"
But : déterminer quand les parties de l'état des autres agents sont visibles par l'agent.

Ces contraintes définissent comment l'agent voit les parties d'état que les autres agents lui rendent accessibles :

- la connaissance "$\mathcal{K}$ (agent . composant / situation)" exprime le fait que, si la situation est vérifiée, la valeur du composant de l'agent est perçue;

- l'ignorance "I (agent . composant / situation)" exprime le fait que, si la situation est vérifiée, la valeur du composant de l'agent n'est pas perçue;

- la connaissance exclusive "$\mathcal{XK}$ (agent . composant / situation)" équivaut à l'union de "$\mathcal{K}$ (agent . composant / situation)" et "I (agent . composant / $\neg$ situation)"; la valeur du composant de l'agent est perçue si et seulement si la situation est reconnue.

La règle par défaut est que tout composant d'état importé peut être perçu quelle que soit la situation. Chaque possibilité (composant perçu ou non) doit être prise en considération à tout instant possible afin de déduire les modèles valides.

L'information d'action

sous l'intitulé "ACTION INFORMATION"
But : déterminer quand les actions de l'agent sont visibles par les autres agents.

Ces contraintes définissent comment les occurrences des actions exécutées par un agent sont rendues accessibles aux autres agents :

- la connaissance "$\mathcal{K}$(action interne . agent / situation)" signifie que, si la situation est vérifiée, les occurrences de l'action interne sont perceptibles par l'agent;

- l'ignorance "I (action interne . agent / situation)" signifie que, si la situation est vérifiée, les occurrences de l'action interne ne sont pas perceptibles par l'agent;

- la connaissance exclusive "$\mathcal{XK}$ (action interne . agent / situation)" est l'union de "$\mathcal{K}$(action interne . agent / situation)" et "I (action interne . agent / $\neg$ situation)"; les occurrences de l'action interne sont perceptibles par l'agent si et seulement si les conditions exprimées dans la situation sont vérifiées.

La règle par défaut est que toute action exportée vers un agent peut être perçue par ce dernier et ce quelle que soit la situation. Chaque possibilité (action perçue ou non) doit être prise en considération à tout instant possible afin de déduire les modèles valides.

L'information d'état

sous l'intitulé "STATE INFORMATION"
But : déterminer quand les composants d'état de l'agent sont visibles par les autres agents.

Ces contraintes définissent comment l'agent montre les parties de son état aux autres agents :

- la connaissance "$\mathcal{K}$(composant . agent / situation)" signifie que, si la situation est vérifiée, la valeur du composant est accessible à l'agent;

- l'ignorance "I (composant . agent / situation)" signifie que, si la situation est vérifiée, la valeur du composant est cachée de l'agent;

- la connaissance exclusive "$\mathcal{XK}$ (composant . agent / situation)" est l'union de "$\mathcal{K}$(composant . agent / situation)" et "I (composant . agent / $\neg$ situation)"; la valeur du composant n'est accessible à l'agent que si et seulement si la situation est vérifiée.

La règle par défaut est que tout composant exporté vers un agent peut être perçu par ce dernier et ce quelle que soit la situation. Chaque possibilité (composant perçu ou non) doit être prise en considération à tout instant possible afin de déduire les modèles valides.

La clause with

Une clause formée du mot "with" et d'une expression logique permet de limiter les effets des 8 derniers types de contraintes :

- EFFECTS OF ACTIONS : elle spécifie le fait que la condition de démarrage d'une action peut différer selon la valeur des composants d'état;
- CAPABILITY : cette contrainte peut être déclarée ne s'appliquer qu'à un sous-ensemble des occurrences possibles d'une action, la clause restreint la portée de la contrainte selon la valeur des arguments de l'action;
- ACTION COMPOSITION : elle peut mettre une contrainte sur les arguments des actions impliquées dans la composition;
- ACTION DURATION : elle peut en restreindre la portée en donnant une relation entre les arguments et les composants d'état, la contrainte ne s'applique pas aux occurrences d'actions qui ne vérifient pas ces relations;
- ACTION PERCEPTION & INFORMATION : elle peut être utilisée pour restreindre la portée de la contrainte à un sous-ensemble d'occurrences d'action et/ou renforcer les conditions d'applicabilité dépendant des valeurs des composants d'état et des liens avec/entre arguments de l'action;
- STATE PERCEPTION & INFORMATION : elle peut être utilisée pour restreindre la portée de la contrainte à un sous-ensemble des composants et/ou renforcer les conditions d'applicabilité dépendant des valeurs des composants d'état;

De telles clauses doivent être vérifiées dans tous les modèles de la spécification.

Inspiré du manuel de référence d'Albert II, ce document ne constitue qu'une brève description du langage. Notamment, ce résumé n'évoque pas les différents points énumérés ci-dessous :

- la liste des opérations prédéfinies sur chaque type de données;
- la priorité et l'associativité des opérateurs;
- la liste des constantes prédéfinies de chaque type de données;
- les syntaxes ASCII et non-ASCII (pretty) pour la représentation des constructions;
- **la syntaxe concrète du langage** (notation BNF).

Tous ces sujets figurent respectivement aux chapitres 3, 4, 6, 7 et **10** du manuel de référence [JUN96].

Annexe

2

Spécification formelle en Albert II

SPECIFICATION Etude de cas GRUE

Constructed types

| Ensemble des entiers naturels
NATURAL = INTEGER
|! with $\forall$ n : n $\geq$ 0

| Etats des moteurs de translation (chariot et châssis de la grue)
ETAT_TRANSL = ENUM[Arriere, Stop, Avant]

| Etats du moteur de giration
ETAT_GIR = ENUM[Gauche, Stop, Droite]

| Vitesse du moteur de levage
VIT_MOT = NATURAL
|! with $\forall$ n : n $\leq$ 3

| Etats du moteur de levage : vitesse et sens
ETAT_LEV = CP[Vit : VIT_MOT, Sens : ENUM[Haut, Stop, Bas]]
|! with Sens = Stop $\Leftrightarrow$ Vit = 0

SOCIETY ChantierSecurisant

(OPERATEUR)
(SYSTEME)
(ENVIRONNEMENT)

AGENT Operateur

Actions

On → SYSTEME
Off → SYSTEME*
Stop → SYSTEME*

CderMonteeCharge (VIT_MOT) → SYSTEME*
CderDescenteCharge (VIT_MOT) → SYSTEME*
CderStopLevage → SYSTEME*

CderChariotAvant → SYSTEME*
CderChariotArriere → SYSTEME*
CderStopChariot → SYSTEME*

CderTranslationAvant → SYSTEME*
CderTranslationArriere → SYSTEME*
CderStopTranslation → SYSTEME*

CderGirationGauche → SYSTEME*
CderGirationDroite → SYSTEME*
CderStopGiration → SYSTEME*

EnclencherFreinGiration → SYSTEME
RelacherFreinGiration → SYSTEME

EnclencherKlaxon → SYSTEME*
RelacherKlaxon → SYSTEME

EnclencherModeDegrade → SYSTEME

AGENT Environnement

State components

| Indique si la limite de montée la de charge est atteinte
LimiteMonteeCharge **instance-of** BOOLEAN → SYSTEME
| Indique si la limite de descente de la charge est atteinte
LimiteDescenteCharge **instance-of** BOOLEAN → SYSTEME

| Indique si le chariot est en bout de flèche
LimiteChariotAvant **instance-of** BOOLEAN → SYSTEME
| Indique si le chariot est au pied de la grue
LimiteChariotArriere **instance-of** BOOLEAN → SYSTEME

| Indique si la limite de translation avant est atteinte
LimiteTranslationAvant **instance-of** BOOLEAN → SYSTEME
| Indique si la limite de translation arrière est atteinte
LimiteTranslationArriere **instance-of** BOOLEAN → SYSTEME

Actions

ChgtMonteeCharge (BOOLEAN)
ChgtDescenteCharge (BOOLEAN)

ChgtChariotAvant (BOOLEAN)
ChgtChariotArriere (BOOLEAN)

ChgtTranslationAvant (BOOLEAN)
ChgtTranslationArriere (BOOLEAN)

ChgtVitesseVent (NATURAL) → SYSTEME

ChgtMoment (NATURAL) → SYSTEME

ChgtCharge (NATURAL) → SYSTEME

Effects of actions

ChgtMonteeCharge (v) : LimiteMonteeCharge := v
ChgtDescenteCharge (v) : LimiteDescenteCharge := v
ChgtChariotAvant (v) : LimiteChariotAvant := v
ChgtChariotArriere (v) : LimiteChariotArriere := v
ChgtTranslationAvant (v) : LimiteTranslationAvant := v
ChgtTranslationArriere (v) : LimiteTranslationArriere := v

Action information

| L'environnement est évidemment fiable...
𝒳𝒦 (ChgtVitesseVent(_).Systeme / TRUE)
𝒳𝒦 (ChgtMoment(_).Systeme / TRUE)
𝒳𝒦 (ChgtCharge(_).Systeme / TRUE)

State information

| L'environnement est évidemment fiable...
𝒳𝒦 (LimiteMonteeCharge.Systeme / TRUE)
𝒳𝒦 (LimiteDescenteCharge.Systeme / TRUE)
𝒳𝒦 (LimiteChariotAvant.Systeme / TRUE)
𝒳𝒦 (LimiteChariotArriere.Systeme / TRUE)
𝒳𝒦 (LimiteTranslationAvant.Systeme / TRUE)
𝒳𝒦 (LimiteTranslationArriere.Systeme / TRUE)

AGENT Systeme

State components

| Indique si la grue est alimentée et correctement montée sur un bon sol (prête à fonctionner)
GrueEnService **instance-of** BOOLEAN

| Etat du moteur levant/abaissant les charges
MoteurLevage **instance-of** ETAT_LEV

| Etat du moteur déplaçant le chariot
MoteurChariot **instance-of** ETAT_TRANSL

| Etat du moteur déplaçant la grue sur ses rails
MoteurTranslation **instance-of** ETAT_TRANSL

| Etat du moteur de giration
MoteurGiration **instance-of** ETAT_GIR

| Indique si le frein de giration est enclenché
FreinGirationOn **instance-of** BOOLEAN

| Indique si l'opérateur a demandé de forcer la grue dans une situation dangereuse
ModeDegrade **instance-of** BOOLEAN

| Indique si le stop d'urgence a été enclenché
StopUrgence **instance-of** BOOLEAN

| Indique si le klaxon est enclenché
KlaxonOn **instance-of** BOOLEAN

| Charge perçue
Charge **instance-of** NATURAL → OPERATEUR

| Moment perçu
Moment **instance-of** NATURAL → OPERATEUR

| Vitesse du vent perçue
VitesseVent **instance-of** NATURAL → OPERATEUR

| Charge maximale pouvant être levée
*ChargeMax **instance-of** NATURAL

| Moment maximal permis
*MomentMax **instance-of** NATURAL

| Vitesse maximale du vent ne compromettant pas la sécurité de la grue
*VitesseVentMax **instance-of** NATURAL

| Indique si la charge maximale est atteinte
ChargeAtteinte **instance-of** BOOLEAN **derived-from** Charge, ChargeMax

| Indique si le moment maximal est atteint
MomentAtteint **instance-of** BOOLEAN **derived-from** Moment, MomentMax

| Indique si la vitesse du vent maximale est atteinte
VitesseVentAtteinte **instance-of** BOOLEAN **derived-from** VitesseVent, VitesseVentMax

| Pourcentage de la charge maximale autorisé sur le câble lorsque la grue est hors service
*PourcentageChargeMax **instance-of** RATIONAL

| Indicateurs de fin des temporisations utilisées pour la manipulation correcte des moteurs
FinTempoMontee **instance-of** BOOLEAN
FinTempoDescente **instance-of** BOOLEAN
FinTempoVitesseSup **instance-of** BOOLEAN

FinTempoChariotAvant **instance-of** BOOLEAN
FinTempoChariotArriere **instance-of** BOOLEAN

FinTempoTranslationAvant **instance-of** BOOLEAN
FinTempoTranslationArriere **instance-of** BOOLEAN

FinTempoGirationDroite **instance-of** BOOLEAN
FinTempoGirationGauche **instance-of** BOOLEAN

Actions

| Enchaînement d'arrêt (off)
EnchainementOff

| Enchaînement d'arrêt d'urgence (stop)
EnchainementStop

*StopperMoteurs

| Enclencher le moteur de levage
*EnclencherLevage (ETAT_LEV)
| Activer les temporisations
*BloquerMonteeCharge
*BloquerDescenteCharge
*BloquerVitesseSup

```
| Processus de montée et de descente de la charge
EnchainMonteeCharge
EnchainDescenteCharge
| Processus de fin de levage
*FinMontee (NATURAL)
*FinDescente (NATURAL)
*ArretMontee (NATURAL)
*ArretDescente (NATURAL)

| Enclencher le déplacement du chariot
*EnclencherChariot (ETAT_TRANSL)
| Activer les temporisations
*BloquerChariotAvant
*BloquerChariotArriere
| Processus de déplacement du chariot vers l'avant et vers l'arrière
EnchainChariotAvant
EnchainChariotArriere
| Processus de fin de déplacement du chariot
*FinChariot

| Enclencher le moteur de translation de la grue
*EnclencherTranslation (ETAT_TRANSL)
| Activer les temporisations
*BloquerTranslationAvant
*BloquerTranslationArriere
| Processus de translation avant et arrière
EnchainTranslationAvant
EnchainTranslationArriere
| Processus de fin de translation
*FinTranslation

| Enclencher la giration
*EnclencherGiration (ETAT_GIR)
| Activer les temporisations
*BloquerGirationDroite
*BloquerGirationGauche
| Processus de giration vers la gauche et vers la droite
EnchainGirationGauche
EnchainGirationDroite
| Processus de fin de giration
*FinGiration

| Avertisseur sonore
EmettreSon → OPERATEUR, ENVIRONNEMENT
| Activité continue du klaxon
Klaxonner

| Retour en situation normale
RetourModeNormal
```

Derived components

| Indique s'il y a surcharge de levage
ChargeAtteinte == Charge > ChargeMax

| Indique s'il y a dépassement de moment
MomentAtteint == Moment > MomentMax

| Indique si la vitesse du vent a dépassé sa valeur limite
VitesseVentAtteinte == VitesseVent > VitesseVentMax

Initial Valuation

| Au départ, la grue est hors service dans une situation non dégradée, moteurs et frein à l'arrêt
GrueEnService = FALSE
StopUrgence = FALSE
ModeDegrade = FALSE
FreinGirationOn = FALSE
KlaxonOn = FALSE
MoteurLevage = <0, Stop>
MoteurChariot = Stop
MoteurTranslation = Stop
MoteurGiration = Stop

State behaviour

¬ GrueEnService ⇒ ¬ FreinGirationOn ∨ StopUrgence
¬ GrueEnService ⇒ (Charge < PourcentageChargeMax * ChargeMax) ∨ StopUrgence
¬ GrueEnService ⇒ Environnement.LimiteMonteeCharge ∨ StopUrgence
¬ GrueEnService ⇒ Environnement.LimiteChariotArriere ∨ StopUrgence
¬ GrueEnService ⇒ ¬ ModeDegrade ∨ StopUrgence

(MoteurLevage.Vit ≥ 0 ∧ MoteurLevage.Vit ≤ 3)
MoteurLevage.Sens = Stop ⇒ MoteurLevage.Vit = 0

Effects of actions

Operateur.EnclencherFreinGiration : FreinGirationOn := TRUE
Operateur.RelacherFreinGiration : FreinGirationOn := FALSE
Operateur.EnclencherModeDegrade : ModeDegrade := TRUE
Operateur.EnclencherKlaxon : KlaxonOn := TRUE
Operateur.RelacherKlaxon : KlaxonOn := FALSE
Operateur.Stop : StopUrgence := TRUE

Environnement.ChgtVitesseVent (v) : VitesseVent := v
Environnement.ChgtMoment (m) : Moment := m
Environnement.ChgtCharge (c) : Charge := c

RetourModeNormal : ModeDegrade := FALSE

EnchainementOff : GrueEnService := FALSE
EnchainementStop : GrueEnService := FALSE

Operateur.On : GrueEnService := TRUE ; StopUrgence := FALSE ; KlaxonOn := FALSE ;
FinTempoMontee := TRUE ; FinTempoDescente := TRUE ;
FinTempoVitesseSup := TRUE ;
FinTempoChariotAvant := TRUE ; FinTempoChariotArriere := TRUE ;
FinTempoTranslationAvant := TRUE ; FinTempoTranslationArriere := TRUE ;
FinTempoGirationDroite := TRUE ; FinTempoGirationGauche := TRUE ;

EnclencherChariot (Avant) : MoteurChariot := Avant ; FinTempoChariotAvant := FALSE
EnclencherChariot (Arriere) : MoteurChariot := Arriere ; FinTempoChariotArriere := FALSE
EnclencherChariot (Stop) : MoteurChariot := Stop
Operateur.CderStopChariot : MoteurChariot := Stop
BloquerChariotAvant : FinTempoChariotAvant := TRUE
BloquerChariotArriere : FinTempoChariotArriere := TRUE

EnclencherTranslation (Avant) : MoteurTranslation := Avant ;
FinTempoTranslationAvant := FALSE
EnclencherTranslation (Arriere) : MoteurTranslation := Arriere ;
FinTempoTranslationArriere := FALSE
EnclencherTranslation (Stop) : MoteurTranslation := Stop
Operateur.CderStopTranslation : MoteurTranslation := Stop
BloquerTranslationAvant : FinTempoTranslationAvant := TRUE
BloquerTranslationArriere : FinTempoTranslationArriere := TRUE

EnclencherGiration (Droite) : MoteurGiration := Droite ; FinTempoGirationDroite := FALSE
EnclencherGiration (Gauche) : MoteurGiration:=Gauche ; FinTempoGirationGauche := FALSE
EnclencherGiration (Stop) : MoteurGiration := Stop
Operateur.CderStopGiration : MoteurGiration := Stop
BloquerGirationDroite : FinTempoGirationDroite := TRUE
BloquerGirationGauche : FinTempoGirationGauche := TRUE

EnclencherLevage (<v, Haut>) : MoteurLevage:= <v, Haut> ; FinTempoVitesseSup:= FALSE ;
FinTempoMontee := FALSE
EnclencherLevage (<v, Bas>) : MoteurLevage := <v, Bas> ; FinTempoVitesseSup := FALSE ;
FinTempoDescente := FALSE
EnclencherLevage (<0, Stop>) : MoteurLevage := <0, Stop>
Operateur.CderStopLevage : MoteurLevage := <0, Stop>
BloquerMonteeCharge : FinTempoMontee := TRUE
BloquerDescenteCharge : FinTempoDescente := TRUE
BloquerVitesseSup : FinTempoVitesseSup := TRUE

Capability

| Arrêt automatique quand limite atteinte ou problème

O (EnclencherLevage (<0, Stop>) / Sens(MoteurLevage) ≠ Stop ∧ GrueEnService
∧ (Environnement.LimiteMonteeCharge ∨ Environnement.LimiteDescenteCharge
∨ (MoteurLevage = Haut ∧ ¬ ModeDegrade ∧ (ChargeAtteinte ∨ MomentAtteint))))

O (EnclencherChariot (Stop) / MoteurChariot ≠ Stop ∧ GrueEnService
∧ (Environnement.LimiteChariotAvant ∨ Environnement.LimiteChariotArriere
∨ (MoteurChariot = Avant ∧ ¬ ModeDegrade ∧ (ChargeAtteinte ∨ MomentAtteint))))

O (EnclencherTranslation (Stop) / MoteurTranslation ≠ Stop ∧ GrueEnService
∧ (Environnement.LimiteTranslationAvant ∨ Environnement.LimiteTranslationArriere
∨ (¬ ModeDegrade ∧ (ChargeAtteinte ∨ MomentAtteint ∨ VitesseVentAtteinte))))

O (EmettreSon / (ModeDegrade ∨ VitesseVentAtteinte) ∧ GrueEnService ∧ ¬ KlaxonOn)

XO (RetourModeNormal / ModeDegrade
∧ ¬ ChargeAtteinte
∧ ¬ MomentAtteint
∧ ¬ Environnement.LimiteDescenteCharge
∧ ¬ Environnement.LimiteChariotAvant
∧ GrueEnService

Action composition

EnchainementOff ↔ Operateur.Off ; StopperMoteurs

EnchainementStop ↔ Operateur.Stop ; StopperMoteurs

StopperMoteurs ↔ EnclencherLevage (<0, Stop>) ||
EnclencherChariot (Stop) ||
EnclencherGiration (Stop) ||
EnclencherTranslation (Stop)

| Gestion de temporisation des moteurs avec possibilités d'arrêt
| (arrêt automatique ou stop d'urgence)

EnchainMonteeCharge ↔ Operateur.CderMonteeCharge (v) ; EnclencherLevage (v, Haut) ;
FinMontee (v) ; BloquerDescenteCharge
EnchainDescenteCharge ↔ Operateur.CderDescenteCharge (v) ; EnclencherLevage (v, Bas) ;
FinDescente (v) ; BloquerMonteeCharge
FinMontee (v) ↔ BloquerVitesseSup || ArretMontee (v)
ArretMontee (v) ↔ Operateur.CderStopLevage ⊕ (Operateur.CderMonteeCharge (v+1) ∨
Operateur.CderMonteeCharge (v-1)) ⊕ Operateur.Stop ⊕
EnclencherLevage (<0, Stop>)
FinDescente (v) ↔ BloquerVitesseSup || ArretDescente (v)
ArretDescente (v) ↔ Operateur.CderStopLevage ⊕ (Operateur.CderDescenteCharge (v+1) ∨
Operateur.CderDescenteCharge (v-1)) ⊕ Operateur.Stop ⊕
EnclencherLevage (<0, Stop>)

EnchainChariotAvant ↔ Operateur.CderChariotAvant ; EnclencherChariot (Avant) ;
FinChariot ; BloquerChariotArriere
EnchainChariotArriere ↔ Operateur.CderChariotArriere ; EnclencherChariot (Arriere) ;
FinChariot ; BloquerChariotAvant
FinChariot ↔ Operateur.CderStopChariot ⊕ Operateur.Stop ⊕ EnclencherChariot (Stop)

EnchainTranslationAvant ↔ Operateur.CderTranslationAvant ; EnclencherTranslation (Avant) ;
FinTranslation ; BloquerTranslationArriere
EnchainTranslationArriere ↔ Operateur.CderTranslationArriere; EnclencherTranslation(Arriere);
FinTranslation ; BloquerTranslationAvant
FinTranslation ↔ Operateur.CderStopTranslation ⊕ Operateur.Stop ⊕
EnclencherTranslation (Stop)

EnchainGirationGauche ↔ Operateur.CderGirationGauche ; EnclencherGiration (Gauche) ;
FinGiration ; BloquerGirationDroite
EnchainGirationDroite ↔ Operateur.CderGirationDroite ; EnclencherGiration (Droite) ;
FinGiration ; BloquerGirationGauche
FinGiration ↔ Operateur.CderStopGiration ⊕ Operateur.Stop

Klaxonner ↔ Operateur.EnclencherKlaxon ; EmettreSon ; Operateur.RelacherKlaxon

Action duration

| EnclencherLevage (_) | = 0s
| BloquerMonteeCharge | = 3s
| BloquerDescenteCharge | = 3s
| BloquerVitesseSup | = 3s

| EnclencherChariot (_) | = 0s

| BloquerChariotAvant | = 3s
| BloquerChariotArriere | = 3s

| EnclencherTranslation (_) | = 0s
| BloquerTranslationAvant | = 3s
| BloquerTranslationArriere | = 3s

| EnclencherGiration (_) | = 0s
| BloquerGirationDroite | = 3s
| BloquerGirationGauche | = 3s

Action perception

$\mathcal{XK}$(Environnement.ChgtVitesseVent (_) / TRUE)
$\mathcal{XK}$(Environnement.ChgtMoment (_) / TRUE)
$\mathcal{XK}$(Environnement.ChgtCharge (_) / TRUE)

$\mathcal{XK}$(Operateur.On / ¬ GrueEnService)

$\mathcal{XK}$(Operateur.Off / GrueEnService
∧ ¬ FreinGirationOn
∧ Environnement.LimiteMonteeCharge
∧ Environnement.LimiteChariotArriere
∧ ¬ ModeDegrade
∧ (Charge < PourcentageChargeMax * ChargeMax))

$\mathcal{XK}$(Operateur.Stop / GrueEnService)

$\mathcal{XK}$(Operateur.CderMonteeCharge (v) / GrueEnService
∧ ¬ Environnement.LimiteMonteeCharge
∧ (ModeDegrade ∨ (¬ MomentAtteint ∧ ¬ ChargeAtteinte))
∧ (
(MoteurLevage.Vit = v+1 ∨
(MoteurLevage.Vit = v-1 ∧ FinTempoVitesseSup))
∧ (MoteurLevage.Sens = Haut ∨
(MoteurLevage = <0, Stop> ∧ FinTempoDescente))
))

$\mathcal{XK}$(Operateur.CderDescenteCharge (v) / GrueEnService
∧ ¬ Environnement.LimiteDescenteCharge
∧ (
(MoteurLevage.Vit = v+1 ∨
(MoteurLevage.Vit = v-1 ∧ FinTempoVitesseSup))

$\wedge$ (MoteurLevage.Sens = Bas $\vee$
(MoteurLevage = <0, Stop> $\wedge$ FinTempoMontee))
))

$\mathcal{XK}$(Operateur.CderStopLevage / GrueEnService $\wedge$
(MoteurLevage = Haut $\vee$ MoteurLevage = Bas))

$\mathcal{XK}$(Operateur.CderChariotAvant / GrueEnService $\wedge \neg$ Environnement.LimiteChariotAvant
$\wedge$ (ModeDegrade $\vee$ ($\neg$ MomentAtteint $\wedge \neg$ ChargeAtteinte))
$\wedge$ MoteurChariot = Stop $\wedge$ FinTempoChariotArriere)

$\mathcal{XK}$(Operateur.CderChariotArriere / GrueEnService $\wedge \neg$ Environnement.LimiteChariotArriere
$\wedge$ MoteurChariot = Stop $\wedge$ FinTempoChariotAvant)

$\mathcal{XK}$(Operateur.CderStopChariot / GrueEnService $\wedge$
(MoteurChariot = Avant $\vee$ MoteurChariot = Arriere))

$\mathcal{XK}$(Operateur.CderTranslationAvant / GrueEnService
$\wedge \neg$ Environnement.LimiteTranslationAvant
$\wedge$ (ModeDegrade $\vee$ ($\neg$ MomentAtteint $\wedge \neg$ ChargeAtteinte $\wedge \neg$ VitesseVentAtteinte))
$\wedge$ MoteurTranslation = Stop $\wedge$ FinTempoTranslationArriere)

$\mathcal{XK}$(Operateur.CderTranslationArriere / GrueEnService
$\wedge \neg$ Environnement.LimiteTranslationArriere
$\wedge$ (ModeDegrade $\vee$ ($\neg$ MomentAtteint $\wedge \neg$ ChargeAtteinte $\wedge \neg$ VitesseVentAtteinte))
$\wedge$ MoteurTranslation = Stop $\wedge$ FinTempoTranslationAvant)

$\mathcal{XK}$(Operateur.CderStopTranslation / GrueEnService $\wedge$
(MoteurTranslation = Avant $\vee$ MoteurTranslation = Arriere))

$\mathcal{XK}$(Operateur.CderGirationGauche / GrueEnService $\wedge \neg$ FreinGirationOn
$\wedge$ MoteurGiration = Stop $\wedge$ FinTempoGirationDroite

$\mathcal{XK}$(Operateur.CderGirationDroite / GrueEnService $\wedge \neg$ FreinGirationOn
$\wedge$ MoteurGiration = Stop $\wedge$ FinTempoGirationGauche

$\mathcal{XK}$(Operateur.CderStopGiration / GrueEnService $\wedge$
(MoteurGiration = Droite $\vee$ MoteurGiration = Gauche))

$\mathcal{XK}$(Operateur.EnclencherFreinGiration / GrueEnService
$\wedge \neg$ FreinGirationOn $\wedge$ MoteurGiration = Stop)
$\mathcal{XK}$(Operateur.RelacherFreinGiration / GrueEnService $\wedge$ FreinGirationOn)

$\mathcal{XK}$(Operateur.EnclencherKlaxon / GrueEnService $\wedge \neg$ KlaxonOn)
$\mathcal{XK}$(Operateur.RelacherKlaxon / GrueEnService $\wedge$ KlaxonOn)

$\mathcal{XK}$(Operateur.EnclencherModeDegrade / GrueEnService $\wedge \neg$ ModeDegrade)

State perception

$\mathcal{XK}$(Environnement.LimiteMonteeCharge / GrueEnService)
$\mathcal{XK}$(Environnement.LimiteDescenteCharge / GrueEnService)
$\mathcal{XK}$(Environnement.LimiteChariotAvant / GrueEnService)
$\mathcal{XK}$(Environnement.LimiteChariotArriere / GrueEnService)
$\mathcal{XK}$(Environnement.LimiteTranslationAvant / GrueEnService)
$\mathcal{XK}$(Environnement.LimiteTranslationArriere / GrueEnService)

Action information

$\mathcal{XK}$(EmettreSon.Operateur / GrueEnService)
$\mathcal{XK}$(EmettreSon.Environnement / GrueEnService)

State information

$\mathcal{XK}$(Charge.Operateur / GrueEnService)
$\mathcal{XK}$(Moment.Operateur / GrueEnService)
$\mathcal{XK}$(VitesseVent.Operateur / GrueEnService)

Annexe

3

Programme de l'automate SLC 150

Dans cette annexe,

- la liste des différentes instructions du programme classées par type;
- le code du programme;
- une table de références croisées afin de faciliter une éventuelle recherche.

Commandes diverses

```
| Rung: 001 Variable intermediaire pour la construction de APPUI ONOFF          |
| Rung: 002 Detecteur du front montant de TOUCH ONOFF                           |
| Rung: 003 Mise en service de la grue (premier appui sur [0])                  |
| Rung: 004 Mise hors service de la grue (second appui sur [0])                 |
| Rung: 005 Variable intermediaire pour la construction de APPUI FREIN          |
| Rung: 006 Detecteur du front montant de TOUCH FREIN                           |
| Rung: 007 Enclenchement du frein de giration (premier appui sur [F])          |
| Rung: 008 Declenchement du frein de giration (second appui sur [F])           |
| Rung: 009 Passage en mode degrade (appui sur [Md])                            |
| Rung: 010 Retour en mode normal (detecte suivant la position des elements)    |
| Rung: 011 Activation de l'avertisseur sonore                                  |
```

Mouvement de distribution (translation du chariot)

```
| Rung: 012 Commande d'avance du chariot                                        |
| Rung: 013 Commande de recul du chariot                                        |
| Rung: 014 Sens de rotation du moteur (avance du chariot)                      |
| Rung: 015 Sens de rotation du moteur (recul du chariot)                       |
| Rung: 016 Variable intermediaire pour la construction de ARRET DISTR          |
| Rung: 017 Detecteur du front descendant de -MOT- DISTR                        |
| Rung: 018 Reinitialisation de la temporisation de blocage                     |
| Rung: 019 Blocage consecutif au recul du chariot                              |
| Rung: 020 Blocage consecutif a l'avance du chariot                            |
| Rung: 021 Incrementation de la temporisation de blocage                       |
```

Mouvement de translation de la grue

```
| Rung: 022 Commande d'avance de la grue                                        |
| Rung: 023 Commande de recul de la grue                                        |
| Rung: 024 Sens de rotation du moteur (avance de la grue)                      |
| Rung: 025 Sens de rotation du moteur (recul de la grue)                       |
| Rung: 026 Variable intermediaire pour la construction de ARRET TRANS          |
| Rung: 027 Detecteur du front descendant de -MOT- TRANS                        |
| Rung: 028 Reinitialisation de la temporisation de blocage                     |
| Rung: 029 Blocage consecutif au recul de la grue                              |
| Rung: 030 Blocage consecutif a l'avance de la grue                            |
| Rung: 031 Incrementation de la temporisation de blocage                       |
```

Mouvement de giration de la flèche

```
| Rung: 032 Commande de giration de la fleche vers la droite                    |
| Rung: 033 Commande de giration de la fleche vers la gauche                    |
| Rung: 034 Sens de rotation du moteur (giration vers la droite)                |
| Rung: 035 Sens de rotation du moteur (giration vers la gauche)                |
| Rung: 036 Variable intermediaire pour la construction de ARRET GIRAT          |
| Rung: 037 Detecteur du front descendant de -MOT- GIRAT                        |
| Rung: 038 Reinitialisation de la temporisation de blocage                     |
| Rung: 039 Blocage consecutif a la giration de la fleche vers la gauche        |
| Rung: 040 Blocage consecutif a la giration de la fleche vers la droite        |
| Rung: 041 Incrementation de la temporisation de blocage                       |
```

Mouvement de levage du crochet

```
| Rung: 042 Commande de montee du crochet                                    |
| Rung: 043 Commande de descente du crochet                                  |
| Rung: 044 Sens de rotation du moteur (montee du crochet)                   |
| Rung: 045 Sens de rotation du moteur (descente du crochet)                 |
| Rung: 046 Selection de la sortie moteur (premiere vitesse)                 |
| Rung: 047 Selection de la sortie moteur (deuxieme vitesse)                 |
| Rung: 048 Selection de la sortie moteur (troisieme vitesse)                |
| Rung: 049 Desactivation de la troisieme vitesse en cas d'arret             |
| Rung: 050 Desactivation de la deuxieme vitesse en cas d'arret              |
| Rung: 051 Passage de la premiere vitesse a la deuxieme                     |
| Rung: 052 Passage de la deuxieme vitesse a la troisieme                    |
| Rung: 053 Passage de la troisieme vitesse a la deuxieme                    |
| Rung: 054 Passage de la deuxieme vitesse a la premiere                     |

| Rung: 055 Variable intermediaire pour la construction de DEMAR VIT 1      |
| Rung: 056 Detecteur du front montant de -MOT- VIT 1                        |
| Rung: 057 Reinitialisation de la temporisation de blocage                  |
| Rung: 058 Blocage consecutif au demarrage du moteur (premiere vitesse)     |
| Rung: 059 Incrementation de la temporisation de blocage                    |

| Rung: 060 Variable intermediaire pour la construction de DEMAR VIT 2      |
| Rung: 061 Detecteur du front montant de -MOT- VIT 2                        |
| Rung: 062 Reinitialisation de la temporisation de blocage                  |
| Rung: 063 Blocage consecutif au demarrage de la deuxieme vitesse           |
| Rung: 064 Incrementation de la temporisation de blocage                    |

| Rung: 065 Variable intermediaire pour la construction de ARRET VIT 2      |
| Rung: 066 Detecteur du front descendant de -MOT- VIT 2                     |
| Rung: 067 Variable intermediaire pour la construction de ARRET CHUT3      |
| Rung: 068 Detecteur du front descendant de VIT 3 CHUTE                     |
| Rung: 069 Reinitialisation de la temporisation d'attente                   |
| Rung: 070 Blocage consecutif a la diminution de vitesse 2->1               |
| Rung: 071 Incrementation de la temporisation d'attente                     |

| Rung: 072 Variable intermediaire pour la construction de ARRET VIT 3      |
| Rung: 073 Detecteur du front descendant de -MOT- VIT 3                     |
| Rung: 074 Reinitialisation de la temporisation d'attente                   |
| Rung: 075 Blocage consecutif a la diminution de vitesse 3->2               |
| Rung: 076 Incrementation de la temporisation d'attente                     |

| Rung: 077 Variable intermediaire pour la construction de ARRET VIT 1      |
| Rung: 078 Detecteur du front descendant de -MOT- VIT 1                     |
| Rung: 079 Variable intermediaire pour la construction de ARRET CHUT2      |
| Rung: 080 Detecteur du front descendant de VIT 2 CHUTE                     |
| Rung: 081 Reinitialisation de la temporisation de blocage                  |
| Rung: 082 Blocage consecutif a la descente du crochet                      |
| Rung: 083 Blocage consecutif a la montee du crochet                        |
| Rung: 084 Incrementation de la temporisation                               |
```

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 1

Rung: 001 Variable intermediaire pour la construction de APPUI ONOFF

```
TOUCH                                                                AUXIL
ONOFF                                                                ONOFF
 001                                                                  801
++-]\[-------+-------------------------------------------------------(   )----+
||           |
||APPUI AUXIL|
||ONOFF ONOFF|
|| 802   801 |
|+-]\[---] [-+
```

Rung: 002 Detecteur du front montant de TOUCH ONOFF

```
TOUCH AUXIL                                                          APPUI
ONOFF ONOFF                                                          ONOFF
 001   801                                                           802
+--] [---] [---------------------------------------------------------(   )----+
```

Rung: 003 Mise en service de la grue (premier appui sur [0])

```
APPUI GR EN                                                          GR EN
ONOFF SERVI                                                          SERVI
 802   011                                                           011
+--] [---]\[---------------------------------------------------------( L )----+
```

Rung: 004 Mise hors service de la grue (second appui sur [0])

```
APPUI FREIN CHAR  CROCH MODE  CHARG GR EN                             GR EN
ONOFF GIRAT SECU  HAUT  DEGRA < 20% SERVI                            SERVI
 802   116   105   102   800   202   011                             011
+--] [---]\[---] [---] [---]\[---] [---] [---------------------------( U )----+
```

Rung: 005 Variable intermediaire pour la construction de APPUI FREIN

```
TOUCH                                                                AUXIL
FREIN                                                                FREIN
 010                                                                  803
++-]\[-------+-------------------------------------------------------(   )----+
||           |
||APPUI AUXIL|
||FREIN FREIN|
|| 804   803 |
|+-]\[---] [-+
```

Rung: 006 Detecteur du front montant de TOUCH FREIN

```
TOUCH AUXIL                                                          APPUI
FREIN FREIN                                                          FREIN
 010   803                                                           804
+--] [---] [---------------------------------------------------------(   )----+
```

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 2

Rung: 007 Enclenchement du frein de giration (premier appui sur [F])

```
APPUI FREIN -MOT- GR EN                                          FREIN
FREIN GIRAT GIRAT SERVI                                          GIRAT
 804   116   114   011                                          116
--] [---]\[---]\[---] [-----------------------------------------( L )----
```

Rung: 008 Declenchement du frein de giration (second appui sur [F])

```
APPUI FREIN GR EN                                                FREIN
FREIN GIRAT SERVI                                                GIRAT
 804   116   011                                                 116
--] [---] [---] [-----------------------------------------------( U )----
```

Rung: 009 Passage en mode degrade (appui sur [Md])

```
TOUCH MODE  GR EN                                                MODE
DEGRA DEGRA SERVI                                                DEGRA
 101   800   011                                                 800
--] [---]\[---] [-----------------------------------------------( L )----
```

Rung: 010 Retour en mode normal (detecte suivant la position des elements)

```
TOUCH MODE  CHARG MOMEN CROCH CHAR  GR EN                        MODE
DEGRA DEGRA > MAX > MAX  BAS  EXTRM SERVI                        DEGRA
 101   800   201   203   103   104   011                       800
--]\[---] [---]\[---]\[---]\[---]\[---] [---------------------( U )----
```

Rung: 011 Activation de l'avertisseur sonore

```
MODE  GR EN                                                      AVERT
DEGRA SERVI                                                      SONOR
 800   011                                                       211
++-] [-+-] [-----------------------------------------------------(   )----
||     |
||ANEMO|
||V>MAX|
|| 204 |
|+-] [-+
||     |
||TOUCH|
||KLAX |
|| 110 |
|+-] [-+
```

Rung: 012 Commande d'avance du chariot

```
TOUCH TOUCH CHAR  MODE  CHARG MOMEN <- CH GR EN                  -MOT-
CH -> <- CH EXTRM DEGRA > MAX > MAX BLOQU SERVI                  DISTR
 004   005   104   800   201   203   714   011                 016
--] [---]\[---]\[-+-]\[---]\[---]\[-+-]\[---] [-----------------(   )----
                  |                   |
                  |MODE               |
                  |DEGRA              |
                  | 800               |
                  +-] [---------------+
```

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 3

Rung: 013 Commande de recul du chariot

```
TOUCH TOUCH CHAR  CH -> GR EN                                    -MOT-
<- CH CH -> SECU  BLOQU SERVI                                    DISTR
 005   004   105   724   011                                      016
+--] [---]\[---]\[---]\[---] [-----------------------------------(   )----+
```

Rung: 014 Sens de rotation du moteur (avance du chariot)

```
TOUCH TOUCH                                                      SENS
CH -> <- CH                                                      DISTR
 004   005                                                       111
+--] [---]\[-----------------------------------------------------( L )----+
```

Rung: 015 Sens de rotation du moteur (recul du chariot)

```
TOUCH TOUCH                                                      SENS
<- CH CH ->                                                      DISTR
 005   004                                                       111
+--] [---]\[-----------------------------------------------------( U )----+
```

Rung: 016 Variable intermediaire pour la construction de ARRET DISTR

```
-MOT-                                                            AUXIL
DISTR                                                            DISTR
 016                                                             703
++-] [-------+-------------------------------------------------(   )----+
||           |
||ARRET AUXIL|
||DISTR DISTR|
|| 704   703 |
|+-]\[---] [-+
```

Rung: 017 Detecteur du front descendant de -MOT- DISTR

```
-MOT- AUXIL                                                      ARRET
DISTR DISTR                                                      DISTR
 016   703                                                       704
+--]\[---] [-----------------------------------------------------(   )----+
```

Rung: 018 Reinitialisation de la temporisation de blocage

```
ARRET                                                            TEMPO
DISTR                                                            DISTR
 704                                                             904
+--] [-----------------------------------------------------------(RST)----+
                                                                RE 0000
```

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 4

Rung: 019 Blocage consecutif au recul du chariot

ARRET DISTR 704	SENS DISTR 111	TEMPO DISTR 904	<- CH BLOQU 714

<- CH BLOQU 714

Rung: 020 Blocage consecutif a l'avance du chariot

ARRET DISTR 704	SENS DISTR 111	TEMPO DISTR 904	CH -> BLOQU 724

CH -> BLOQU 724

Rung: 021 Incrementation de la temporisation de blocage

<- CH BLOQU 714	TEMPO DISTR 904

CH -> BLOQU 724

PR 0030

Rung: 022 Commande d'avance de la grue

TOUCH GR -> 006	TOUCH <- GR 007	GRUE LIMAV 106	MODE DEGRA 800	ANEMO V>MAX 204	CHARG > MAX 201	MOMEN > MAX 203	<- GR BLOQU 716	GR EN SERVI 011	-MOT- TRANS 112

MODE DEGRA 800

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 5

Rung: 023 Commande de recul de la grue

TOUCH <- GR 007	TOUCH GR -> 006	GRUE LIMAR 107	MODE DEGRA 800	ANEMO V>MAX 204	CHARG > MAX 201	MOMEN > MAX 203	GR -> BLOQU 726	GR EN SERVI 011	-MOT- TRANS 112
] [	]\[	]\[	-+-]\[	]\[	]\[	]\[	-+-]\[	] [	( )
			|MODE DEGRA 800						

Rung: 024 Sens de rotation du moteur (avance de la grue)

TOUCH GR -> 006	TOUCH <- GR 007	SENS TRANS 113
] [	]\[	(L)

Rung: 025 Sens de rotation du moteur (recul de la grue)

TOUCH <- GR 007	TOUCH GR -> 006	SENS TRANS 113
] [	]\[	(U)

Rung: 026 Variable intermediaire pour la construction de ARRET TRANS

-MOT- TRANS 112	AUXIL TRANS 705
] [	( )

ARRET TRANS 706	AUXIL TRANS 705
]\[	] [

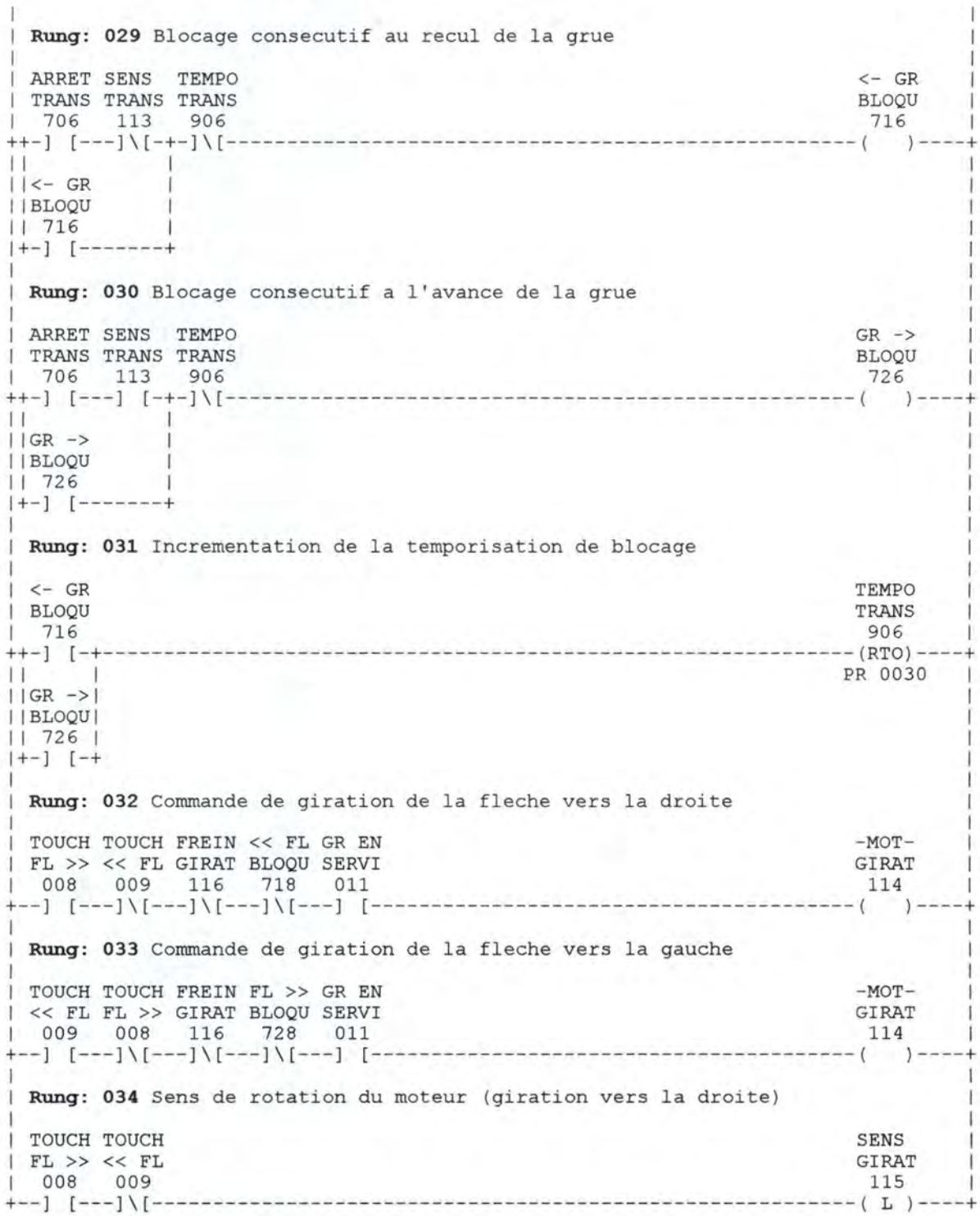
Rung: 027 Detecteur du front descendant de -MOT- TRANS

-MOT- TRANS 112	AUXIL TRANS 705	ARRET TRANS 706
]\[	] [	( )

Rung: 028 Reinitialisation de la temporisation de blocage

ARRET TRANS 706	TEMPO TRANS 906
] [	(RST)
	RE 0000

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 6



Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 7

Rung: 035 Sens de rotation du moteur (giration vers la gauche)

```
TOUCH TOUCH                                                    SENS
<< FL FL >>                                                    GIRAT
 009   008                                                      115
--] [---]\[-----------------------------------------------------( U )----
```

Rung: 036 Variable intermediaire pour la construction de ARRET GIRAT

```
-MOT-                                                          AUXIL
GIRAT                                                          GIRAT
 114                                                           707
--] [-------+-----------------------------------------------------(   )----
            |
ARRET AUXIL|
GIRAT GIRAT|
 708   707 |
]\[---] [-+
```

Rung: 037 Detecteur du front descendant de -MOT- GIRAT

```
-MOT- AUXIL                                                    ARRET
GIRAT GIRAT                                                    GIRAT
 114   707                                                     708
--]\[---] [-----------------------------------------------------(   )----
```

Rung: 038 Reinitialisation de la temporisation de blocage

```
ARRET                                                          TEMPO
GIRAT                                                          GIRAT
 708                                                           908
--] [-------------------------------------------------------------(RST)----
                                                              RE 0000
```

Rung: 039 Blocage consecutif a la giration de la fleche vers la gauche

```
ARRET SENS  TEMPO                                              << FL
GIRAT GIRAT GIRAT                                              BLOQU
 708   115   908                                               718
--] [---]\[-+-]\[-----------------------------------------------(   )----
            |
<< FL       |
BLOQU       |
 718        |
-] [-------+
```

Rung: 040 Blocage consecutif a la giration de la fleche vers la droite

```
ARRET SENS  TEMPO                                              FL >>
GIRAT GIRAT GIRAT                                              BLOQU
 708   115   908                                               728
--] [---] [-+-]\[-----------------------------------------------(   )----
            |
FL >>       |
BLOQU       |
 728        |
-] [-------+
```

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 8

Rung: 041 Incrementation de la temporisation de blocage

```
| << FL                                                                  TEMPO |
| BLOQU                                                                  GIRAT |
|  718                                                                    908 |
++-] [-+-------------------------------------------------------------(RTO)----+
||     |                                                              PR 0030   |
||FL >>|                                                                       |
||BLOQU|                                                                       |
|| 728 |                                                                       |
|+-] [-+                                                                       |
```

Rung: 042 Commande de montee du crochet

```
| TOUCH TOUCH CROCH MODE  CHARG MOMEN CR \/ GR EN                        -MOT- |
| CR /\ CR \/ HAUT  DEGRA > MAX > MAX BLOQU SERVI                        LEVAG |
|  002   003   102   800   201   203   712   011                         730   |
+--] [---]\[---]\[-+-]\[---]\[---]\[-+-]\[---] [----------------------(   )----+
|                   |                 |                                        |
|                   |MODE             |                                        |
|                   |DEGRA            |                                        |
|                   | 800             |                                        |
|                   +-] [-------------+                                        |
```

Rung: 043 Commande de descente du crochet

```
| TOUCH TOUCH CROCH CR /\ GR EN                                          -MOT- |
| CR \/ CR /\  BAS  BLOQU SERVI                                          LEVAG |
|  003   002   103   722   011                                            730   |
+--] [---]\[---]\[---]\[---] [-----------------------------------------(   )----+
```

Rung: 044 Sens de rotation du moteur (montee du crochet)

```
| TOUCH TOUCH VIT 2 VIT 3                                                SENS  |
| CR /\ CR \/ CHUTE CHUTE                                                LEVAG |
|  002   003   755   757                                                  015   |
+--] [---]\[---]\[---]\[-----------------------------------------------( L )----+
```

Rung: 045 Sens de rotation du moteur (descente du crochet)

```
| TOUCH TOUCH VIT 2 VIT 3                                                SENS  |
| CR \/ CR /\ CHUTE CHUTE                                                LEVAG |
|  003   002   755   757                                                  015   |
+--] [---]\[---]\[---]\[-----------------------------------------------( U )----+
```

Rung: 046 Selection de la sortie moteur (premiere vitesse)

```
| -MOT- LEVAG LEVAG VIT 2 VIT 3                                          -MOT- |
| LEVAG BIT 2 BIT 1 CHUTE CHUTE                                          VIT 1 |
|  730   731   732   755   757                                            012   |
+--] [---]\[---]\[---]\[---]\[-----------------------------------------(   )----+
```

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 9

Rung: 047 Selection de la sortie moteur (deuxieme vitesse)

```
-MOT- LEVAG LEVAG VIT 3                                          -MOT-
LEVAG BIT 2 BIT 1 CHUTE                                          VIT 2
 730   731   732   757                                           013
+--] [---]\[---] [---]\[-------------------------------------------(   )----+
```

Rung: 048 Selection de la sortie moteur (troisieme vitesse)

```
-MOT- LEVAG LEVAG                                                -MOT-
LEVAG BIT 2 BIT 1                                                VIT 3
 730   731   732                                                 014
+--] [---] [---] [-------------------------------------------------(   )----+
```

Rung: 049 Desactivation de la troisieme vitesse en cas d'arret

```
-MOT- LEVAG                                                      LEVAG
LEVAG BIT 2                                                      BIT 2
 730   731                                                       731
+--]\[---] [-----------------------------------------------------( U )----+
```

Rung: 050 Desactivation de la deuxieme vitesse en cas d'arret

```
-MOT- LEVAG                                                      LEVAG
LEVAG BIT 1                                                      BIT 1
 730   732                                                       732
+--]\[---] [-----------------------------------------------------( U )----+
```

Rung: 051 Passage de la premiere vitesse a la deuxieme

```
TOUCH TOUCH -MOT- LEVAG LEVAG VIT 1                              LEVAG
VIT + VIT - VIT 1 BIT 2 BIT 1 BLOQU                              BIT 1
 108   109   012   731   732   751                               732
+--] [---]\[---] [---]\[---]\[---]\[-------------------------------( L )----+
```

Rung: 052 Passage de la deuxieme vitesse a la troisieme

```
TOUCH TOUCH -MOT- LEVAG LEVAG VIT 2                              LEVAG
VIT + VIT - VIT 2 BIT 2 BIT 1 BLOQU                              BIT 2
 108   109   013   731   732   753                               731
+--] [---]\[---] [---]\[---] [---]\[-------------------------------( L )----+
```

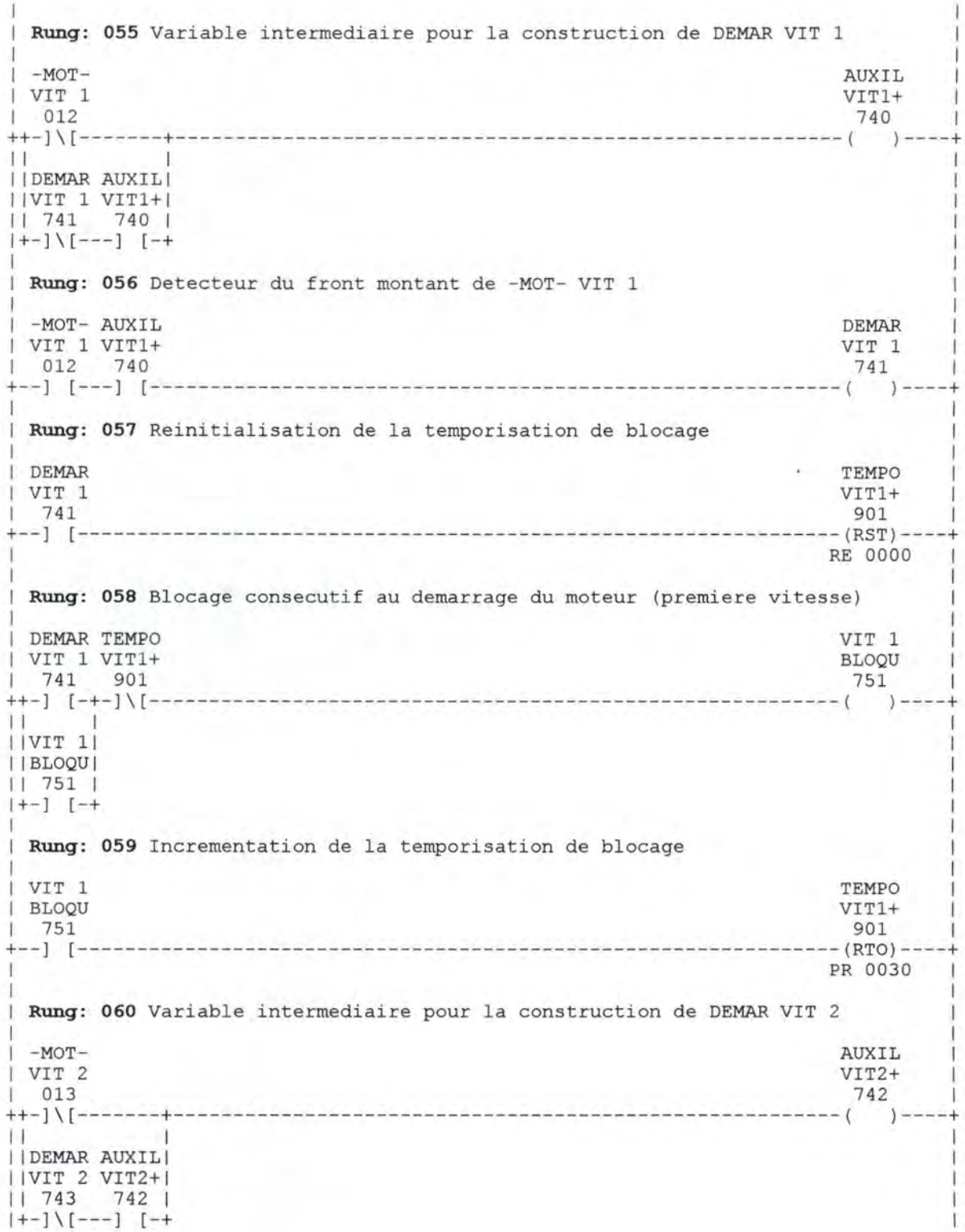
Rung: 053 Passage de la troisieme vitesse a la deuxieme

```
TOUCH TOUCH -MOT- LEVAG LEVAG                                    LEVAG
VIT - VIT + VIT 3 BIT 2 BIT 1                                    BIT 2
 109   108   014   731   732                                     731
+--] [---]\[---] [---] [---] [-------------------------------------( U )----+
```

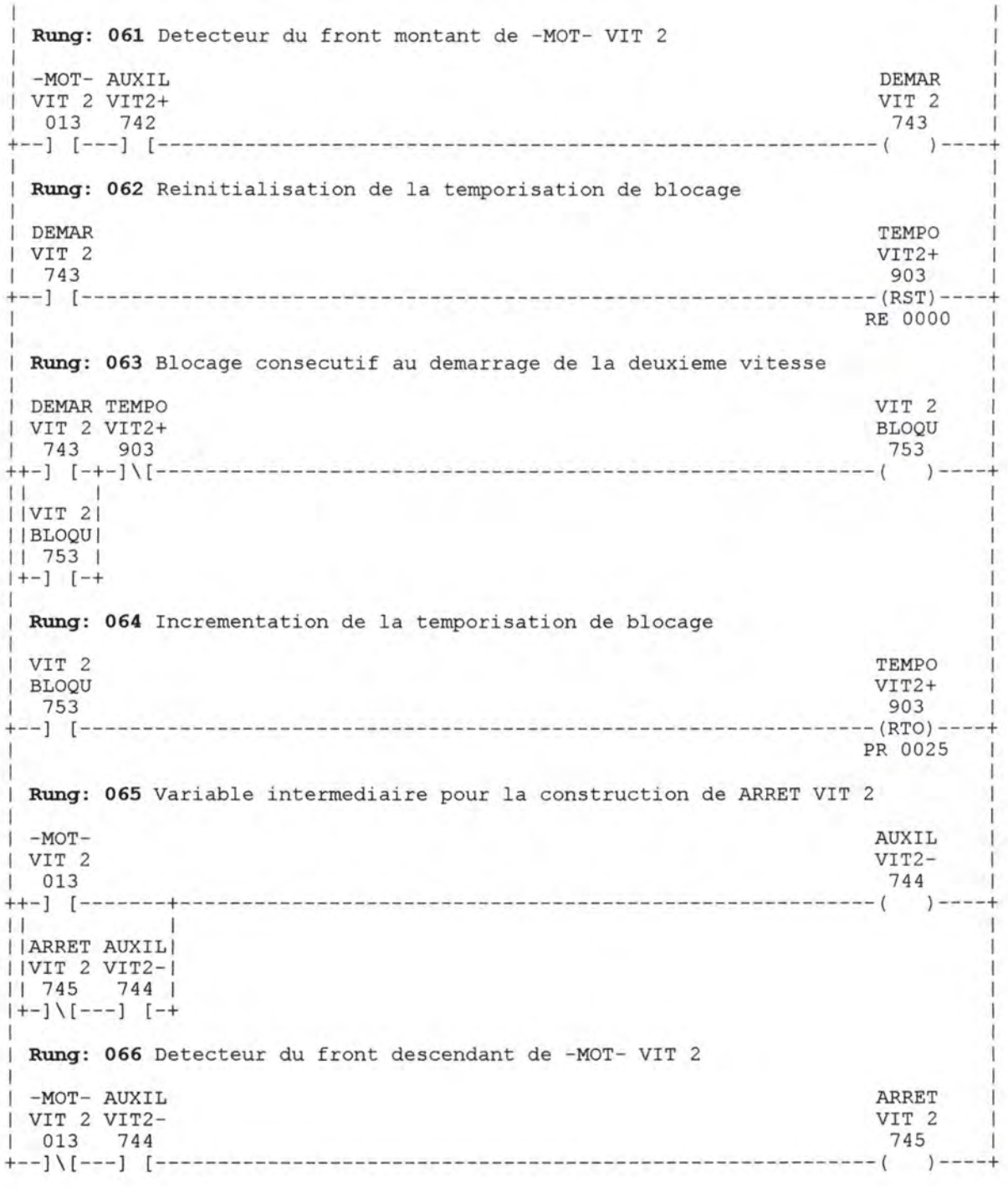
Rung: 054 Passage de la deuxieme vitesse a la premiere

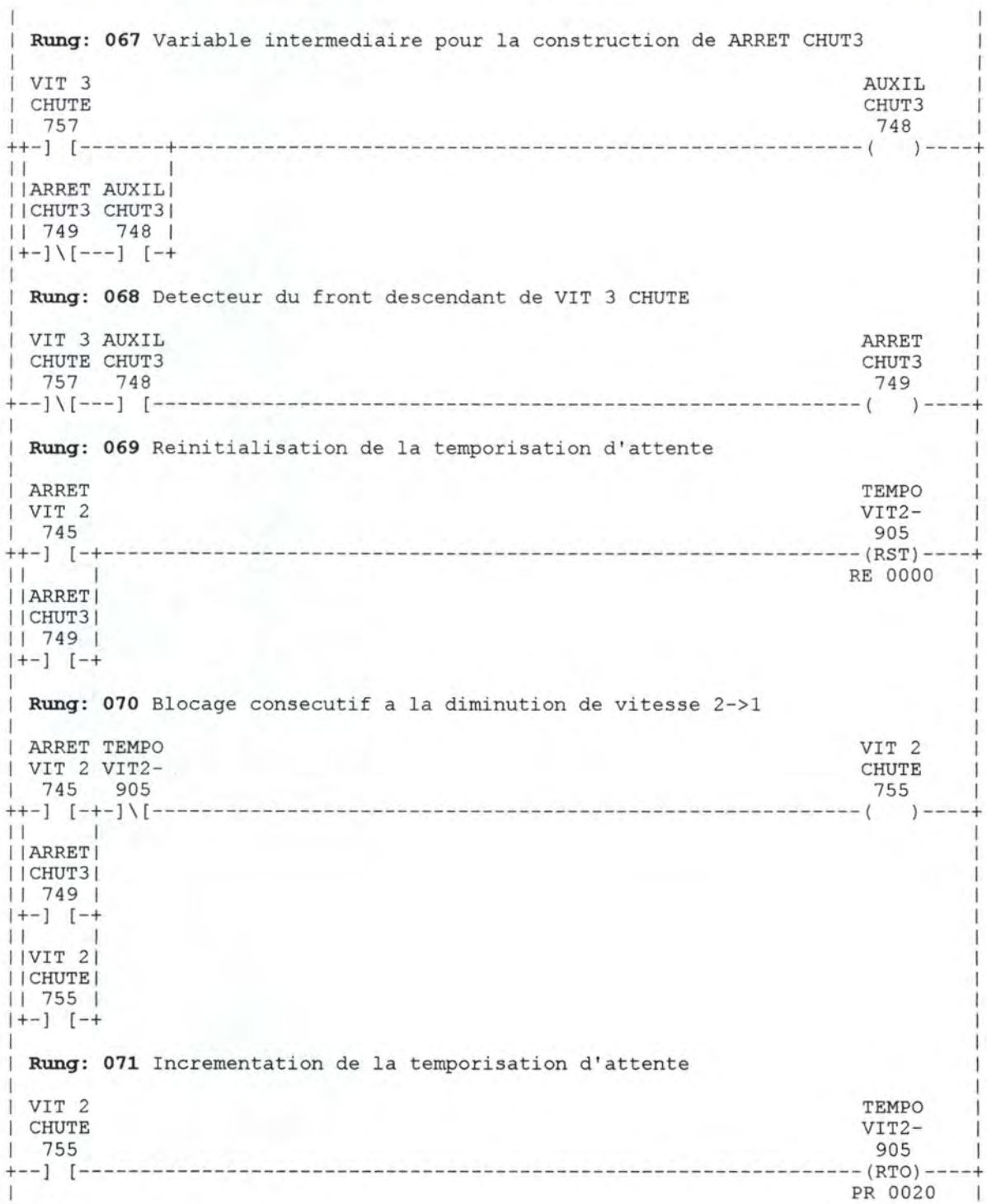
```
TOUCH TOUCH -MOT- LEVAG LEVAG VIT 3                              LEVAG
VIT - VIT + VIT 2 BIT 2 BIT 1 CHUTE                              BIT 1
 109   108   013   731   732   757                               732
+--] [---]\[---] [---]\[---] [---]\[-------------------------------( U )----+
```

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 10



Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 11




Rung: 067 Variable intermediaire pour la construction de ARRET CHUT3
VIT 3 CHUTE 757
AUXIL CHUT3 748
ARRET CHUT3 749
AUXIL CHUT3 748
Rung: 068 Detecteur du front descendant de VIT 3 CHUTE
VIT 3 CHUTE 757
AUXIL CHUT3 748
ARRET CHUT3 749
Rung: 069 Reinitialisation de la temporisation d'attente
ARRET 749
ARRET VIT 2 745
TEMPO VIT2- 905
RE 0000
Rung: 070 Blocage consecutif a la diminution de vitesse 2->1
ARRET 749
TEMPO VIT2- 905
ARRET VIT 2 745
VIT 2 CHUTE 755
Rung: 071 Incrementation de la temporisation d'attente
VIT 2 CHUTE 755
TEMPO VIT2- 905
PR 0020

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 13

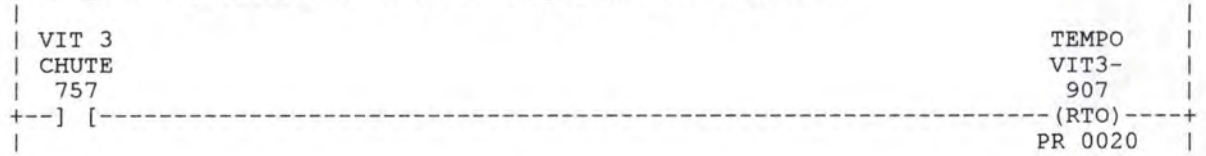
Rung: 072 Variable intermediaire pour la construction de ARRET VIT 3

Rung: 073 Detecteur du front descendant de -MOT- VIT 3

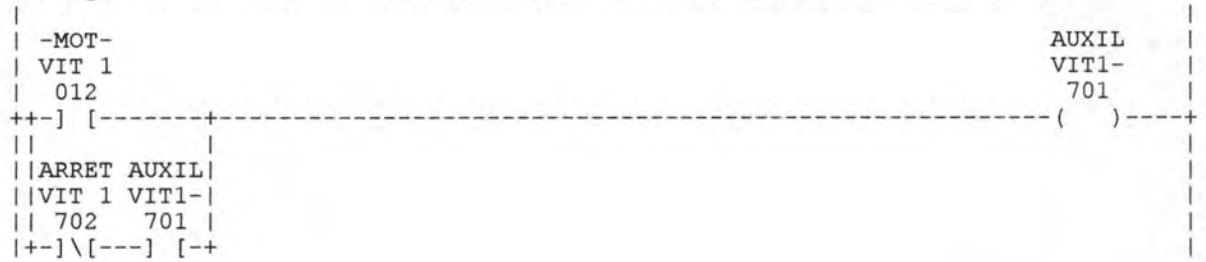
Rung: 074 Reinitialisation de la temporisation d'attente

Rung: 075 Blocage consecutif a la diminution de vitesse 3->2

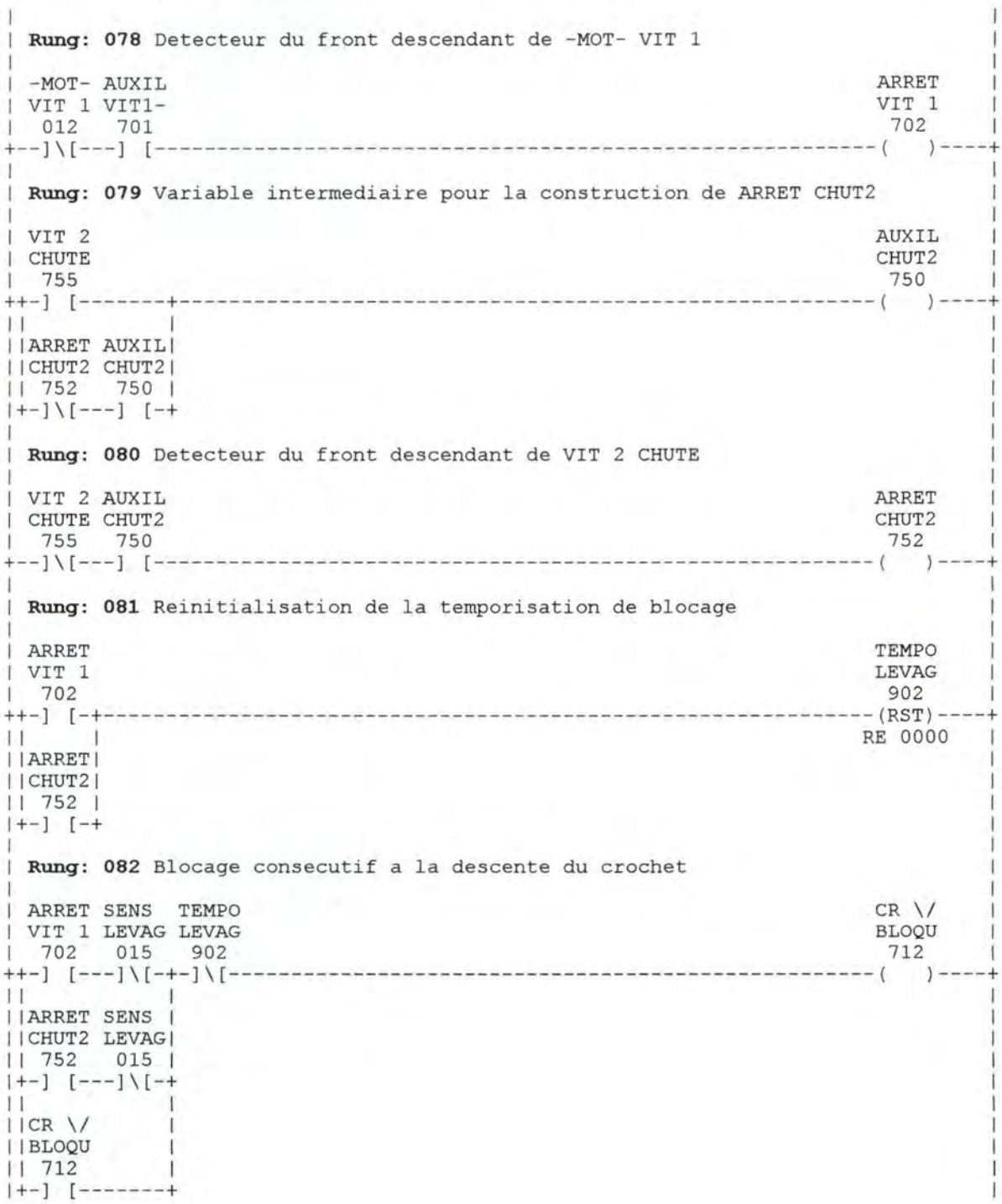
Rung: 076 Incrementation de la temporisation d'attente



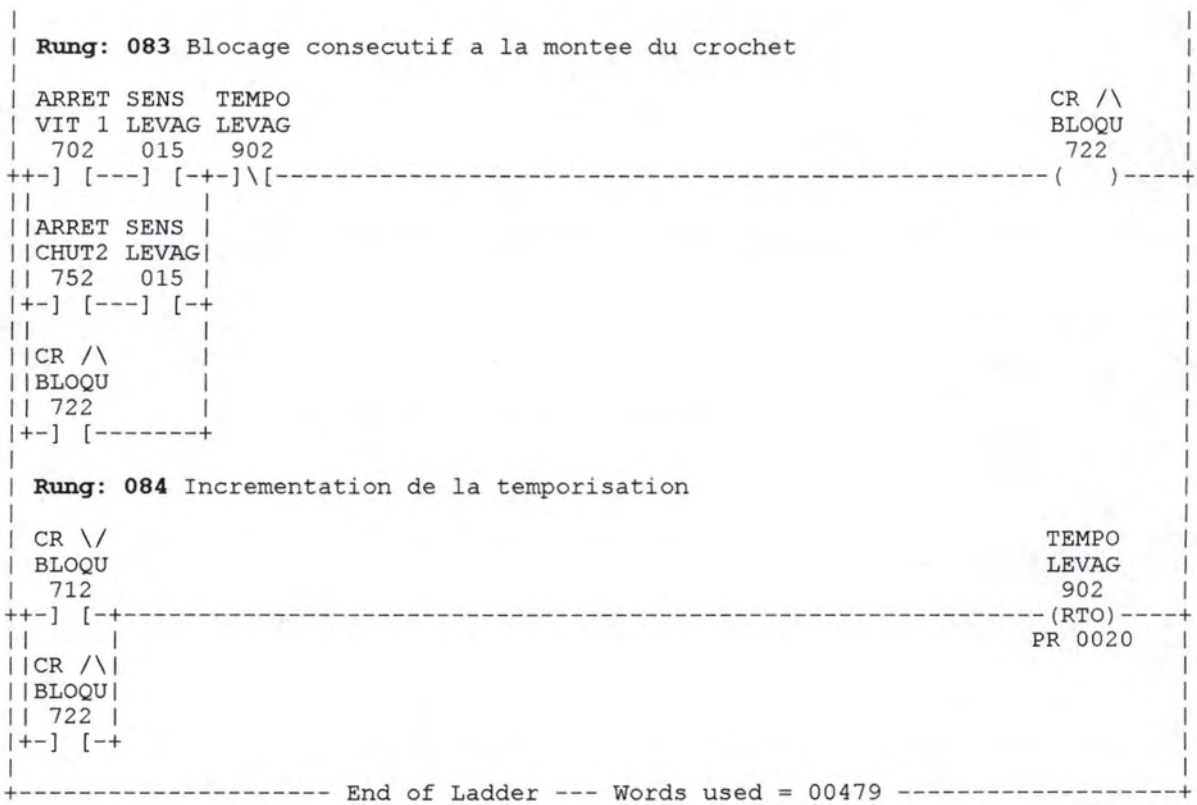
Rung: 077 Variable intermediaire pour la construction de ARRET VIT 1



Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 14



Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Ladder Diagram Page 15



Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Cross Reference Page 16

INPUT

Address	Element	Rung Number(s)	Instruction Comment
001	-] [-	002	TOUCH ONOFF: Bouton-poussoir commandant la mise en/hors service de la grue
001	-]\[-	001	TOUCH ONOFF: Bouton-poussoir commandant la mise en/hors service de la grue
002	-] [-	042, 044	TOUCH CR /\: Bouton-poussoir commandant la montee du crochet
002	-]\[-	043, 045	TOUCH CR /\: Bouton-poussoir commandant la montee du crochet
003	-] [-	043, 045	TOUCH CR \/: Bouton-poussoir commandant la descente du crochet
003	-]\[-	042, 044	TOUCH CR \/: Bouton-poussoir commandant la descente du crochet
004	-] [-	012, 014	TOUCH CH ->: Bouton-poussoir commandant l'avance du chariot
004	-]\[-	013, 015	TOUCH CH ->: Bouton-poussoir commandant l'avance du chariot
005	-] [-	013, 015	TOUCH <- CH: Bouton-poussoir commandant le recul du chariot
005	-]\[-	012, 014	TOUCH <- CH: Bouton-poussoir commandant le recul du chariot
006	-] [-	022, 024	TOUCH GR ->: Bouton-poussoir commandant l'avance de la grue
006	-]\[-	023, 025	TOUCH GR ->: Bouton-poussoir commandant l'avance de la grue
007	-] [-	023, 025	TOUCH <- GR: Bouton-poussoir commandant le recul de la grue
007	-]\[-	022, 024	TOUCH <- GR: Bouton-poussoir commandant le recul de la grue
008	-] [-	032, 034	TOUCH FL >>: Bouton-poussoir commandant la rotation de la fleche vers la droite
008	-]\[-	033, 035	TOUCH FL >>: Bouton-poussoir commandant la rotation de la fleche vers la droite
009	-] [-	033, 035	TOUCH << FL: Bouton-poussoir commandant la rotation de la fleche vers la gauche
009	-]\[-	032, 034	TOUCH << FL: Bouton-poussoir commandant la rotation de la fleche vers la gauche
010	-] [-	006	TOUCH FREIN: Bouton-poussoir commandant l'enclenchement/declenchement du frein
010	-]\[-	005	TOUCH FREIN: Bouton-poussoir commandant l'enclenchement/declenchement du frein
101	-] [-	009	TOUCH DEGRA: Bouton-poussoir destine au passage en mode degrade
101	-]\[-	010	TOUCH DEGRA: Bouton-poussoir destine au passage en mode degrade
102	-] [-	004	CROCH HAUT : Capteur indiquant que le crochet a atteint la limite superieure
102	-]\[-	042	CROCH HAUT : Capteur indiquant que le crochet a atteint la limite superieure
103	-]\[-	010, 043	CROCH BAS : Capteur indiquant que le crochet a atteint la limite inferieure
104	-]\[-	010, 012	CHAR EXTRM: Capteur indiquant la pos. du chariot a l'extremite de la fleche

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Cross Reference Page 17

INPUT

Address	Element	Rung Number(s)	Instruction Comment
105	-] [-	004	CHAR SECU : Capteur indiquant la pos. du chariot au pied du mat
105	-]\[-	013	CHAR SECU : Capteur indiquant la pos. du chariot au pied du mat
106	-]\[-	022	GRUE LIMAV: Capteur indiquant que la grue a atteint l'extr. avant du rail
107	-]\[-	023	GRUE LIMAR: Capteur indiquant que la grue a atteint l'extr. arriere du rail
108	-] [-	051, 052	TOUCH VIT +: Bouton-poussoir destine a augmenter la vitesse du levage
108	-]\[-	053, 054	TOUCH VIT +: Bouton-poussoir destine a augmenter la vitesse du levage
109	-] [-	053, 054	TOUCH VIT -: Bouton-poussoir destine a diminuer la vitesse du levage
109	-]\[-	051, 052	TOUCH VIT -: Bouton-poussoir destine a diminuer la vitesse du levage
110	-] [-	011	TOUCH KLAX : Bouton-poussoir commandant l'avertisseur sonore
201	-]\[-	010, 012, 022, 023, 042	CHARG > MAX: Capteur indiquant que la charge depasse le maximum admissible
202	-] [-	004	CHARG < 20%: Capteur indiquant que la charge est inferieure a 20% du maximum
203	-]\[-	010, 012, 022, 023, 042	MOMEN > MAX: Capteur indiquant que le moment depasse le maximum admissible
204	-] [-	011	ANEMO V>MAX: Capteur indiquant que la vitesse du vent depasse le seuil max.
204	-]\[-	022, 023	ANEMO V>MAX: Capteur indiquant que la vitesse du vent depasse le seuil max.

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Cross Reference Page 18

OUTPUT

Address	Element	Rung Number(s)	Instruction Comment
011	-] [-	004, 007, 008, 009, 010, 011, 012, 013, 022, 023, 032, 033, 042, 043	GR EN SERVI: Sortie entrainant la mise sous tension des organes de la machine
011	-]\[-	003	GR EN SERVI: Sortie entrainant la mise sous tension des organes de la machine
011	-(L)-	003	GR EN SERVI: Sortie entrainant la mise sous tension des organes de la machine
011	-(U)-	004	GR EN SERVI: Sortie entrainant la mise sous tension des organes de la machine
012	-] [-	051, 056, 077	-MOT- VIT 1: Sortie alimentant le moteur de levage en premiere vitesse
012	-]\[-	055, 078	-MOT- VIT 1: Sortie alimentant le moteur de levage en premiere vitesse
012	-( )-	046	-MOT- VIT 1: Sortie alimentant le moteur de levage en premiere vitesse
013	-] [-	052, 054, 061, 065	-MOT- VIT 2: Sortie alimentant le moteur de levage en deuxieme vitesse
013	-]\[-	060, 066	-MOT- VIT 2: Sortie alimentant le moteur de levage en deuxieme vitesse
013	-( )-	047	-MOT- VIT 2: Sortie alimentant le moteur de levage en deuxieme vitesse
014	-] [-	053, 072	-MOT- VIT 3: Sortie alimentant le moteur de levage en troisieme vitesse
014	-]\[-	073	-MOT- VIT 3: Sortie alimentant le moteur de levage en troisieme vitesse
014	-( )-	048	-MOT- VIT 3: Sortie alimentant le moteur de levage en troisieme vitesse
015	-] [-	083	SENS LEVAG: Sortie indiquant le sens de rotation au moteur de levage
015	-]\[-	082	SENS LEVAG: Sortie indiquant le sens de rotation au moteur de levage
015	-(L)-	044	SENS LEVAG: Sortie indiquant le sens de rotation au moteur de levage
015	-(U)-	045	SENS LEVAG: Sortie indiquant le sens de rotation au moteur de levage
016	-] [-	016	-MOT- DISTR: Sortie alimentant le moteur responsable de la distribution
016	-]\[-	017	-MOT- DISTR: Sortie alimentant le moteur responsable de la distribution
016	-( )-	012, 013	-MOT- DISTR: Sortie alimentant le moteur responsable de la distribution
111	-] [-	020	SENS DISTR: Sortie indiquant le sens de rotation au moteur de distribution
111	-]\[-	019	SENS DISTR: Sortie indiquant le sens de rotation au moteur de distribution
111	-(L)-	014	SENS DISTR: Sortie indiquant le sens de rotation au moteur de distribution
111	-(U)-	015	SENS DISTR: Sortie indiquant le sens de rotation au moteur de distribution
112	-] [-	026	-MOT- TRANS: Sortie alimentant le moteur responsable de la translation

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Cross Reference Page 19

OUTPUT

Address	Element	Rung Number(s)	Instruction Comment
112	-]\[-	027	-MOT- TRANS: Sortie alimentant le moteur responsable de la translation
112	-( )-	022, 023	-MOT- TRANS: Sortie alimentant le moteur responsable de la translation
113	-] [-	030	SENS TRANS: Sortie indiquant le sens de rotation au moteur de translation
113	-]\[-	029	SENS TRANS: Sortie indiquant le sens de rotation au moteur de translation
113	-(L)-	024	SENS TRANS: Sortie indiquant le sens de rotation au moteur de translation
113	-(U)-	025	SENS TRANS: Sortie indiquant le sens de rotation au moteur de translation
114	-] [-	036	-MOT- GIRAT: Sortie alimentant le moteur responsable de la giration
114	-]\[-	007, 037	-MOT- GIRAT: Sortie alimentant le moteur responsable de la giration
114	-( )-	032, 033	-MOT- GIRAT: Sortie alimentant le moteur responsable de la giration
115	-] [-	040	SENS GIRAT: Sortie indiquant le sens de rotation au moteur de giration
115	-]\[-	039	SENS GIRAT: Sortie indiquant le sens de rotation au moteur de giration
115	-(L)-	034	SENS GIRAT: Sortie indiquant le sens de rotation au moteur de giration
115	-(U)-	035	SENS GIRAT: Sortie indiquant le sens de rotation au moteur de giration
116	-] [-	008	FREIN GIRAT: Sortie correspondant au frein de giration
116	-]\[-	004, 007, 032, 033	FREIN GIRAT: Sortie correspondant au frein de giration
116	-(L)-	007	FREIN GIRAT: Sortie correspondant au frein de giration
116	-(U)-	008	FREIN GIRAT: Sortie correspondant au frein de giration
211	-( )-	011	AVERT SONOR: Sortie actionnant l'avertisseur sonore

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Cross Reference Page 20

INTERNAL

Address	Element	Rung Number(s)	Instruction Comment
701	-] [-	077, 078	AUXIL VIT1-: Variable interne utilisee pour la construction de ARRET VIT 1
701	-( )-	077	AUXIL VIT1-: Variable interne utilisee pour la construction de ARRET VIT 1
702	-] [-	081, 082, 083	ARRET VIT 1: Front descendant de la sortie -MOT- VIT 1
702	-]\[-	077	ARRET VIT 1: Front descendant de la sortie -MOT- VIT 1
702	-( )-	078	ARRET VIT 1: Front descendant de la sortie -MOT- VIT 1
703	-] [-	016, 017	AUXIL DISTR: Variable interne utilisee pour la construction de ARRET DISTR
703	-( )-	016	AUXIL DISTR: Variable interne utilisee pour la construction de ARRET DISTR
704	-] [-	018, 019, 020	ARRET DISTR: Front descendant de la sortie -MOT- DISTR (arret du moteur)
704	-]\[-	016	ARRET DISTR: Front descendant de la sortie -MOT- DISTR (arret du moteur)
704	-( )-	017	ARRET DISTR: Front descendant de la sortie -MOT- DISTR (arret du moteur)
705	-] [-	026, 027	AUXIL TRANS: Variable interne utilisee pour la construction de ARRET TRANS
705	-( )-	026	AUXIL TRANS: Variable interne utilisee pour la construction de ARRET TRANS
706	-] [-	028, 029, 030	ARRET TRANS: Front descendant de la sortie -MOT- TRANS (arret du moteur)
706	-]\[-	026	ARRET TRANS: Front descendant de la sortie -MOT- TRANS (arret du moteur)
706	-( )-	027	ARRET TRANS: Front descendant de la sortie -MOT- TRANS (arret du moteur)
707	-] [-	036, 037	AUXIL GIRAT: Variable interne utilisee pour la construction de ARRET GIRAT
707	-( )-	036	AUXIL GIRAT: Variable interne utilisee pour la construction de ARRET GIRAT
708	-] [-	038, 039, 040	ARRET GIRAT: Front descendant de la sortie -MOT- GIRAT (arret du moteur)
708	-]\[-	036	ARRET GIRAT: Front descendant de la sortie -MOT- GIRAT (arret du moteur)
708	-( )-	037	ARRET GIRAT: Front descendant de la sortie -MOT- GIRAT (arret du moteur)
712	-] [-	082, 084	CR \/ BLOQU: Blocage de la montee du crochet active par la descente
712	-]\[-	042	CR \/ BLOQU: Blocage de la montee du crochet active par la descente
712	-( )-	082	CR \/ BLOQU: Blocage de la montee du crochet active par la descente
714	-] [-	019, 021	<- CH BLOQU: Blocage de l'avance du chariot active par le recul de celui-ci
714	-]\[-	012	<- CH BLOQU: Blocage de l'avance du chariot active par le recul de celui-ci
714	-( )-	019	<- CH BLOQU: Blocage de l'avance du chariot active par le recul de celui-ci

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Cross Reference Page 21

INTERNAL

Address	Element	Rung Number(s)	Instruction Comment
716	-] [-	029, 031	<- GR BLOQU: Blocage de l'avance de la grue active par le recul de celle-ci
716	-]\[-	022	<- GR BLOQU: Blocage de l'avance de la grue active par le recul de celle-ci
716	-( )-	029	<- GR BLOQU: Blocage de l'avance de la grue active par le recul de celle-ci
718	-] [-	039, 041	<< FL BLOQU: Blocage de la rot. droite de la fleche active par la rot. gauche
718	-]\[-	032	<< FL BLOQU: Blocage de la rot. droite de la fleche active par la rot. gauche
718	-( )-	039	<< FL BLOQU: Blocage de la rot. droite de la fleche active par la rot. gauche
722	-] [-	083, 084	CR /\ BLOQU: Blocage de la descente du crochet active par la montee
722	-]\[-	043	CR /\ BLOQU: Blocage de la descente du crochet active par la montee
722	-( )-	083	CR /\ BLOQU: Blocage de la descente du crochet active par la montee
724	-] [-	020, 021	CH -> BLOQU: Blocage du recul du chariot active par l'avance de celui-ci
724	-]\[-	013	CH -> BLOQU: Blocage du recul du chariot active par l'avance de celui-ci
724	-( )-	020	CH -> BLOQU: Blocage du recul du chariot active par l'avance de celui-ci
726	-] [-	030, 031	GR -> BLOQU: Blocage du recul de la grue active par l'avance de celle-ci
726	-]\[-	023	GR -> BLOQU: Blocage du recul de la grue active par l'avance de celle-ci
726	-( )-	030	GR -> BLOQU: Blocage du recul de la grue active par l'avance de celle-ci
728	-] [-	040, 041	FL >> BLOQU: Blocage de la rot. gauche de la fleche active par la rot. droite
728	-]\[-	033	FL >> BLOQU: Blocage de la rot. gauche de la fleche active par la rot. droite
728	-( )-	040	FL >> BLOQU: Blocage de la rot. gauche de la fleche active par la rot. droite
730	-] [-	046, 047, 048	-MOT- LEVAG: Variable interne utilisee pour l'alimentation du moteur de levage
730	-]\[-	049, 050	-MOT- LEVAG: Variable interne utilisee pour l'alimentation du moteur de levage
730	-( )-	042, 043	-MOT- LEVAG: Variable interne utilisee pour l'alimentation du moteur de levage
731	-] [-	048, 049, 053	LEVAG BIT 2: Bit indicateur de vitesse utilise conjointement avec LEVAG BIT 1
731	-]\[-	046, 047, 051, 052, 054	LEVAG BIT 2: Bit indicateur de vitesse utilise conjointement avec LEVAG BIT 1
731	-(L)-	052	LEVAG BIT 2: Bit indicateur de vitesse utilise conjointement avec LEVAG BIT 1
731	-(U)-	049, 053	LEVAG BIT 2: Bit indicateur de vitesse utilise conjointement avec LEVAG BIT 1
732	-] [-	047, 048, 050, 052, 053, 054	LEVAG BIT 1: Bit indicateur de vitesse utilise conjointement avec LEVAG BIT 2

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Cross Reference Page 22

INTERNAL

Address	Element	Rung Number(s)	Instruction Comment
732	-]\[-	046, 051	LEVAG BIT 1: Bit indicateur de vitesse utilise conjointement avec LEVAG BIT 2
732	-(L)-	051	LEVAG BIT 1: Bit indicateur de vitesse utilise conjointement avec LEVAG BIT 2
732	-(U)-	050, 054	LEVAG BIT 1: Bit indicateur de vitesse utilise conjointement avec LEVAG BIT 2
740	-] [-	055, 056	AUXIL VIT1+: Variable interne utilisee pour la construction de DEMAR VIT 1
740	-( )-	055	AUXIL VIT1+: Variable interne utilisee pour la construction de DEMAR VIT 1
741	-] [-	057, 058	DEMAR VIT 1: Front montant de la sortie -MOT- VIT 1 (demarrage du moteur)
741	-]\[-	055	DEMAR VIT 1: Front montant de la sortie -MOT- VIT 1 (demarrage du moteur)
741	-( )-	056	DEMAR VIT 1: Front montant de la sortie -MOT- VIT 1 (demarrage du moteur)
742	-] [-	060, 061	AUXIL VIT2+: Variable interne utilisee pour la construction de DEMAR VIT 2
742	-( )-	060	AUXIL VIT2+: Variable interne utilisee pour la construction de DEMAR VIT 2
743	-] [-	062, 063	DEMAR VIT 2: Front montant de la sortie -MOT- VIT 2 (demarrage de la deuxieme)
743	-]\[-	060	DEMAR VIT 2: Front montant de la sortie -MOT- VIT 2 (demarrage de la deuxieme)
743	-( )-	061	DEMAR VIT 2: Front montant de la sortie -MOT- VIT 2 (demarrage de la deuxieme)
744	-] [-	065, 066	AUXIL VIT2-: Variable interne utilisee pour la construction de ARRET VIT 2
744	-( )-	065	AUXIL VIT2-: Variable interne utilisee pour la construction de ARRET VIT 2
745	-] [-	069, 070	ARRET VIT 2: Front descendant de la sortie -MOT- VIT 2 (retour en premiere)
745	-]\[-	065	ARRET VIT 2: Front descendant de la sortie -MOT- VIT 2 (retour en premiere)
745	-( )-	066	ARRET VIT 2: Front descendant de la sortie -MOT- VIT 2 (retour en premiere)
746	-] [-	072, 073	AUXIL VIT3-: Variable interne utilisee pour la construction de ARRET VIT 3
746	-( )-	072	AUXIL VIT3-: Variable interne utilisee pour la construction de ARRET VIT 3
747	-] [-	074, 075	ARRET VIT 3: Front descendant de la sortie -MOT- VIT 3 (retour en deuxieme)
747	-]\[-	072	ARRET VIT 3: Front descendant de la sortie -MOT- VIT 3 (retour en deuxieme)
747	-( )-	073	ARRET VIT 3: Front descendant de la sortie -MOT- VIT 3 (retour en deuxieme)
748	-] [-	067, 068	AUXIL CHUT3: Variable interne utilisee pour la construction de ARRET CHUT3
748	-( )-	067	AUXIL CHUT3: Variable interne utilisee pour la construction de ARRET CHUT3
749	-] [-	069, 070	ARRET CHUT3: Front descendant de la temporisation VIT 3 CHUTE

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Cross Reference Page 23

INTERNAL

Address	Element	Rung Number(s)	Instruction Comment
749	-]\[-	067	ARRET CHUT3: Front descendant de la temporisation VIT 3 CHUTE
749	-( )-	068	ARRET CHUT3: Front descendant de la temporisation VIT 3 CHUTE
750	-] [-	079, 080	AUXIL CHUT2: Variable interne utilisee pour la construction de ARRET CHUT2
750	-( )-	079	AUXIL CHUT2: Variable interne utilisee pour la construction de ARRET CHUT2
751	-] [-	058, 059	VIT 1 BLOQU: Blocage de la deuxieme v. active par le demarrage du moteur
751	-]\[-	051	VIT 1 BLOQU: Blocage de la deuxieme v. active par le demarrage du moteur
751	-( )-	058	VIT 1 BLOQU: Blocage de la deuxieme v. active par le demarrage du moteur
752	-] [-	081, 082, 083	ARRET CHUT2: Front descendant de la temporisation VIT 2 CHUTE
752	-]\[-	079	ARRET CHUT2: Front descendant de la temporisation VIT 2 CHUTE
752	-( )-	080	ARRET CHUT2: Front descendant de la temporisation VIT 2 CHUTE
753	-] [-	063, 064	VIT 2 BLOQU: Blocage de la troisieme v. active par le passage 1->2
753	-]\[-	052	VIT 2 BLOQU: Blocage de la troisieme v. active par le passage 1->2
753	-( )-	063	VIT 2 BLOQU: Blocage de la troisieme v. active par le passage 1->2
755	-] [-	070, 071, 079	VIT 2 CHUTE: Attente de la diminution de la deuxieme vitesse
755	-]\[-	044, 045, 046, 080	VIT 2 CHUTE: Attente de la diminution de la deuxieme vitesse
755	-( )-	070	VIT 2 CHUTE: Attente de la diminution de la deuxieme vitesse
757	-] [-	067, 075, 076	VIT 3 CHUTE: Attente de la diminution de la troisieme vitesse
757	-]\[-	044, 045, 046, 047, 054, 068	VIT 3 CHUTE: Attente de la diminution de la troisieme vitesse
757	-( )-	075	VIT 3 CHUTE: Attente de la diminution de la troisieme vitesse
800	-] [-	010, 011, 012, 022, 023, 042	MODE DEGRA: Variable interne signalant le fonctionnement en mode degrade
800	-]\[-	004, 009, 012, 022, 023, 042	MODE DEGRA: Variable interne signalant le fonctionnement en mode degrade
800	-(L)-	009	MODE DEGRA: Variable interne signalant le fonctionnement en mode degrade
800	-(U)-	010	MODE DEGRA: Variable interne signalant le fonctionnement en mode degrade
801	-] [-	001, 002	AUXIL ONOFF: Variable interne utilisee pour la construction de APPUI ONOFF
801	-( )-	001	AUXIL ONOFF: Variable interne utilisee pour la construction de APPUI ONOFF
802	-] [-	003, 004	APPUI ONOFF: Front montant de l'appui sur le bouton-poussoir TOUCH ONOFF

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Cross Reference Page 24

INTERNAL

Address	Element	Rung Number(s)	Instruction Comment
802	-]\[-	001	APPUI ONOFF: Front montant de l'appui sur le bouton-poussoir TOUCH ONOFF
802	-( )-	002	APPUI ONOFF: Front montant de l'appui sur le bouton-poussoir TOUCH ONOFF
803	-] [-	005, 006	AUXIL FREIN: Variable interne utilisee pour la construction de APPUI FREIN
803	-( )-	005	AUXIL FREIN: Variable interne utilisee pour la construction de APPUI FREIN
804	-] [-	007, 008	APPUI FREIN: Front montant de l'appui sur le bouton-poussoir TOUCH FREIN
804	-]\[-	005	APPUI FREIN: Front montant de l'appui sur le bouton-poussoir TOUCH FREIN
804	-( )-	006	APPUI FREIN: Front montant de l'appui sur le bouton-poussoir TOUCH FREIN

Date : 13 avril 1996 Dernière révision : 27 avril 1997
Commande d'une grue a tour
SLC Personal Computer Software Cross Reference Page 25

TIMER/COUNTER/SEQUENCER/RESET

Address	Element	Rung Number(s)	Instruction Comment
901	-]\[-	058	TEMPO VIT1+: Temporisation declenchee des le demarrage du moteur (premiere)
901	-(RTO)-	059	TEMPO VIT1+: Temporisation declenchee des le demarrage du moteur (premiere)
901	-(RST)-	057	TEMPO VIT1+: Temporisation declenchee des le demarrage du moteur (premiere)
902	-]\[-	082, 083	TEMPO LEVAG: Temporisation declenchee des l'arret du moteur de levage
902	-(RTO)-	084	TEMPO LEVAG: Temporisation declenchee des l'arret du moteur de levage
902	-(RST)-	081	TEMPO LEVAG: Temporisation declenchee des l'arret du moteur de levage
903	-]\[-	063	TEMPO VIT2+: Temporisation declenchee des le demarrage de la deuxieme vitesse
903	-(RTO)-	064	TEMPO VIT2+: Temporisation declenchee des le demarrage de la deuxieme vitesse
903	-(RST)-	062	TEMPO VIT2+: Temporisation declenchee des le demarrage de la deuxieme vitesse
904	-]\[-	019, 020	TEMPO DISTR: Temporisation declenchee des l'arret du moteur de distribution
904	-(RTO)-	021	TEMPO DISTR: Temporisation declenchee des l'arret du moteur de distribution
904	-(RST)-	018	TEMPO DISTR: Temporisation declenchee des l'arret du moteur de distribution
905	-]\[-	070	TEMPO VIT2-: Temporisation declenchee des l'arret de la deuxieme ou auto 3->2
905	-(RTO)-	071	TEMPO VIT2-: Temporisation declenchee des l'arret de la deuxieme ou auto 3->2
905	-(RST)-	069	TEMPO VIT2-: Temporisation declenchee des l'arret de la deuxieme ou auto 3->2
906	-]\[-	029, 030	TEMPO TRANS: Temporisation declenchee des l'arret du moteur de translation
906	-(RTO)-	031	TEMPO TRANS: Temporisation declenchee des l'arret du moteur de translation
906	-(RST)-	028	TEMPO TRANS: Temporisation declenchee des l'arret du moteur de translation
907	-]\[-	075	TEMPO VIT3-: Temporisation declenchee des l'arret de la troisieme vitesse
907	-(RTO)-	076	TEMPO VIT3-: Temporisation declenchee des l'arret de la troisieme vitesse
907	-(RST)-	074	TEMPO VIT3-: Temporisation declenchee des l'arret de la troisieme vitesse
908	-]\[-	039, 040	TEMPO GIRAT: Temporisation declenchee des l'arret du moteur de giration
908	-(RTO)-	041	TEMPO GIRAT: Temporisation declenchee des l'arret du moteur de giration
908	-(RST)-	038	TEMPO GIRAT: Temporisation declenchee des l'arret du moteur de giration

Liste des figures

Bibliographie

Livres et articles

[LUT93] LUTZ, R.,
Analysing Software Requirements Errors in Safety-Critical, Embedded Systems,
RE '93, IEEE International Symposium On Requirements Engineering,
San Diego, California, janvier 1993, p. 126-133,
ISBN 0-8186-3120-1

[MEY85] MEYER, B.,
On Formalisms in Specifications,
IEEE Software, janvier 1985

[MLE79] MICHEL, G., LAURGEAU, C. & ESPIAU B.,
Les automates programmables industriels,
Paris, Bordas, 1979, 273 p.
ISBN 2-04-010874-2

[SOM92] SOMMERVILLE, I.,
Le génie logiciel,
Paris, Addison-Wesley, 1992, 638 p.
ISBN 2-87908-033-9

[WIE95] WIERINGA, R.,
An Introduction to Requirements Traceability,
Rapport n°IR-389, Faculteit der Wiskunde en Informatica,
Amsterdam, septembre 1995, 22 p.

Documents non publiés, notes de travail

[JUN96] JUNGEN, B.,
The Albert II Reference Manual, version 1.1,
FUNDP, février 1996, 127 p.

[ZEI95a] ZEIPPEN, J.M.,
Etude de cas GRUE : "Repository",
CEDITI-FUNDP, mai - septembre 1995, 28 p.

[ZEI95b] ZEIPPEN, J.M.,
A propos de l'étude de cas GRUE réalisée dans le cadre du projet CAT,
CEDITI-FUNDP, décembre 1995, 36 p.

Articles, vade-mecum et manuels

[ABC87] *User's Manual : Bulletin 1745 SLC Programmable Controllers*,
Milwaukee, Allen-Bradley Company, 1987, 192 p.

[ABC89] *User's Manual : Bulletin 1745 SLC Personal Computer Software*,
Milwaukee, Allen-Bradley Company, 1989, 60 p.

[CNAC80] Notes de sécurité de construction, fascicule n° 19,
Les grues à tour de chantier,
Première partie : conseils pour la prévention des accidents,
C.N.A.C. (Comité National d'Action pour la sécurité et l'hygiène dans la Construction), 1980, 23 p.

[RGPT96] Articles 267, 268 et 269,
dans : *Règlement Général pour la Protection du Travail,*
Tome I, Titre III, Ch. 1er, Section II : Appareils de levage
mise à jour de mai 1996, p. 120-124.

Publication officielle en collaboration avec le Ministère de l'Emploi et du Travail, de l'A.N.P.A.T. (Association Nationale pour la Prévention des Accidents du Travail) et de l'A.C.S.H.B. (Association des Chefs de Sécurité et d'Hygiène de Belgique).
Publié par UGA.