Titre: Title:	Télésignalisation d'inicateurs de défaut d'un réseau souterrain 25 kV
Auteur: Author:	Jean-François Théorêt
Date:	1997
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Théorêt, J.-F. (1997). Télésignalisation d'inicateurs de défaut d'un réseau souterrain 25 kV [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/9026/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/9026/
Directeurs de recherche: Advisors:	Guy Olivier
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

TÉLÉSIGNALISATION D'INICATEURS DE DÉFAUT D'UN RÉSEAU SOUTERRAIN 25 kV

JEAN-FRANÇOIS THÉORÊT
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
JANVIER 1997

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

TÉLÉSIGNALISATION D'INICATEURS DE DÉFAUT D'UN RÉSEAU SOUTERRAIN 25 kV

présenté par: THÉORÊT Jean-François

en vue de l'obtention du diplôme de: Maîtrise ès Sciences Appliquées

a été dûment accepté par le jury d'examen constitué de:

M. HOULE Jean-Louis, Ph.D., président

M. OLIVIER Guy, Ph.D., membre et directeur de recherche

M. DAIGNEAULT Gaétan, M.Sc.A., membre et codirecteur de recherche IREQ

M. ST-JEAN Guy, M.Sc., membre

REMERCIEMENTS

Je tiens à remercier sincèrement Monsieur Gaétan Daigneault de m'avoir permis de participer activement à ce projet, de m'avoir ouvert les portes sur différents autres projets reliés à celui-ci et de m'avoir appuyé d'un apport financier appréciable. Je veux aussi remercier Monsieur Guy Olivier pour son appui en tant que directeur de recherche, pour avoir réglé tous les petits problèmes administratifs tant au niveau de la bourse que de mon admission.

Je tiens aussi à remercier les examinateurs, Monsieur Gaétan Daigneault du Service Appareillage Électrique de l'IREQ, Monsieur Jean-Louis Houle de la section Informatique de l'École Polytechnique, Monsieur Guy Olivier du département de Génie Électrique section Électrotechnique de l'École Polytechnique et Monsieur Guy St-Jean du Service Appareillage Électrique de l'IREQ pour leur temps précieux consacré à la lecture de ce document et aux judicieux commentaires et opinions qui ont été formulés.

Je veux aussi exprimer ma reconnaissance à mon père, Richard Théorêt, pour son aide et sa patience pour la correction de la syntaxe et de l'orthographe de ce document, et pour ses conseils judicieux quant à son contenu.

Finalement, je remercie tout particulièrement mon épouse pour son support, sa compréhension et son encouragement tout au long de ce travail.

RÉSUMÉ

L'indice de continuité de service (*Ic*) prend une part de plus en plus importante dans la philosophie de qualité totale d'Hydro-Québec. Cet indicateur, défini comme la moyenne du nombre d'heures d'interruption de chaque client d'un poste de distribution, permet de comparer la performance des différents postes de distribution souterraine entre eux. Il permet de quantifier les pertes d'argent encourues par Hydro-Québec et par le client lors d'une interruption de service, et d'exprimer la qualité de service offert au client.

L'indice de continuité de service dépend donc directement du nombre de défauts annuels sur les lignes de distribution et de leur temps de détection et de réparation. Un défaut sur une ligne de distribution peut présentement être détecté de deux façons:

- un appel du client au service à la clientèle représente souvent le premier indicateur d'une panne du réseau;
- les outils de télécommande des disjoncteurs du poste signalent aussi aux opérateurs qu'un défaut s'est produit. Toutefois, ce moyen est limité, puisque dans plusieurs des cas, chaque disjoncteur protège deux lignes (un départ double). On ne peut alors savoir sur quelle ligne exactement le défaut s'est produit.

Lorsque les opérateurs du poste sont conscients du défaut, ils dépêchent alors un dépanneur au poste afin de savoir laquelle des deux lignes protégées par le disjoncteur ouvert est la cause du problème. Cette opération prend environ une demi-heure. Lorsque le dépanneur a identifié la ligne défectueuse, il en informe aussitôt les opérateurs au Centre d'Exploitation de Distribution (CED). Les opérateurs envoient alors une patrouille le long de la ligne en question afin qu'ils puissent identifier l'endroit où le défaut s'est produit. La patrouille doit se rendre dans chacun des puits de transformation jusqu'à ce qu'elle découvre l'origine

du problème. Cette opération prend dans le meilleur des cas une demi-heure. Il s'est donc écoulé plus d'une heure entre la détection d'un défaut et l'identification de sa source.

Dans le but d'améliorer cet indice de qualité en améliorant le temps réponse lors d'un défaut, une équipe du service Appareillage électrique de l'Institut de recherche d'Hydro-Québec, pilotée par Monsieur Gaétan Daigneault, a mis au point un système de télésignalisation d'indicateurs de défaut. Ce système permet la télétransmission de l'état des lignes de distribution souterraine (en défaut ou non) ainsi que leur charge moyenne.

L'objectif principal du projet est la réduction du temps d'une interruption de service causée par une panne du réseau de distribution réduisant ainsi les pertes monétaires reliées à cette interruption et permettant d'améliorer la continuité de service au client. Cet objectif peut être atteint en améliorant l'indice de continuité de service.

Le projet comporte deux phases. En premier lieu, le système a été déployé dans le poste, aux têtes de câble de distribution. Cette première phase permettra la validation du système avant son implantation généralisée. La seconde phase du projet concernera le déploiement en réseau du système de télésignalisation. L'implantation de ce système permettra d'améliorer considérablement l'indice de continuité de service, en diminuant le temps nécessaire à la détection de l'endroit d'un défaut.

Le système de télésignalisation est composé de trois modules distincts, dont chacun est responsable d'une tâche bien définie.

1. ***Le système d'acquisition de données*** est composé des indicateurs de défaut télésignalisés d'une part et des contrôleurs locaux d'autre part. Les indicateurs choisis pour le projet pilote sont des indicateurs mécaniques triphasés Fisher-Pierce 1547A modifiés. Ces indicateurs transmettent à la fois la valeur du courant nominal de la ligne sous surveillance et la présence d'un défaut sur la phase en question.

Les contrôleurs locaux quant à eux sont des systèmes d'acquisition synchrones, chacun capable de traiter 32 canaux de 12 bits à 1 kHz. Quelques 126 indicateurs de défaut ont été installés sur quatre contrôleurs locaux.

2. ***Le contrôleur central*** est un ordinateur de type PC dans lequel un système d'exploitation UNIX a été installé. Il sert à la fois de base de données pour les signaux échantillonnés, de serveur de requêtes pour les interfaces usagers et de système de contrôle général pour le système de télésignalisation.

3. ***L'interface usager***, une application fenêtrée fonctionnant sous MS-Windows, présente de façon conviviale les différents paramètres du système. Utilisée par les opérateurs du Centre d'Exploitation de Distribution, elle signale la présence d'un défaut, permet l'affichage de la variation des différents signaux dans le temps et permet d'effectuer un suivi des alarmes de fonctionnement.

L'ensemble du système a été conçu pour utiliser une technologie disponible et fiable. L'adaptation d'une technologie existante aux besoins du projet permet une implantation et une mise en marche rapide.

Le coût total du projet est d'environ 1 M$. Au site d'implantation, une amélioration de 100 % de l'indice de continuité de service représenterait une économie de 9,5 M$. L'équipe du système de surveillance prévoit avoir un impact positif minimum de 25 % sur l'indice de continuité, signifiant 2,4 M$ de pertes évitées.

Le poste Guy, situé au coeur du centre ville de Montréal, a été choisi comme lieu d'implantation du projet pilote. Plusieurs raisons ont motivé l'implantation à cet endroit.

1. ***Son importance stratégique.*** Le poste Guy dessert 23 635 clients. La grande majorité de ces clients sont du secteur commercial. Ces clients dépendent du service de distribution énergétique pour le déroulement de leurs activités quotidiennes.

2. ***Sa taille.*** Le poste Guy est le plus gros poste de distribution souterraine d'Hydro-Québec. Ses 21 départs doubles portent une charge moyenne de 16,1 kVA·h par client desservi.

3. ***Son indice de continuité.*** Le poste Guy a un indice de continuité de service d'environ une heure. Cet indice est excellent, compte tenu que les autres postes de distribution ont en moyenne des indices de deux heures ou plus. Si le système de télésignalisation porte fruit à cet endroit, il sera *de facto* profitable sur les autres.

4. ***Sa disponibilité et sa proximité.*** La proximité du site de l'IREQ et la disponibilité des installations et de son personnel en ont fait le site idéal d'implantation.

5. ***Les départs doubles.*** Son architecture à départs doubles permet l'implantation en deux phases. La première phase peut donc servir à justifier la seconde phase plus dispendieuse.

Tous ces facteurs ont contribué à la sélection du poste Guy comme endroit d'implantation du système de télésignalisation des indicateurs de défaut.

Le projet en est présentement à sa première phase: le système entier a été installé, et après l'installation des indicateurs de défaut sur les lignes, il entrera dans sa phase de validation.

L'implantation du système de télésignalisation des indicateurs de défaut amènera des impacts directs sur:

- ***Le temps réponse lors d'une interruption.*** L'implantation des deux phases du projet permettra de sauver de 20 à 30 minutes d'interruption chez le client. Ceci aura des répercussions immédiates sur la diminution des pertes encourues par Hydro-Québec et sur l'amélioration de la qualité de service du client.

- ***La gestion de la charge du réseau de distribution.*** Puisque le système donne une idée de la charge instantanée de chacune des lignes de distribution, il permet une

gestion de charge, pouvant être utilisée lors de la planification de retraits ou lors de déviations.

- ***La validation de la technologie des indicateurs de défaut télésignalisés.*** L'implantation du projet permettra d'établir l'utilité et la rentabilité de la télésignalisation des indicateurs de défaut.

À ce jour, les indicateurs sont installés sur cinq des vingt et une lignes doubles prévues; même si l'installation n'est pas encore complétée, le système permet déjà de voir la variation de la charge des quatre lignes et de détecter les défauts sur ces mêmes lignes. Pour que les utilisateurs puissent tirer profit de ces informations, la formation des utilisateurs du système est présentement en cours.

L'implantation du projet de télésignalisation des indicateurs de défaut sur les lignes de distribution souterraine permettra à Hydro-Québec d'économiser sur les pertes d'argent encourues lors de défauts en plus de fournir un service de meilleure qualité au client. Quoique l'implantation ne soit pas complétée, elle permet déjà d'obtenir des résultats intéressants et utilisables.

De plus, l'intérêt suscité par le projet a poussé l'équipe responsable du système de télésignalisation des indicateurs de défaut à envisager un système équivalent pour les indicateurs de défaut aérien.

ABSTRACT

The continuity index (*Ic*) is taking more and more importance in Hydro-Quebec's total quality approach to power distribution. This quality indicator, defined as the average number of hours of interrupted service of all customers of a distribution substation, helps in comparing the performance of different substations. It can also be used to quantify the financial losses incurred by Hydro-Quebec and its customers, and to clearly define the quality of the service offered to the customers.

The continuity index depends directly on the number of faults that occurred on distribution lines and the time taken to detect and repair these faults. Up to now, a fault can be detected in two ways:

- a call from a customer to Hydro-Québec's customer service is often the first indication of a problem on the distribution network;
- remote control systems on high voltage breakers in the distribution substation signal the network operators whenever a fault occurs. This detection method lacks precision: most of the lines are installed on parallel feeders: when a breaker opens in the event of a fault, it isolates both lines installed on the feeder. The operators cannot identify which line has caused the fault.

When the operators notice a fault either through customer calls or through the breaker control systems, they send an intervention team to the substation to have the defective line identified. This operation takes about half an hour. As soon as the faulty line is identified, an other intervention team is dispatched to patrol the problem line to identify the exact location of the fault. This last operation also takes at least half an hour. Between the time a fault is detected and the time its exact location is detected, at least one hour will have passed.

A research team from Hydro-Québec's Research Institute Power Apparatus Group, lead by Mister Gaétan Daigneault, has put together a system for the remote signaling of fault indicators. This system transmits both the indicator's status and the average load of the line on which the fault indicator is installed.

The principal objective of the project is to reduce the time needed to locate a fault, lessening monetary losses caused by the interruption and to improve the quality of the service offered to the customers.

The project comprises two phases, the first of which is successfully completed. It involved instaling remotely signalled fault indicators at the distribution cable heads of the 21 parallel feeders of the substation's 25 kV RCL lines. The second phase of the project involves installing remote fault indicators in transformer vaults strategically located along the substation lines. This system will reduce the time taken to repair a fault, since it will enable the network operators to pinpoint the location of the fault as soon as it happens.

The system for the remote signalling of fault indicators is divided in three distinct modules, each module having a well defined task:

1. ***The data acquisition system*** includes the remotely signalled fault indicators and the local controllers. The indicators chosen are Fisher Pierce mechanical indictors model 1547A-SCADA. The SCADA (Supervisory Control And Data Acquisition) version can transmit wether the indicator detected a fault or not, and a tap was installed on the indicator's current transformer to be able to read the current of each line.

   The local controllers are synchronous acquisition systems, capable of sampling 32 channels at 1 kHz. 126 indicators have been installed for the system to monitor the 21 parallel feeders of Guy substation.

2. ***The central controller*** is a PC type computer running a UNIX operating system. It serves as a database for the sampled datum, as a request server for the user interfaces and as a general control system for the remote signaling system;

3. ***The user interface***, a windowed application under MS-Windows, displays the sampled datum to an end user in an easily comprehensible fashion. Used by the network technicians, it enables them to immediately detect and pinpoint the location of a fault, to determine the average load of each monitored line and manage operating alarms.

The whole of the system has been built with readily available parts that have matured and attained a reliable state; use of such technology reduces development and set up times. This inexpensive technology also reduced the global cost of the project, which rounded to about 1 M$. Even though this amount seems considerable, it is substantially smaller than the savings that will result from the use of the system: at Guy substation, a 100 % improvement in the continuity index would mean savings of 9,5 M$. We have established a minimum improvement of 25 % on the continuity index, rendering a minimum gain of 2,4 M$.

Guy substation, located in the heart of Montreal's commercial district, has been chosen as the system's test site. Five reasons motivated this choice.

1. ***Its strategic importance.*** Guy substation serves 23 635 customers. The great majority of these customers are businesses that depend on electricity for their daily activities.

2. ***Its size.*** Guy substation is Hydro-Québec's largest underground distribution center. Its 21 25 kV parallel feeders distribute an average of 16,1 kVA·h per customer.

3. ***Its continuity index.*** Guy substation has a continuity index of about one hour. Compared to other underground distribution centers, this is excellent: most substa-

tions average around two to three hours. If the system is successful in improving Guy's continuity index, it will also help other substations.

4. ***Its availability and proximity.*** Guy substation is located near IREQ, and the facilities and its personnel were fully available, making it an ideal site.

5. ***Its parallel feeders.*** Guy substations's architecture enabled a two phase set up. The first phase will serve in justifying the second phase, which is more complex and more expensive.

All these factors contributed in making Guy substation the ideal site to establish the system for the remote signalling of fault indicators.

Up to now, the first phase of the system has presently been fully deployed, and the second phase will start as soon as all the fault indicators have been installed. As soon as this second stage will start, the system will have immediate impacts on:

- ***the response time during an interruption caused by a fault:*** about 20 to 30 minutes of interruption will be avoided. This will have direct repercussions on Hydro-Québec's losses and on the customer's quality of service.

- ***load management:*** since the system gives an idea of the instant load of each monitored line, a basic load management is possible whenever load balancing is necessary or when withdrawal of lines for maintenance is necessary.

- ***validation of remote signalling of indicators technology***: the installation of the system will help to establish the usefulness and the profitability of the remote signaling of fault indicators technology.

Up to now, six out of the 21 parallel feeders have their fault indicators installed; even if the whole system has not been installed, benefits are already being identified. For the first time,

system load can be examined in real time and faults can be located as soon as they appear. For system users to exploit the full capabilities of the system, training is under way.

On the long run, the set up of a system for the remote signaling of underground fault indicators will enable Hydro-Quebec to control some of its losses and to offer a better quality of service to its customers.

Furthermore, interest generated by the project pushed the team into developing a system for the remote signaling of aerial fault indicators in rural distribution networks, that uses RF technology to transmit the contact and load current information.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABBRÉVIATIONS

BNF: *Baccus–Naur Formalism.* Formalisme de Baccus–Naur; formalisme permettant de décrire une unité lexicale de façon récursive.

BSD: *Berkeley System Distribution.* Variante du système UNIX, originaire de l'Université de Berkeley en Californie.

DEL: Diode électro-luminescente.

CED: Centre d'Exploitation de Distribution.

CRC: *Cyclic Redundancy Check.* Contrôle cyclique par redondance. Méthode de calcul permettant de détecter des erreurs de transmission dans une chaîne d'informations.

FFT: *Fast Fourrier Transform.* Transformée de Fourrier rapide.

FIFO: *First In First Out.* Tampon "premier entré, premier sorti".

ISO: *International Standards Organisation*: organisme d'accréditation de normes internationales.

MDI: *Multiple Document Interface.* Norme définissant le comportement d'interfaces.

MCP: *Master Control Processor.* Contrôleur principal du système de télésignalisation des indicateurs de défaut.

OSI: *Open Systems Interconnection*: protocole de communications à sept niveaux défini par l'ISO.

POSIX: *Portable Operating System Interface, IEEE.* Norme sur la portabilité des systèmes d'exploitation.

RCP: *Remore Control Processor.* Contrôleur secondaire esclave du système de télésignalisation des indicateurs de défaut.

RTI: *Return From Interrupt.* Instruction d'assembleur utilisée pour terminer le traitement d'une interruption.

SCADA: *Système de Commande et d'Acquisition de Données Automatiques.* Terme indiquant qu'un équipement peut être utilisé pour le contrôle et l'acquisition de données.

SVR4: *System V (5), Release 4.* Variante de système UNIX, originaire des laboratoires Bell.

XON: Caractère ASCII '^Q' utilisé dans le protocole de contrôle de flux. Demande la fin de l'interruption des communications.

XOFF: Caractère ASCII '^S' utilisé dans le protocole de contrôle de flux. Demande l'interruption des communications.

LISTE DES ANNEXES

INTRODUCTION

L'indice de continuité de service (I_c) prend une part de plus en plus importante dans la philosophie de qualité totale d'Hydro-Québec. Cet indicateur, défini comme la moyenne du nombre d'heures d'interruption de chaque client d'un poste de distribution, permet de comparer la performance des différents postes entre eux. Il permet de quantifier les pertes d'argent encourues par Hydro-Québec et par le client lors d'une interruption de service et d'exprimer la qualité de service offerte au client.

Dans le but d'améliorer cet indice de qualité, une équipe du Service Appareillage électrique de l'Institut de recherche d'Hydro-Québec, pilotée par Monsieur Gaétan Daigneault, a mis au point un système de télésignalisation d'indicateurs de défaut. Ce système permet la télétransmission de l'état des lignes de distribution souterraines (en défaut ou non) ainsi que leur charge moyenne.

Le projet comporte deux phases: en premier lieu, le système sera déployé dans le poste, aux têtes de câble de distribution. Cette première phase permet la validation du système avant son implantation plus généralisée. La seconde phase du projet déploiera en réseau le système de télésignalisation. L'implantation de ce système permettra de réduire considérablement l'indice de continuité de service, en diminuant le temps nécessaire à établir l'endroit d'un défaut.

Le document donne les principaux objectifs et justifications du projet, établit son contenu, et explique son fonctionnement ainsi que son implantation. Certaines sections pourront tout aussi bien servir de manuel d'instruction que de document de formation.

CHAPITRE I

OBJECTIFS ET JUSTIFICATION DU PROJET

L'objectif principal du projet est la réduction du temps d'une interruption de service causée par une panne du réseau de distribution, permettant de diminuer les pertes monétaires reliées à cette interruption et d'améliorer la continuité de service au client. Cet objectif peut être atteint en améliorant l'indice de continuité de service. Nous expliquerons ici de quelle façon le projet prévoit réduire l'indice de continuité d'au moins 25 %; nous démontrerons la procédure suivie lors de la détection d'un défaut avant l'implantation du projet et après l'implantation des deux phases prévues. Nous discuterons ensuite du choix du site d'implantation et du seuil de rentabilité du projet.

1.1 Procédure actuellement suivie lors de la détection d'un défaut

Un défaut sur une ligne de distribution peut présentement être détecté de deux façons:

- un appel du client au service à la clientèle représente souvent le premier indicateur d'une panne du réseau;
- les outils de télécommande des disjoncteurs du poste signalent aussi aux opérateurs qu'un défaut s'est produit. Toutefois, ce moyen est limité, puisque dans plusieurs des cas, chaque disjoncteur protège deux lignes (un départ double). On ne peut alors savoir sur quelle ligne exactement le défaut s'est produit.

Lorsque les opérateurs du poste sont conscients du défaut, ils dépêchent alors un technicien au poste afin de savoir laquelle des deux lignes protégées par le disjoncteur ouvert est la cause du problème. Cette opération prend environ une demi-heure.

Lorsque le technicien sur place a identifié la ligne défectueuse, il en informe aussitôt les opérateurs au Centre d'exploitation de distribution (CED). Les opérateurs envoient un second technicien patrouiller la ligne en question, afin qu'il puisse identifier l'endroit ou le tronçon problématique. Le technicien doit se rendre dans chacun des puits de transformation (PT), jusqu'à ce qu'il découvre l'origine du problème. Cette opération prend dans le meilleur des cas une demi-heure.

Il s'est donc écoulé plus d'une heure entre la détection d'un défaut et l'identification de sa source.

1.2 Utilité du système de télésignalisation des indicateurs de défaut

Comme la première phase du projet prévoit l'installation d'indicateurs de défaut aux têtes de câble, la première intervention du technicien sera éliminée, sauvant ainsi une demi-heure d'interruption. La seconde phase, telle qu'illustrée à la figure 1.1, permettra d'éliminer la seconde intervention humaine, réduisant le temps d'interruption d'au minimum une heure.

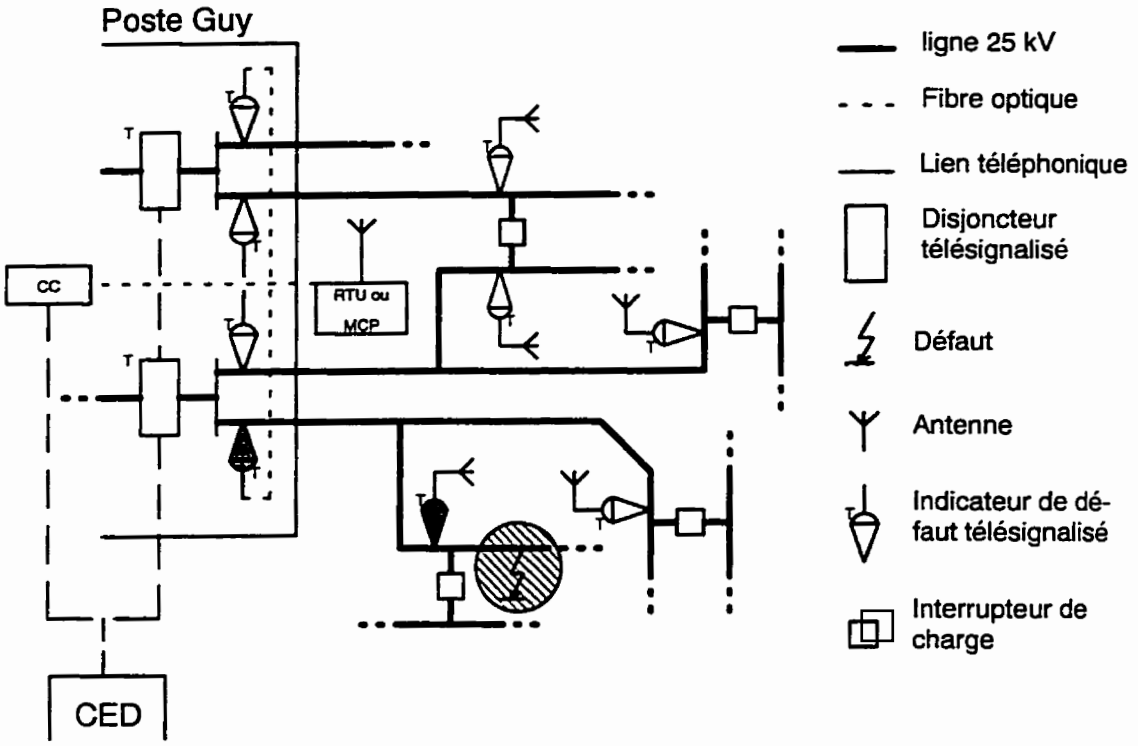


Figure 1.1: Exemple d'utilisation du système de télésignalisation en réseau

Les deux indicateurs de défaut plus foncés de la figure 1.1 indiquent une condition de défaut. Pour cet exemple, un défaut s'est produit à l'endroit indiqué par une zone circulaire hachurée. Les indicateurs de défaut permettent alors d'identifier approximativement l'endroit du problème.

1.3 Choix du lieu de l'implantation

Le poste Guy, situé au coeur du centre ville a été choisi comme lieu d'implantation du projet pilote. Plusieurs raisons ont motivé l'implantation à cet endroit:

- *son importance stratégique*: le poste Guy dessert 23 635 clients. La grande majorité de ces clients sont du secteur commercial. Ces clients dépendent du service de distribution énergétique pour leurs activités quotidiennes;

- *sa taille*: le poste Guy est le plus gros poste de distribution souterraine. Ses 21 départs doubles portent une charge moyenne de 16,1 kVA par client;
- *son indice de continuité*: le poste Guy a un indice de continuité de service d'environ une heure. Cet indice est excellent comparativement à ceux des autres postes qui ont des indices de deux heures ou plus. Si le système de télésignalisation porte fruit à cet endroit, il sera *de facto* profitable sur les autres;
- *sa disponibilité et sa proximité*: la proximité du site de l'IREQ et la disponibilité des installations et des gens en on fait le site idéal d'implantation;
- *les départs doubles*: les départs doubles permettent l'implantation en deux phases, la première servant à justifier la seconde phase plus dispendieuse.

Tous ces facteurs ont contribué à la sélection du poste Guy comme endroit d'installation du système de télésignalisation des indicateurs de défaut.

1.4 Facteurs économiques

L'indice de continuité devient donc un facteur primordial dans l'exploitation du réseau et de plus en plus d'efforts devront y être consacrés. Plusieurs études internes, dont l'étude RCRSD-91-22 (Groupe RCRSD, 1991), indiquent que toute action permettant d'améliorer l'indice de continuité de service, dont le coût net à l'investissement serait inférieur à 25 $ du kVA·h interrompu évité, serait rentable. Comme le poste Guy a actuellement 23 635 clients, une charge moyenne de 16,1 kVA par client et un *Ic* de 1 heure, le gain que l'on pourrait obtenir à réduire l'Ic à zéro serait de 9,5 M $:

$$\frac{25\$}{kVA \cdot h} \times 16,1\,kVA \times 1\,h \times 23635 = 9,5\,M\$$$

Le coût approximatif des deux phases du projet se chiffre à environ 1 M $. Le seuil de rentabilité est établi à partir du graphique de la figure 1.2. Le projet est rentable au moment où l'amélioration de l'indice de continuité est supérieure à 10,5 %. Or, on prévoit une amélioration bien supérieure à cette valeur, soit au minimum 25 % d'augmentation.

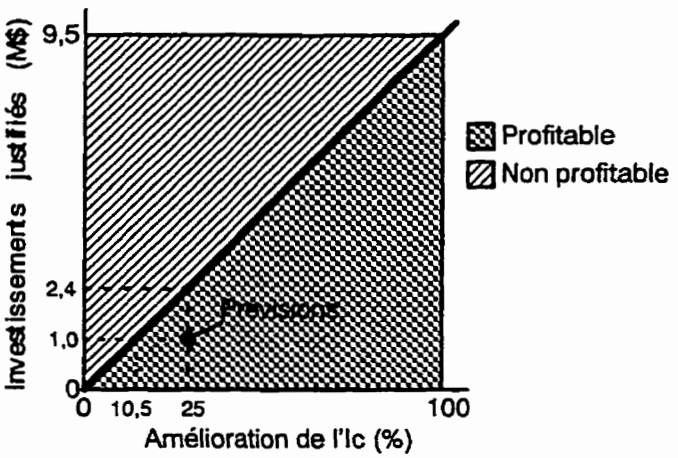


Figure 1.2: Rentabilité du système de télésignalisation

CHAPITRE II
LE SYSTÈME DANS SON ENSEMBLE

Le système de télésignalisation est basé sur le système de surveillance de disjoncteurs haute tension (Bennet, Schwabe *et.al.*, 1993; Landry, Mercier *et.al.*, 1994), conçu à l'origine pour la surveillance de disjoncteurs haute tension situés dans les postes de transformation. Installé sur des disjoncteurs SF_6 à double pression de type SFA de la compagnie Westinghouse Electric Corporation et des disjoncteurs de modèle PK et PKV de la compagnie GEC Alsthom, le système a fait ses preuves maintes fois en permettant l'intervention rapide sur des systèmes défectueux avant leur destruction; de même il a permis des études *à posteriori* lorsqu'il était impossible d'établir des indices de la défaillance éventuelle d'un équipement. L'architecture générale du système de surveillance de disjoncteurs haute tension a été récupérée pour la télésignalisation des indicateurs de défaut. Les sections portant sur l'acquisition de données, sur la transmission et le stockage des données ainsi qu'une partie du logiciel sont récupérables avec un minimum d'adaptation.

Nous présenterons ici la fonctionnalité de base des principales composantes du système, telles qu'elles ont été adaptées du système de surveillance de disjoncteurs de l'IREQ. Chacun des éléments constituants sera détaillé ultérieurement.

2.1 Architecture générale

L'architecture du système est composée de trois modules distincts, chacun ayant une tâche bien définie. Nous décrivons ici ces trois modules.

2.1.1 Le système d'acquisition de données

Le système d'acquisition de données est composé de capteurs non intrusifs installés sur les équipements à surveiller et du contrôleur local, lequel nous désignerons "LC", pour "Local Controller". Pour le projet en cours, les capteurs utilisés sont des indicateurs de défaut triphasés. Les indicateurs choisis pour le projet pilote sont des indicateurs mécaniques Fisher-Pierce, modèle 1547. Pour servir à la télésignalisation, les indicateurs ont dû être modifiés de deux façons: un raccordement a été effectué sur un transformateur de courant à l'intérieur de l'indicateur permettant ainsi la télésignalisation de la valeur du courant, et un contact sec a été installé sur une partie mécanique mobile à l'intérieur de l'indicateur pour la télésignalisation de l'état de la ligne. Des détails supplémentaires sur les indicateurs de défaut apparaissent au chapitre V. En ce qui concerne le contrôleur local, c'est un système en temps réel décomposable à son tour en trois sections spécifiques:

1. plusieurs processeurs esclaves sur lesquels sont raccordés les modules d'acquisition et les indicateurs de défaut. Ces modules portent le nom de "RCP";
2. un processeur maître coordonne l'acquisition des données et la détermination du dépassement des seuils d'alarme et d'événements. Ce module se nomme "MCP";
3. un processeur, appelé "STD" en charge de transférer les informations à la base de données. Un système de logiciels régit et contrôle toutes ces sections et leur interaction.

2.1.2 Le contrôleur central

Le contrôleur central est le second module. Ses tâches sont d'aller chercher les informations du contrôleur local et de les stocker, de répondre aux requêtes des interfaces usagers ainsi que d'effectuer certains calculs sur les données stockées avant de les transmettre aux inter-

faces usagers. Il est possible de relier au moyen de fibres optiques au-delà de 32 contrôleurs locaux à chaque contrôleur central. Plusieurs logiciels sur une plate-forme UNIX pour PC se chargent de remplir ces fonctions.

2.1.3 L'interface usager

L'interface usager a deux utilités; avertir l'usager en cas d'alarmes et fournir de l'information sur les événements et signaux échantillonnées sous forme de graphiques et de tables. L'interface requiert l'utilisation d'un ordinateur personnel ou d'un poste de travail, ainsi qu'un moyen de communication vers l'extérieur. Ce port de sortie peut être soit un port série, un modem, une connexion ethernet sur le réseau Internet, ou une connexion via un serveur de modems sur un réseau ethernet.

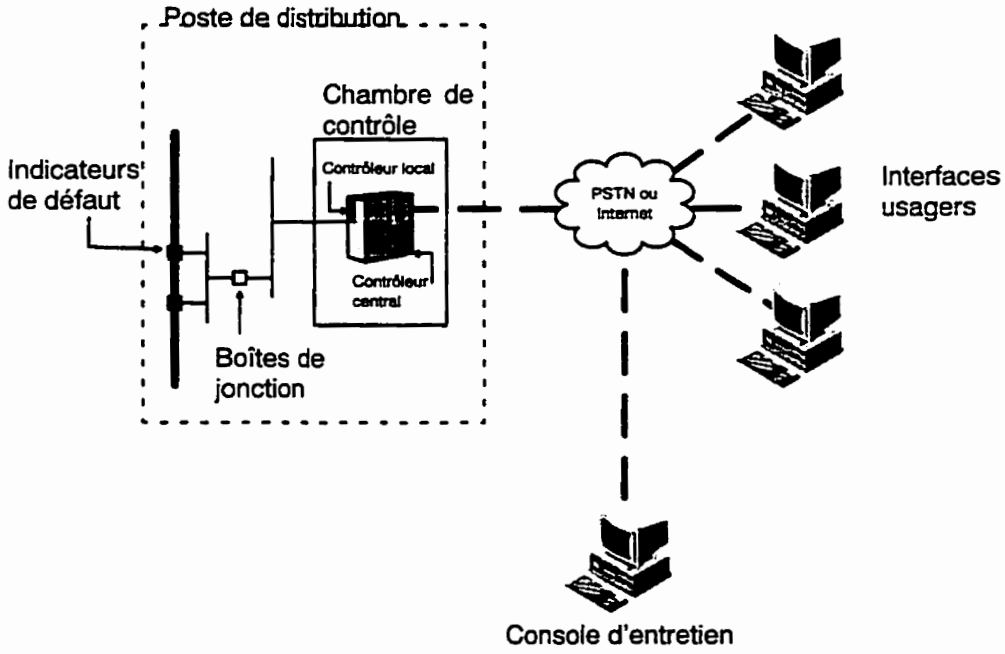


Figure 2.1: Architecture globale du système de télésignalisation

La figure 2.1 illustre la configuration globale du système. À l'intérieur du poste de transformation, on retrouve le contrôleur local ainsi que le contrôleur central. Branchées sur le contrôleur local, plusieurs boîtes de jonction relient les contrôleurs locaux aux indicateurs de défaut. Ces boîtes de jonction sont munies de sectionneurs pour permettre d'isoler les indicateurs de défaut des contrôleurs locaux en cas de besoin. Les lignes grasses pointillées représentent soit un lien IEEE 802.3 (communément appelé ethernet), ou bien un lien série modem (RS-232). Les interfaces usagers et la console d'entretien sont branchés sur le contrôleur central à travers le réseau téléphonique traditionnel (*"PSTN" - "Packet Switched Telephone Network"*) ou à travers le réseau semi-public Internet. Les informations sont extraites des capteurs, traités dans les contrôleurs locaux, transmis à la base de données du contrôleur central, pour finalement être mis à la disponibilité des usagers via l'interface.

2.2 Description sommaire du contrôleur local

Les contrôleurs locaux sont les points d'entrée des informations dans le système. Pour le présent projet, quatre contrôleurs locaux sont nécessaires étant donné le nombre de signaux à échantillonner. Chacun des contrôleurs locaux est apte à faire l'acquisition de 32 canaux rapides (c.f. section pour la description complète des signaux manipulés par le système). Comme le présente la figure 2.2, les quatre contrôleurs locaux sont placés dans une seule et même armoire adjacente au contrôleur central. On remarque au bas du cabinet du contrôleur local les liens de communications vers les boîtes de jonction.

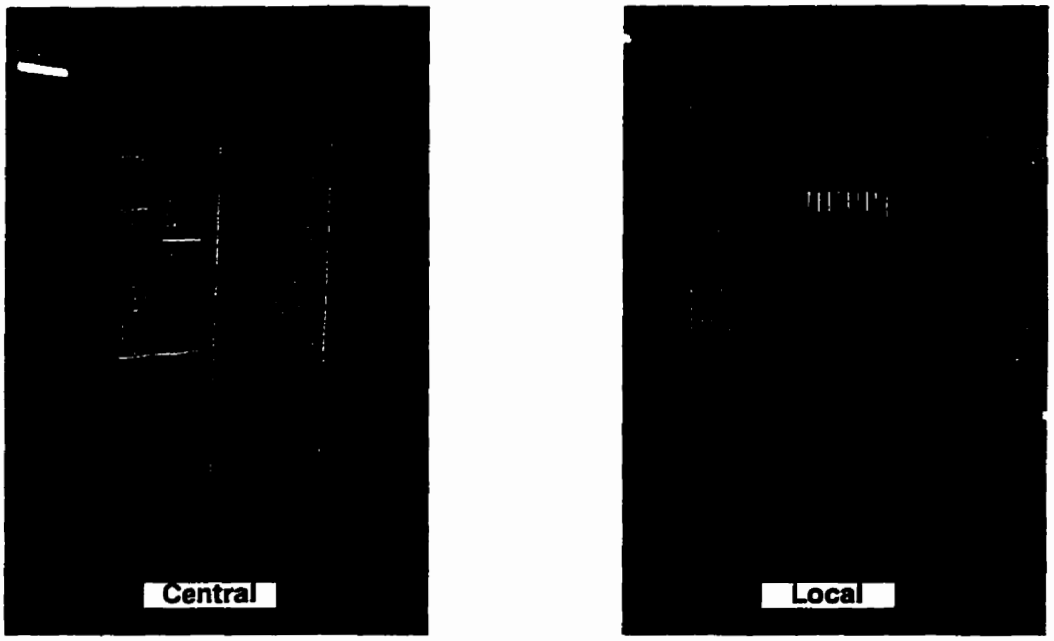


Figure 2.2: Contrôleur central et du contrôleur local

Chaque contrôleur local est relié au contrôleur central au moyen d'un câble série, et aux indicateurs de défaut par le biais d'une boîte de jonction.

Les contrôleurs locaux exécutent une acquisition synchrone des informations. À chaque milliseconde (i.e. à un taux de 1 kHz), le logiciel déclenche l'acquisition de tous les signaux, et ce pour les deux principaux modes de fonctionnement, soit le mode continu et le mode événement (section).

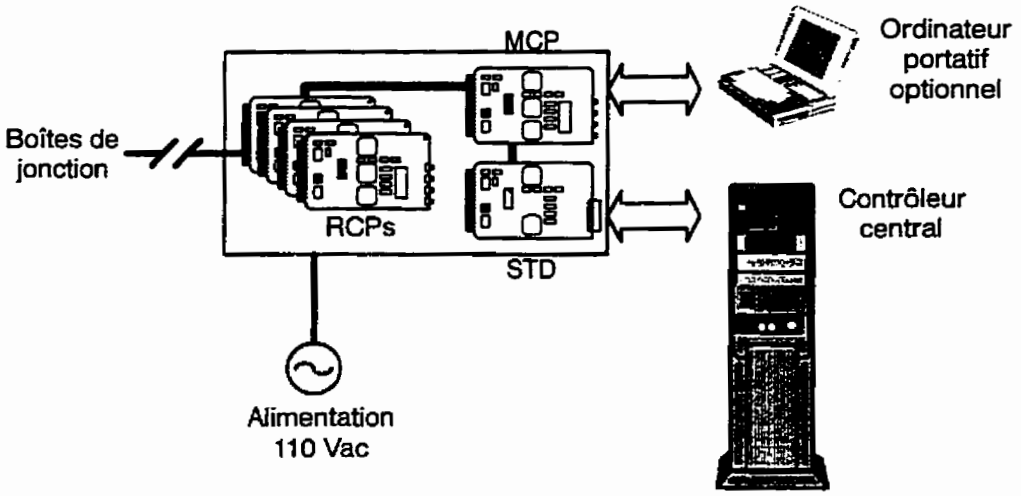


Figure 2.3: Diagramme fonctionnel d'un contrôleur local

En régime continu, le contrôleur local effectue une moyenne des signaux acquisitionnés pour la dernière heure et pour la dernière minute de chaque signal. En mode d'événement, tous les signaux échantillonnés sont conservés sur une plage de temps fixe de 300 ms. Le contrôleur local possède un tampon suffisamment grand pour conserver cinq événements consécutifs. Ces informations peuvent être transmises par un lien série permanent (au moyen de fibre optique ou câble série, selon les besoins de protection contre les interférences électromagnétiques) établi avec le contrôleur central, ou temporairement à un ordinateur PC portatif, permettant l'entretien du système.

2.3 Description sommaire du contrôleur central

Le contrôleur central est un ordinateur PC de type Pentium, servant à la fois de serveur de requêtes pour la base de données et de logiciel de contrôle général du système de surveillance. Étant donné la nature asynchrone des tâches à effectuer, l'utilisation d'un système d'exploitation multitâches est de mise. Le système d'exploitation Linux à été choisi pour cette application à cause de sa compatibilité BSD et SVR4 UNIX, de la disponibilité de son code source, de sa flexibilité, de la quantité de logiciels de qualité disponibles et enfin de son coût négligeable.

En plus du système d'exploitation, le contrôleur central est muni de plusieurs logiciels. Le vérificateur assure la synchronisation de la base de données avec les informations des contrôleurs locaux. Le répartiteur distribue les messages reçus des autres logiciels et composantes matérielles au bon destinataire. Le serveur de commandes vérifie la validité des requêtes en provenance des interfaces usagers et lorsqu'elles sont valides, effectue des requêtes SQL à la base de données. Les résultats de la requête SQL sont alors réacheminées à l'interface usager ayant requis les services de la base de données. Plusieurs petits logiciels de support, tel un programme de minuterie (*"watchdog"*) qui vérifie que les logiciels principaux sont fonctionnels, sont présents dans le système. La figure 2.4 suivante schématise l'interaction entre les divers logiciels du contrôleur central. Les traits gras continus sont des liens d'entrée/sortie standard, tandis que les traits fins continus sont des systèmes de communication par FIFOs (ou files d'attente). Les traits pointillés sont des requêtes SQL à la base de données.

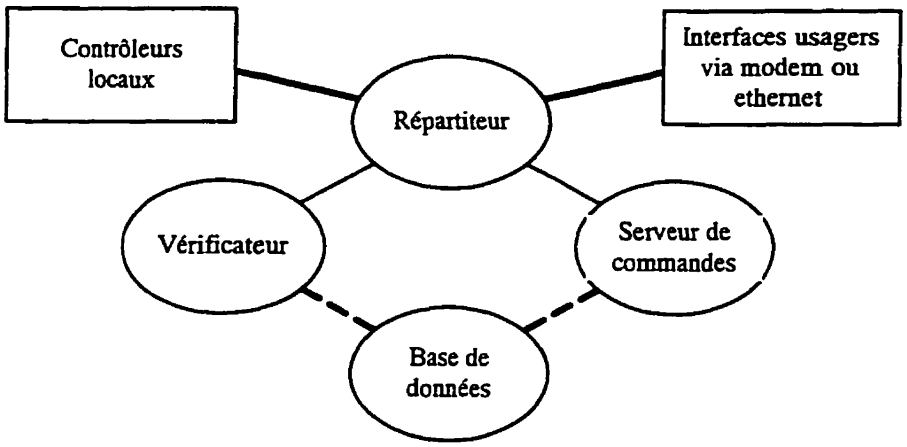


Figure 2.4: Diagramme d'interaction des logiciels du contrôleur central

En plus du logiciel et du boîtier d'ordinateur, plusieurs périphériques sont rajoutés au contrôleur central, tel qu'illustré à la figure 2.5: un écran, un clavier, des modems, des ports série RS-232-D et une unité d'alimentation de sécurité sans coupure (UPS - *"Uninterruptible Power Supply"*) doivent absolument être présents. Le contrôleur central peut aussi être constitué de plusieurs modules optionnels:

- une imprimante servant à générer de façon automatique des rapports;
- une unité de sauvegarde;
- un port ethernet, permettant l'accès simultané de plusieurs usagers via l'interface usager;
- une horloge externe, pour synchroniser plusieurs ordinateurs;
- un panneau d'alarmes, indiquant au moyen de témoins lumineux la présence d'alarmes dans le système.

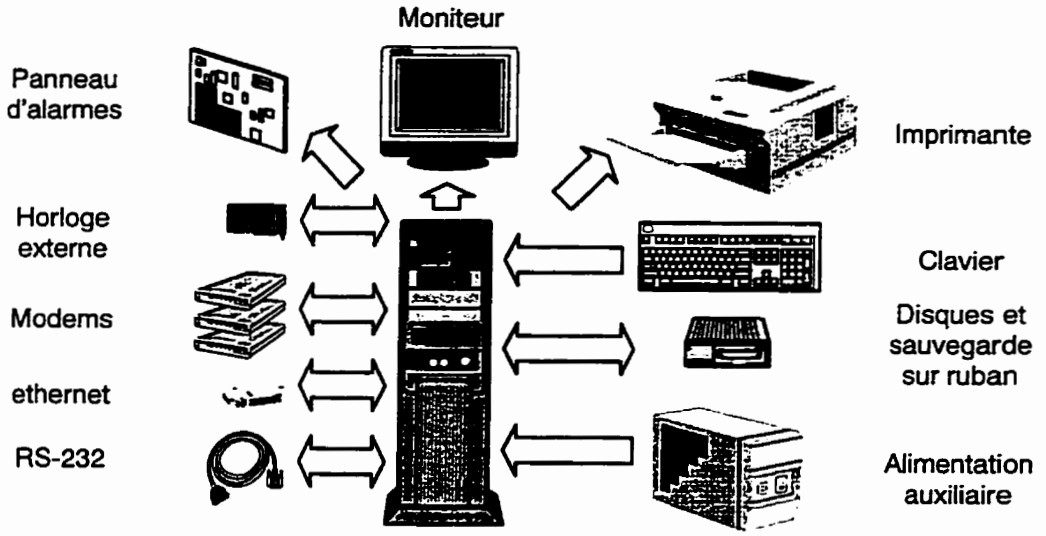


Figure 2.5: Composantes du contrôleur central

CHAPITRE III
DONNÉES DU SYSTÈME DE TÉLÉSIGNALISATION

Dans le système de télésignalisation, on retrouve principalement quatre types de données. Les deux premiers types sont des données échantillonnées, tandis que les deux autres sont des données générées par le MCP.

3.1 Les signaux rapides

Les signaux rapides sont des signaux qui sont échantillonnés à un taux de 1 kHz. Ces signaux sont, par exemple, le courant instantané d'une phase. Certains de ces signaux peuvent être utilisés pour la génération automatique d'événements (voir section 3.4). On conserve la moyenne du signal pour la dernière minute et de la dernière heure en régime permanent. Ces signaux sont vérifiés à chaque milliseconde. Si la valeur du signal dépasse une des bornes supérieures ou inférieures préétablies et modifiables, une alarme est générée. Les alarmes des signaux rapides sont gérées par le MCP. Tous les signaux implantés pour la phase 1 du présent projet sont de cette catégorie;

3.2 Les signaux lents

Les signaux lents sont échantillonnés environ une fois la minute. De la même façon que pour les signaux rapides, ils sont comparés à des valeurs seuil pour des alarmes, mais cette fois-ci par le processeur STD. Ces signaux sont utilisés pour des informations relativement statiques. Des signaux comme la température extérieure sont relativement invariants, et n'ont pas besoin d'une meilleure résolution. L'utilisation de ces signaux plutôt que les signaux rapides permet de décharger le processeur MCP, lui laissant plus de temps pour le

traitement des événements et des alarmes de signaux rapides. Aucun signal de ce genre n'a encore été utilisé pour le projet pilote du poste Guy;

3.3 Les alarmes

Des limites d'alarmes modifiables sont assignées à chacun des signaux échantillonnés par le système. Une alarme est un couple A = {V,T} où V est la valeur référence utilisée lors de la comparaison, et T est soit la fonction " *inférieur à* " ou la fonction " *supérieur à* ".

$$A = \{V, T\}, V \in dom(T) \wedge T \in \{' > ', ' < '\}$$

On distingue trois types d'alarmes: les alarmes majeures, les alarmes mineures et les alarmes de communication. La distinction entre mineure et majeure est subjective et réservée à l'utilisateur; elle lui permet de regrouper des alarmes de son choix selon leur importance relative. Le MCP détecte les alarmes des signaux rapides pour ensuite les communiquer au STD, et le STD gère les alarmes des signaux lents. Une fois au STD, les alarmes attendent le transfert vers le contrôleur central. L'alarme de communication est un cas particulier: celle-ci est générée par le contrôleur central lorsque des erreurs de communication surviennent fréquemment entre lui-même et les différents contrôleurs locaux.

3.4 Les événements

Ces données sont générées de façon automatique, selon les valeurs de certains signaux rapides. En général (i.e. en régime permanent), on ne conserve que la moyenne de la dernière heure et de la dernière minute pour le signal échantillonné. Cependant, en mode événement, on conserve tous les échantillons, pour une période fixe de 300 ms. Il est possible de générer manuellement un tel événement à partir de l'interface usager. Cependant, la force des évé-

nements réside dans le fait que le processeur MCP est en mesure de générer lui-même un événement au besoin.

Pour que le MCP génère automatiquement un événement, on lui indique quel signal *rapide* est générateur d'événement et on associe à ce signal une valeur limite, tout comme pour les alarmes. Si cette valeur limite est dépassée pour au moins 5 ms *consécutives*, le MCP se considère en état d'événement, et conserve les échantillons pour les 50 ms précédant le début de l'événement et les 250 ms suivantes pour tous les signaux. Ces valeurs sont transférées au STD comme étant un événement. Elles attendent ensuite d'être transférées au contrôleur central.

Le concept d'événement peut représenter un événement physique qui a lieu dans un équipement sous surveillance. Dans le cas du projet pilote du poste Guy, cette capacité de générer des événements n'est pas encore exploitée. Voici cependant un exemple tiré de la surveillance de disjoncteurs à air de type PK (" *air blast breaker* "):

> Admettons qu'un signal de pression de réservoir est un signal rapide apte a générer un événement sur un disjoncteur. Fixons les limites d'opération "normales" à [470,500] psig. Supposons que le signal de pression ait le profil de la figure 3.1.

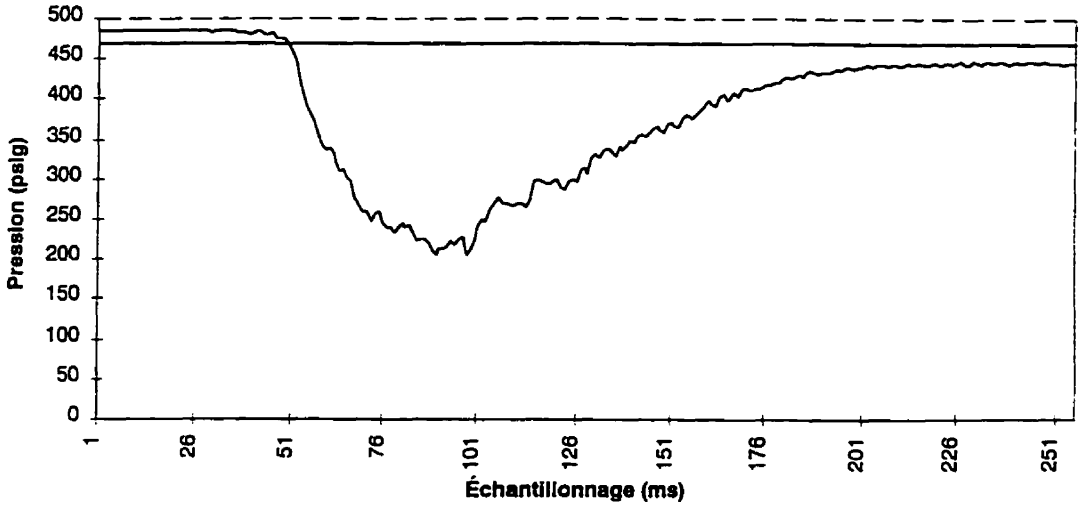


Figure 3.1: Courbe de pression d'ouverture d'une tête de disjoncteur à air

La courbe pointillée représente la limite inférieure d'événement, et la courbe pleine le signal de pression à l'intérieur du réservoir du disjoncteur. On note que vers la 45^{e} milliseconde, il y a chute de pression, mais toujours dans les limites permises par la borne inférieure de limite d'événement. À la 50^{e} milliseconde, la valeur est hors limite pour une première fois. Les quatre mesures subséquentes sont aussi hors limite: à la 55^{e} milliseconde, le MCP génère donc un événement. Il conserve, à partir de la première valeur hors limite, tous les signaux antérieurs pour 50 ms. De plus, il conserve les 250 ms suivantes. Ces informations sont envoyées tour à tour au STD, puis au contrôleur central. Avant de les inclure dans sa base de données, le contrôleur central déterminera au moyen d'outils mathématiques le genre d'événement: dans ce cas-ci, il vérifiera la pente de la dépression pour savoir s'il s'agit d'une ouverture, d'une fermeture ou d'une fermeture/ouverture.

La figure 3.2 donne la hiérarchie des types de signaux présents dans le système.

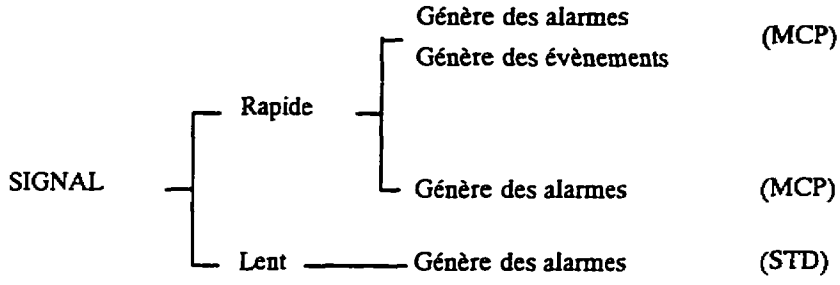


Figure 3.2: Hiérarchie des données dans le système de télésignalisation

CHAPITRE IV

INDICATEURS DE DÉFAUT

L'étude sur les indicateurs de défaut a été effectuée il y a presque deux ans. Depuis ce temps, de nouveaux indicateurs sont disponibles sur le marché et pourront nous amener à reconsidérer les choix que nous avons fait dans une prochaine installation. Cependant, vous trouverez dans ce chapitre les motivations du choix de l'indicateur de défaut sélectionné, ses caractéristiques, ainsi que des considérations sur l'endroit d'implantation des indicateurs dans la seconde phase du projet.

4.1 Choix de l'indicateur de défaut

L'indicateur devrait permettre la télésignalisation de la valeur du contact et du courant circulant dans la ligne. Le rapport RCRSD-91-019(Groupe RCRSD, 1991) donne les caractéristiques nécessaires des indicateurs de défaut souterrains télésignalisés:

> "La séquence doit être initialisée à chaque mise sous tension de la ligne. Elle nécessite donc la détection de la tension sur la ligne. Comme la détection de tension est une technologie non disponible pour l'instant, la détection de courant est une solution alternative qui pourrait s'effectuer avec la valeur de courant la plus basse possible.
>
> L'indicateur de défaut doit être capable de détecter un courant de défaut à la mise sous tension de la ligne. Pour ce faire l'indicateur doit détecter la différence entre les courants de défaut et les courants transitoires. Étant donné qu'aucun appareil sur le marché ne peut faire la différence entre ces deux courants, il fait que l'indicateur soit muni d'une fonction pour éviter les changements d'états causés par des courants transitoires. Donc, pour éviter de détecter des courants transitoires le circuit de détection est bloqué pour une période de 60 secondes qui suit chaque mise sous tension de la ligne.
>
> L'indication de défaut est produite lorsqu'une augmentation brusque de courant est suivie d'une diminution de courant.

L'indicateur doit détecter un défaut inférieur à 600 amps.

L'appareil doit être conçu de façon à permettre la télésignalisation."

Ce document fixe donc les besoins de base quant à la télésignalisation des indicateurs de défaut. Suite à ces besoins, plusieurs indicateurs de défaut ont été évalués. La grille suivante effectue une comparaison de quelques indicateurs de défaut selon la grille d'évaluation d'Hydro-Québec (Daigneault, 1994):

Tableau 4.1: Évaluation d'indicateurs de défaut selon les besoins Hydro-Québec

	Kearney FCI222	**F-Pierce 1547**	**F-Pierce 1514**	**F-Pierce 1541**	**F-Pierce 1546**	**Bardin DDS 180**
Détection du courant lors de la mise sous tension	NON	OUI si I > 1 A	OUI si I>1,2 A	NON	NON	OUI(1)
Détection d'un défaut à la mise sous tension	NON	NON	NON	NON	NON	NON
Indication lorsqu'on a augmentation brusque de courant suivi d'une perte de courant	NON	OUI	NON	NON	NON	OUI
Délai de 60 secondes lors de la mise sous tension	NON	OUI	OUI	NON	NON	NON
Détection des courants de défaut inférieur à 600 A	OUI	OUI	OUI	OUI	OUI	OUI(2)
Conçu de façon à permettre la télésignalisation	OUI	OUI	NON	OUI	NON	OUI
Type d'indication	Affichage lumineux avec câble(3)	Affichage lumineux avec câble(3)	Affichage lumineux (3)	Affichage lumineux avec câble(3)	Affichage lumineux (3)	Relié à un microprocesseur

Les notes apparaissant dans le tableau sont les suivantes:

(1) détection de la tension au secondaire du transformateur;

(2) indication si courant homopolaire est supérieur à 20 A;

(3) nécessite l'installation d'une batterie.

En étudiant les résultats, on remarque que l'indicateur Fisher-Pierce 1547 était celui, au moment des études, qui répondait le mieux aux critères. Il a donc été choisi pour la première phase du projet pilote.

4.2 Caractéristiques de l'indicateur de défaut choisi

L'indicateur 1547 est disponible sous plusieurs versions. Il peut indiquer un défaut à l'aide d'un drapeau mécanique d'une DEL ou d'un émetteur radio; il est disponible en version SCADA, et offre plusieurs autres options. Les tableaux suivants donnent les caractéristiques disponibles des indicateurs:

Tableau 4.2: Différents modèles de base de l'indicateur Fisher-Pierce 1547

Modèle	Mode d'indication de défaut
1547A	Affichage mécanique à l'aide d'un drapeau.
1547B	Affichage à l'aide d'une DEL clignotante. Nécessite l'ajout d'une pile.
1547C	Affichage à distance à l'aide d'une DEL clignotante et d'une fibre optique. Nécessite l'ajout d'une pile.
1547D	Signalisation de défaut au moyen de transmission d'un signal radio et d'une DEL clignotante. Nécessite l'ajout d'une pile.
1547E	Signalisation de défaut au moyen de transmission d'un signal radio. Nécessite l'ajout d'une pile.

Tableau 4.3: Délais de réarmement de l'indicateur Fisher-Pierce 1547

Code	Délai de réarmement de l'indicateur
Indicateurs à mécaniques à drapeaux à réarmement manuel.	
P	Réarmement automatique 24 heures après la fin de l'interruption.
W	Réarmement automatique 4 heures après la fin de l'interruption.

Code	Délai de réarmement de l'indicateur
Z	Réarmement automatique 60 secondes après la fin de l'interruption.
Indicateurs à DEL, affichage à distance ou non	
C	Réarmement automatique 4 heures après la fin de l'interruption et réarmement selon le courant.
E	Réarmement automatique 4 heures après la fin de l'interruption, réarmement selon le courant et réarmement manuel.
M	Réarmement automatique 4 heures après la fin de l'interruption et réarmement manuel.
N	Réarmement automatique 4 heures après la fin de l'interruption et réarmement/armement manuel.
W	Réarmement automatique 4 heures après la fin de l'interruption.
Indicateurs radio	
J	Réarmement automatique 4 heures après la fin de l'interruption et réarmement selon le courant.
K	Réarmement automatique 4 heures après la fin de l'interruption.

Tableau 4.4: Courant de réarmement de l'indicateur Fisher-Pierce 1547

Modèle	Courant de réarmement
M	1.0 A
L	1.5 A
B	2.0 A
D	3.0 A

Tableau 4.5: Sorties SCADA de l'indicateur Fisher-Pierce 1547

Modèle	Indication SCADA
A	Indication normalement ouverte, 3,05 m de câble.
B	Indication normalement fermée, 3,05 m de câble.
N	Aucune indication SCADA.

Il existe en plus diverses versions mécaniques, permettant l'installation avec ou sans "hot-stick".

La version préconisée pour le projet de télésignalisation est celle par indication mécanique, à réarmement automatique 60 secondes après la fin de l'interruption, à courant de réarmement de 1.0 A et en version SCADA à indication normalement ouverte.

Nous ne pouvions penser utiliser les versions nécessitant l'installation de piles, puisqu'au moment du choix, les piles au Lithium disponibles ne garantissaient que 10 ans d'usage normal, ce qui n'est pas acceptable pour Hydro-Québec.

Le contact sec fourni pour SCADA donnera l'indication de l'état de la ligne: la détection du signal est réalisée par la circulation d'un faible courant (20 mA) de basse tension à la fermeture du contact. Ce circuit alimente un coupleur optique pour transmettre l'information au contrôleur local. Pour mesurer le courant, une dérivation est effectuée sur le transformateur de courant de l'indicateur de défaut pour en obtenir la valeur de courant nominal circulant dans le câble; cette mesure, quoique précise à environ 5 %, demeure très valable.

Figure 4.1: Indicateur F-P 1547A monté sur une ligne du poste Guy

Pour s'armer lors de l'alimentation de la ligne, l'indicateur doit voir au moins 1 A sur la ligne surveillée. Le courant maximum supporté est de 800 A. Pendant les premières 60 secondes après une remise sous tension, l'indicateur est bloqué, le temps de laisser passer les perturbations transitoires. Une fois ainsi armé, l'indicateur déclenchera selon les conditions de défaut sont les suivantes:

1. lors d'un défaut, le courant de ligne doit augmenter d'au moins 100 A en moins de 50 ms (3 cycles);
2. le courant total (courant de charge et courant de défaut) doit alors être de 200 A ou plus après cette augmentation;
3. finalement, l'indicateur doit détecter la perte du courant de ligne dans les 40 à 60 secondes qui suivent pour finalement être déclenché.

La figure 4.2 illustre les trois conditions énumérées précédemment.

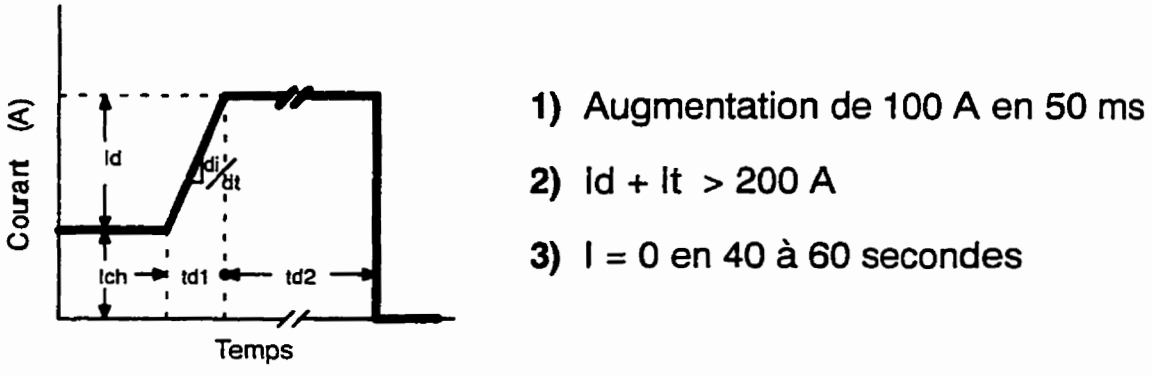


Figure 4.2: Conditions de détection de défaut

4.3 Positionnement des indicateurs de défaut en réseau

Dans la première phase du projet pilote, les indicateurs de défaut ont été installés sur 21 départs d'artères doubles RCL à 25 kV (voir figure 2.1). La seconde phase du projet portera

les indicateurs et contrôleurs locaux à l'extérieur du poste. Il existe deux hypothèses quant à l'architecture que prendra le système de télésignalisation lors de la seconde phase:

1. ***système par fibres optiques***: il s'agitait d'installer des indicateurs de défaut dans un puits de transformation, et de les relier à un module RCP (sections 2.1 et 5.3). Ces modules RCP seraient reliés au MCP au moyen de fibres optiques.

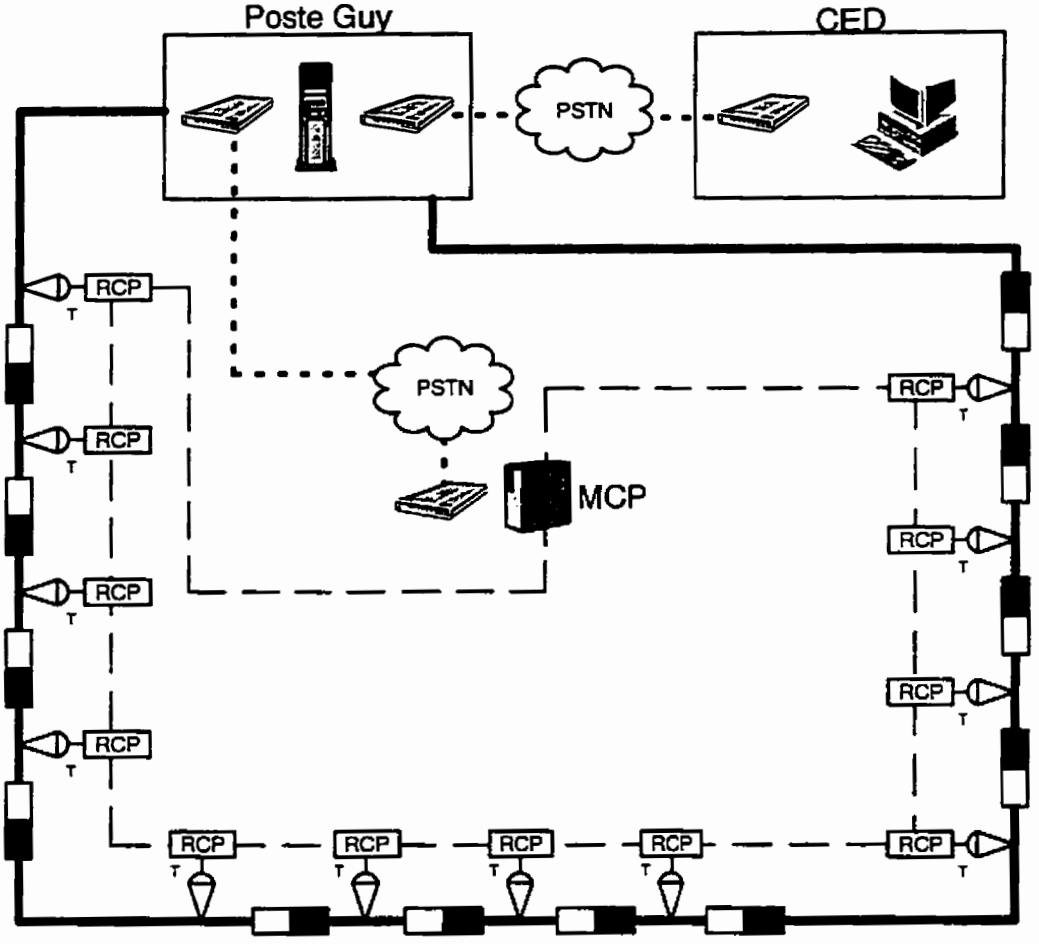


Figure 4.3: Implantation de la seconde phase du projet avec fibres optiques

Pour les figures 4.3 et 4.4, la ligne grasse et continue représente la ligne 25 kV du poste, la ligne hachurée à longs traits représente une connexion fibre optique et la ligne grasse hachurée à petits traits représente un lien téléphonique.

2. ***système radio***: des recherches sont présentement en cours conjointement avec la société Soulé-Bardin S.A. quant à la production d'indicateurs de défaut à radio transmission aptes à la transmission de valeurs de contacts et de courant. En utilisant des indicateurs de défaut capables de transmettre par ondes VHF, les fibres optiques seraient ainsi éliminées.

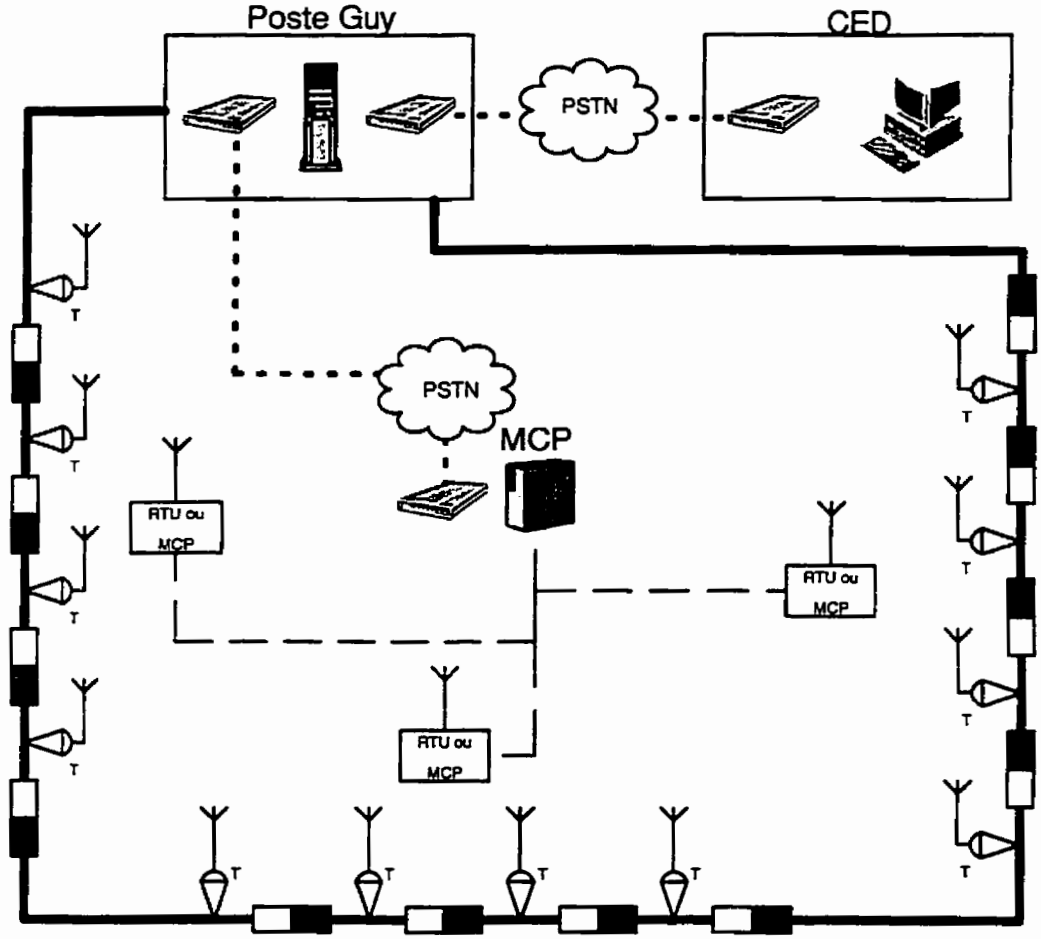


Figure 4.4: Implantation de la seconde phase du projet avec indicateurs radio

4.4 Fiabilité des indicateurs Fisher-Pierce 1547A en réseau

La section mécanique de l'indicateur de défaut Fisher-Pierce 1547A semble fragile aux vibrations et chocs mécaniques. En effet, le ressort servant au mécanisme de drapeau et à la télésignalisation de la valeur du contact à l'aide du contact sec se décroche assez facilement. Avant même la mise en service du système de télésignalisation, un indicateur a dû être retiré pour que son ressort soit replacé adéquatement. De plus, à cause de leur système d'attache en 'U', certains indicateurs se sont détachés des câbles lorsqu'il y avait de fortes vibrations. Des recherches se poursuivent pour trouver un indicateur de défaut plus robuste, ayant les mêmes caractéristiques de détection et un système d'attache plus solide.

CHAPITRE V
CONTRÔLEUR LOCAL

Comme nous l'avons déjà mentionné à la section d'introduction du contrôleur local (section 2.2), le coeur du contrôleur local est constitué de deux composantes: un microprocesseur traditionnel (le STD) et d'un DSP (le MCP). Le microprocesseur est un NEC V53, l'équivalent industriel du i286, et le DSP est un TMS320C26. Le tout communique sur un bus industriel STD32. Cette technologie a été choisie à cause de sa maturité ainsi que sa flexibilité.

Puisque les installations du poste Guy doivent se faire dans les puits d'accès, les contrôleurs locaux ont été placés dans le même cabinet que le contrôleur central, à l'écart des radiations électromagnétiques (EMI) et de l'eau. Cependant, dans d'autres implantations, le contrôleur local doit se trouver près des équipements de haute ou moyenne tension, subissant les rigueurs de l'environnement extérieur.

L'alimentation du contrôleur local peut se faire à partir de plusieurs sources dans le poste: l'alimentation 129 Vcc, 115 Vca ou 230 Vca peuvent être utilisés. Le système est aussi muni d'un système de batteries d'appoint permettant la continuité de service lors d'une interruption temporaire de la source extérieure.

5.1 Master Control Processor: MCP

5.1.1 Description matérielle

La carte MCP, sur laquelle on retrouve un DSP TMS320C26 de Texas Instruments, a été conçue par la compagnie APIX. Ce DSP est supporté par plusieurs composantes, telles que présentées dans le schéma de la figure 5.1.

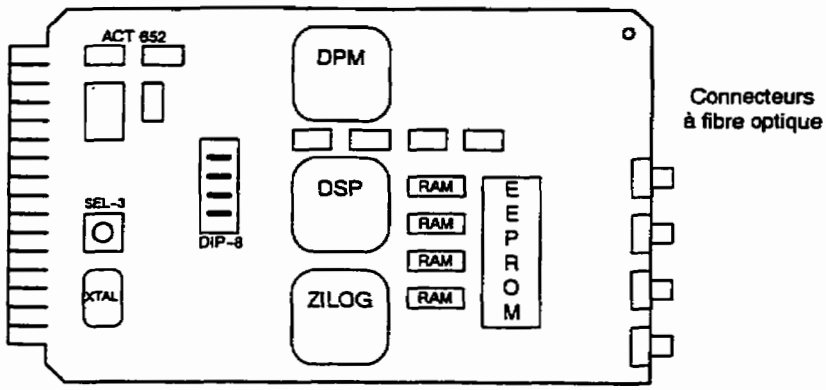


Figure 5.1: Carte MCP - disposition du DSP et de ses composantes

Les composantes de la figure 5.1 sont décrites au tableau 5.1.

Tableau 5.1: Description du matériel de la carte MCP

Objet	Nom	Fabricant	Utilisation
DPM	IDT7133 *"Dual Port Memory"*	Integrated Device Technology	Mémoire RAM à double entrées. Mémoire commune entre le MCP et le STD; elle sert à la communication entre ces deux processeurs.
DSP	TMS320C26BFN	Texas Instruments	UCT du MCP. Contient 1 kilo mot de RAM.
ZILOG	Z16C3010VSC	ZILOG Tech.	Contrôle les communications entre les RCP et le MCP.
RAM	CXK58257	SONY	RAM de 32 kilo mots de données, utilisée comme file circulaire d'événements, et 32 kilo mots de programme.
EEPROM	P28F010	Intel	ROM contenant le programme du DSP, écrit en C et en assembleur TI.
DIP-8	Interrupteurs DIP	--	Sert à l'ajustement de l'adresse mémoire de base de la DPM.
XTAL	Cristal 20 MHz	--	Cristal de 20 MHz synchronisant le tout.
SEL-3	Sélecteur 3 positions	--	Sélecteur d'interruptions pour les interruptions du STD.

Le sélecteur de position sur cette carte permet de sélectionner l'interruption qui sera utilisée pour cadencer le processeur STD.

5.1.2 Description fonctionnelle

L'architecture du MCP est représentée par le diagramme de la figure 5.2.

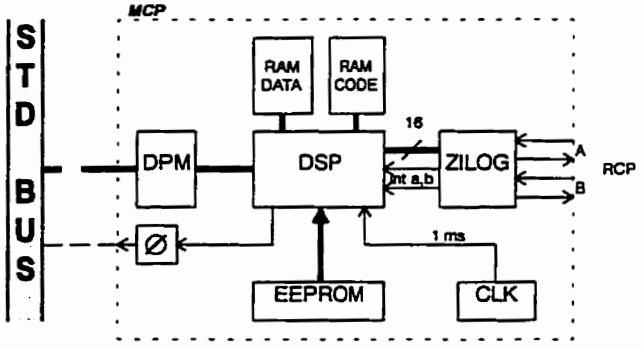


Figure 5.2: Architecture générale du MCP

Le sommaire des tâches du MCP se détaille comme suit:

- chercher l'information des RCP;
- calculer les alarmes à partir des signaux rapides;
- calculer les événements à partir des signaux pouvant générer des événements;
- interrompre le STD aux millisecondes pour le forcer à lire les informations des canaux rapides;
- transférer les données d'événements (s'il y a lieu) au STD;
- avertir le STD (s'il y a lieu) qu'une alarme s'est produite sur un signal rapide.

5.1.3 Description logicielle

Le diagramme de transition d'états du comportement du MCP: le MCP est principalement une machine temps réel à contraintes rigides. À chaque milliseconde, il *doit* lire tous les canaux, effectuer les calculs nécessaires sur ces signaux et cadencer le processeur qui vient chercher cette information (le STD) sans retard sous peine de pertes de données d'où la

défaillance du système. La figure 5.3 décrit le diagramme de transition d'états du processeur MCP.

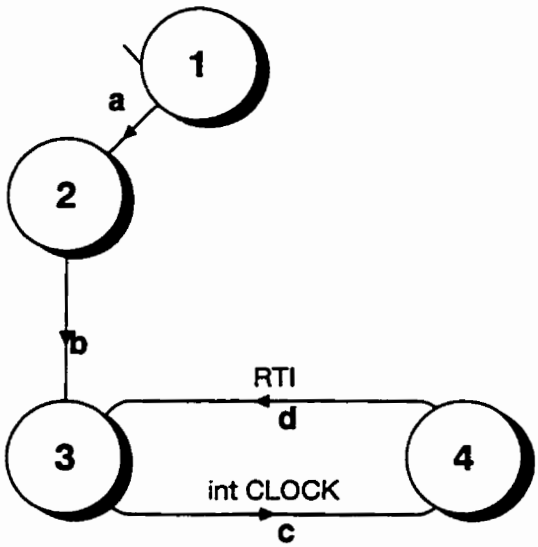


Figure 5.3: Diagramme de transition d'états du processeur MCP

Tableau 5.2: Description du diagramme d'états du MCP de la figure 5.3

Etat	Transition	Description
1	-	**État initial.** Initialisation du système des communications avec les RCP.
	a	L'initialisation est terminée, on passe à l'état suivant.
2	-	**État d'attente.** Le MCP attend trois secondes que le STD ait terminé d'initialiser la DPM.
	b	Les trois secondes sont écoulées, on passe à l'état suivant.
3	-	**État stable.** Le processeur attend qu'une interruption survienne pour agir.
	c	Une interruption interne est survenue.
4	-	État de traitement de l'interruption interne. Voir la section suivante.
	d	L'interruption a été traitée.

Lorsqu'une interruption d'horloge survient dans le MCP, le logiciel principal du MCP agit tel que montré à la figure 5.4 de la page suivante.

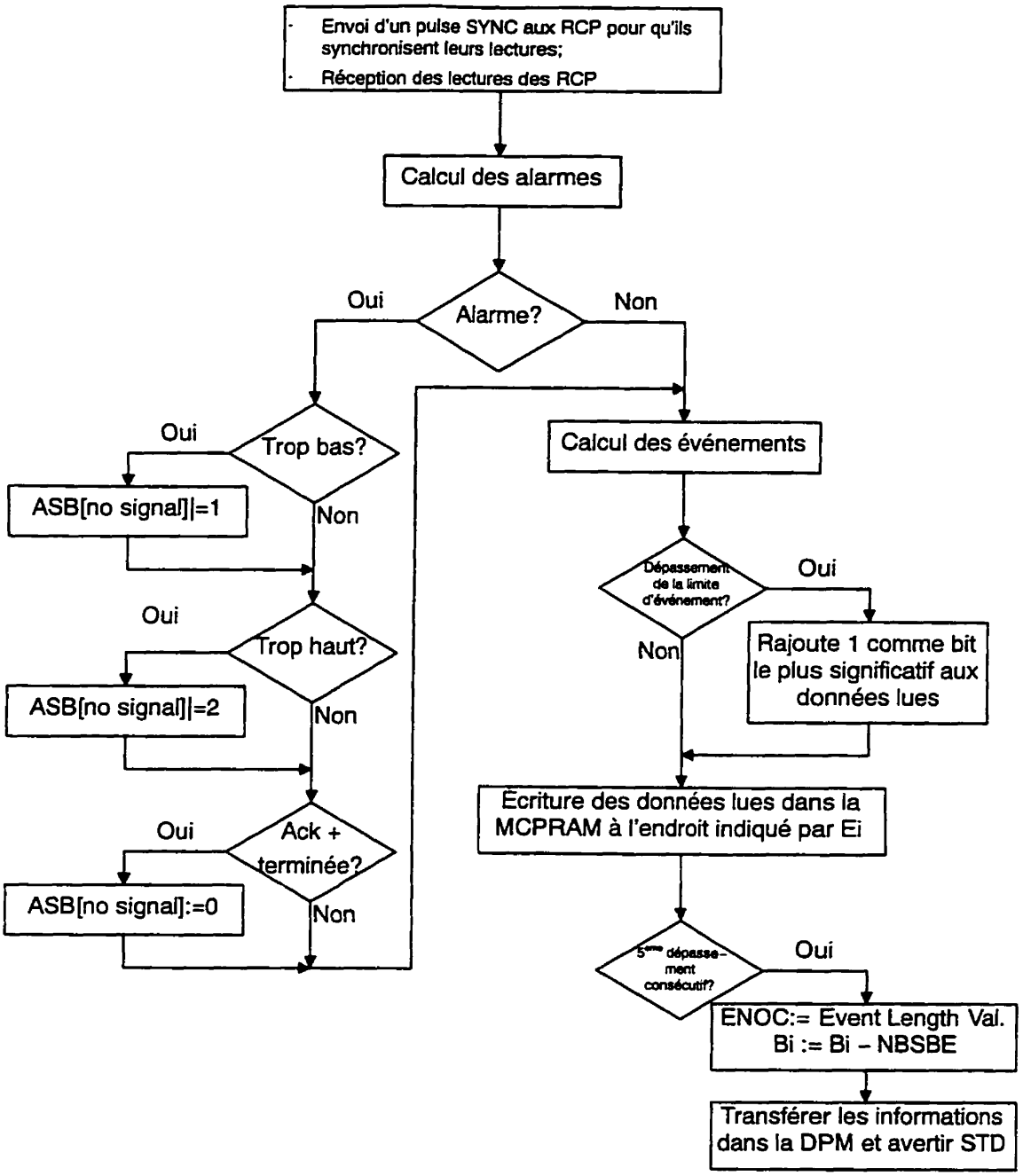


Figure 5.4: Gestion de l'interruption interne du processeur MCP

5.2 Interface STD

5.2.1 Description matérielle

La carte STD (figure 5.5) a été conçue par la compagnie Ziatech. Cette carte, la ZT89CT01, est en fait une carte contenant un CPU compatible 80286 plus performant que celui produit par Intel, supporté d'un système de mémoires et de différents périphériques. Ce processeur, le NEC V53, roule à 12.5 MHz. On peut compter dans ses périphériques six ports parallèles, deux ports série fibre optique avec accès DMA(16552), un contrôleur d'interruptions (8259), des compteurs 16 bits (8254), une DEL programmable.

Le code contenu dans le processeur a été écrit en C, avec Borland C++ 3.1. Le tout est contenu dans un EEPROM Intel.

5.2.2 Description fonctionnelle

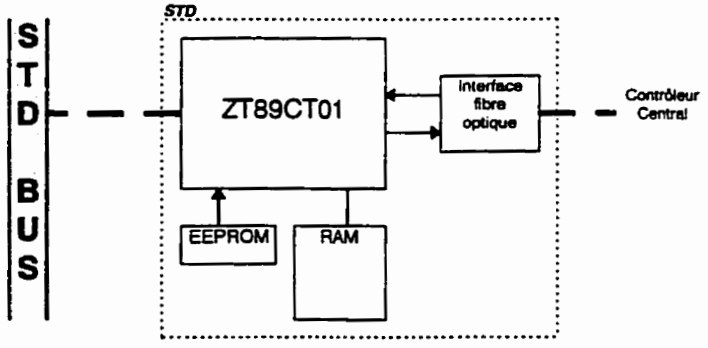


Figure 5.5: Architecture globale du STD

Ses tâches sont les suivantes:

- agir comme serveur de commandes pour les requêtes en provenance du contrôleur central: des requêtes en provenance du contrôleur central arrivent environ toutes les 20 secondes et en provenance de l'interface usager (via le contrôleur central) sui-

vant des temps aléatoires. Le contrôleur central demande périodiquement au STD de lui fournir des informations pour qu'il puisse mettre à jour sa base de données. L'interface usager peut demander au STD des informations directement, sans avoir à faire de requêtes dans la base de données; par exemple, on peut demander la lecture actuelle des capteurs, pour la faire afficher à l'usager;

- aller chercher aux millisecondes les signaux rapides et les stocker en mémoire. On effectue la moyenne de ces signaux pour la dernière heure et pour la dernière minute. On conserve, de plus, les valeurs minimales et maximales pour ces mêmes périodes de temps. Le STD n'est pas absolument obligé de lire et calculer ces informations s'il est occupé à traiter un événement.
- aller chercher lorsque le temps le permet les signaux lents;
- aller chercher les données d'événements et d'alarmes de signaux rapides et les stocker en mémoire;
- détecter et stocker les alarmes pour les signaux lents.

5.2.3 Description logicielle

Le STD est en fait un système temps réel à contraintes douces. Il peut accumuler un certain retard dans le traitement des événements grâce à une file d'attente. Trois sortes d'interruptions peuvent survenir (elles sont en ordre de priorité décroissante) lors de l'exécution normale du système: l'interruption APIX, l'interruption de communications et l'interruption d'horloge. Le tableau 5.3 résume l'utilité des interruptions présentes dans le système.

Tableau 5.3: Description des interruptions affectant le STD

Priorité	Nom	Description	Inter-valle
1	APIX	Cette interruption est générée par le processeur sur la carte APIX (le MCP) pour forcer le STD à lire les canaux rapides.	1 ms
2	COMM	Cette interruption est générée par le système de communications sur la carte STD pour avertir d'un débordement probable du tampon de réception.	au besoin
3	CLOCK	Cette interruption provient de l'horloge interne du processeur STD lui indiquant qu'il est temps d'effectuer certaines manoeuvres.	1 s

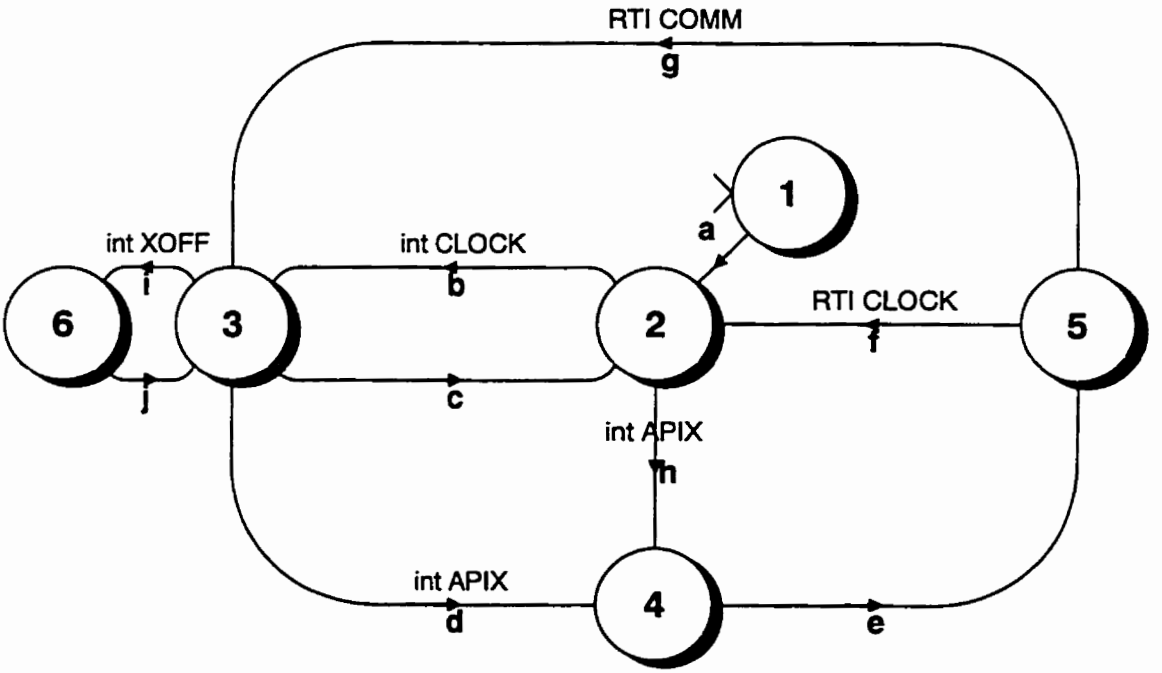


Figure 5.6: Diagramme de transition d'états du processeur STD

Tableau 5.4: Description du diagramme d'états du STD de la figure 5.6

Etats	Transitions	Description
1		**État initial.** La mémoire "Dual Port" est initialisée par le STD.
	a	Lorsque la mémoire DPM a été initialisée des valeurs limites d'alarmes et d'événement, le STD passe dans l'état de communications.
2		**État de communications.** Le STD attend d'avoir des requêtes du contrôleur central pour y répondre.
	b	Le STD reçoit une interruption d'horloge interne à *toutes les secondes*. Cette interruption ne peut être interrompue que par une interruption APIX.
	h	Le STD reçoit une interruption prioritaire du processeur APIX *toutes les millisecondes*. Cet état ne peut être interrompu.
3		**État de mise à jour des canaux lents: alarmes, moyennes et "watchdog".**
	c	L'interruption de l'horloge est terminée, retour à l'état des communications. Si les communications sont suspendues à cause d'un XOFF, on envoie un XON.
	d	Le STD reçoit une interruption prioritaire du processeur APIX.
	I	Le STD reçoit une interruption du système de communications pour l'informer que le tampon de réception (recevant des données du contrôleur central) est a moitié plein.
4		**Sauvegarde de l'état avant l'interruption APIX.** Comme l'interruption peut survenir lors de l'état de communications ou pendant le traitement de l'interruption de l'horloge, on sauvegarde l'état lors de l'interruption pour pouvoir revenir à l'état préinterruption après l'interruption APIX.
	e	Une fois l'état initial sauvegardé, le STD passe au traitement de l'interruption APIX.
5		**Traitement de l'interruption APIX.**
	f	L'interruption a été traitée, et comme le STD procédait aux communications avant l'interruption, on retourne à l'état des communications.
	g	L'interruption a été traitée, et comme le STD traitait l'interruption de l'horloge avant l'interruption APIX, on continue de traiter l'interruption de l'horloge.

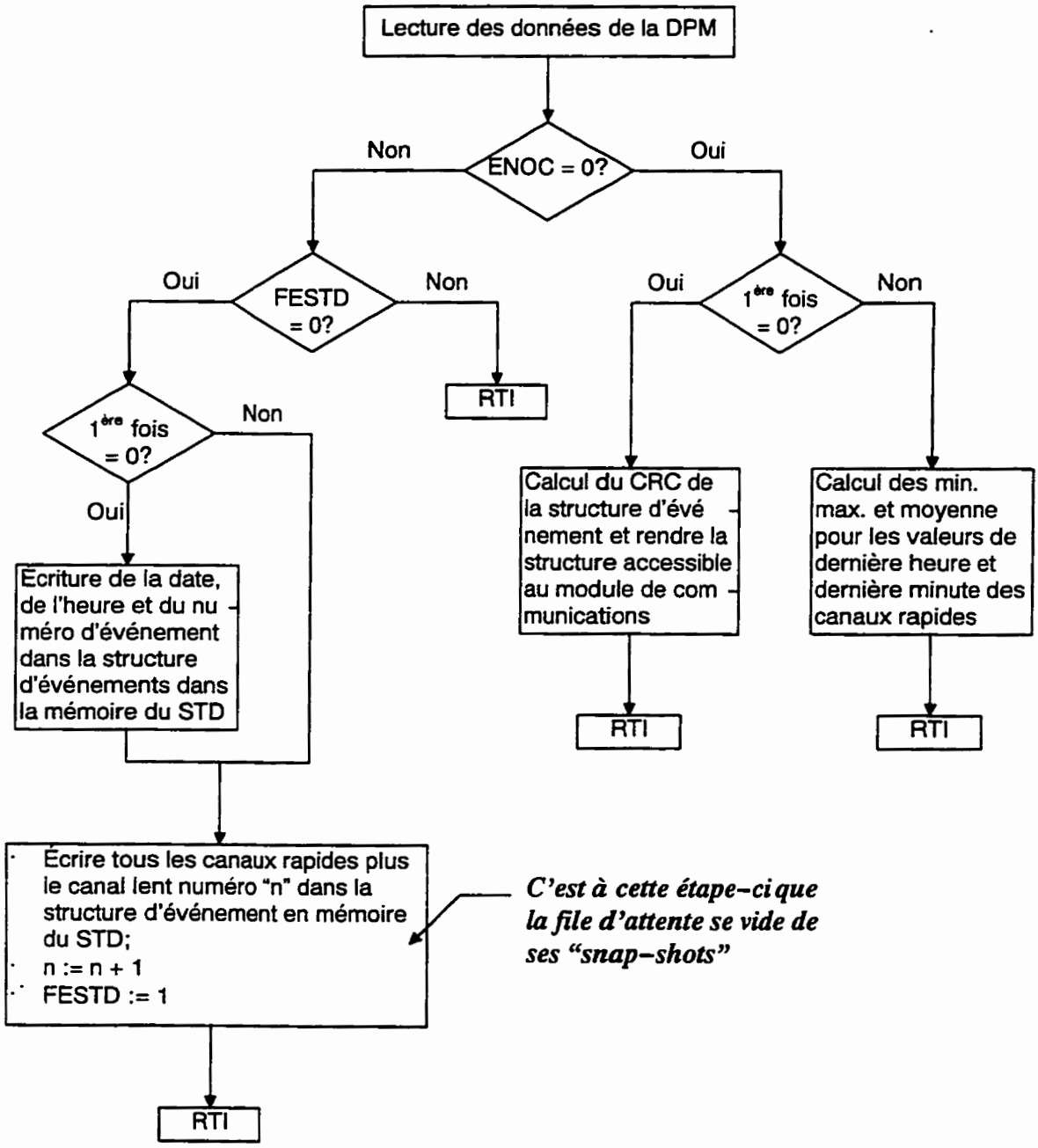
Etats	Transitions	Description
6		**Demande d'envoi de XOFF au contrôleur central.** Le tampon de réception de 16 octets est à moitié plein. Le système de communications en averti le STD. Le STD envoie un XOFF au contrôleur central.
	j	Le XOFF est envoyé, on continue le traitement régulier de l'interruption de l'horloge à l'endroit où on a été interrompu.

Pendant cette interruption générée à toutes les secondes, le STD effectue les tâches suivantes:

- lecture des canaux lents et vérification des limites d'alarmes. Si une alarme est présente, la stocker dans la RAM du STD. La valeur sera éventuellement demandée par le contrôleur central;
- à partir de la lecture des canaux lents, calculer les valeurs minimum, maximum et moyennes de tous les signaux lents pour la dernière minute et pour la dernière heure;
- effectuer un "RESET" du "watchdog";
- si une interruption survient du système de communications, envoyer un caractère XOFF au contrôleur central pour lui demander d'arrêter le flot de communications temporairement, afin de ne perdre aucun caractère.

5.2.4 Traitement de l'interruption du STD par le MCP

Lorsque le STD est interrompu par le MCP, il effectue les tâches décrites par le diagramme de flux de données de la figure 5.7.



Rappelons que la variable ENOC est le compteur d'échantillon de tous les signaux d'une milliseconde contenus dans l'événement, et que FESTD est le fanion de synchronisation de lecture d'un groupe d'échantillons dans la DPM.

Figure 5.7: Gestion de l'interruption APIX du processeur STD

Le diagramme 5.7 peut se résumer ainsi:

- si un événement est présent, lire un groupe d'échantillons de l'événement et indiquer au MCP que la lecture a été effectuée. En plus, si c'est la première lecture effectuée, enregistrer l'heure et la date de l'événement, et y assigner un numéro séquentiel;
- si aucun événement est présent, vérifier si c'est la première fois qu'un événement termine. Si oui, calculer le CRC de la structure d'événement en attente et rendre l'événement disponible au module de communications. Sinon, calculer à partir des valeurs de signaux rapides la moyenne, le maximum et le minimum des valeurs de la dernière minute et de la dernière heure.

5.3 Remote Control Processor: RCP

5.3.1 Description matérielle, fonctionnelle et logicielle

La construction des RCP est très similaire à celle du MCP. Celle-ci est illustrée à la figure 5.8.

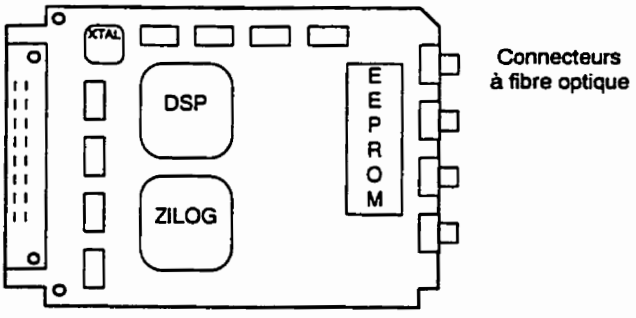


Figure 5.8: Carte RCP – disposition du processeur et de ses composantes

Ses tâches sont les suivantes:

- prendre des lectures des différents capteurs;
- envoyer ces lectures au processeur maître à sa demande.

Le DSP, le cristal, le contrôleur ZILOG et le EEPROM sont identiques en tout point à ceux utilisés pour le MCP. Les RCP sont installés sur une carte mère, contenant aussi tous les modules d'acquisition utilisés par le RCP. Cette carte mère peut contenir 8, 16 ou 32 modules d'acquisition. Présentement, six types de modules d'acquisition peuvent être utilisés. Les types de modules et leur description sont présentés au tableau 5.5.

Tableau 5.5: Modules d'acquisition

Type	Description
RIN–1A	Entrée de résistance
MAI–1B	Entrée de courant
DIN–1A	Entrée digitale
VIN–3B	Entrée de tension
HV–PB	Entrée pour fibre optique
Dout–1A	Sortie digitale

5.3.2 Organisation de la mémoire à double ports DPM

Le transfert d'information entre le STD et le MCP se fait grâce à 2 kilo mots de mémoire à deux ports ("Dual Port Memory") fabriqué par Integrated Device Technology. Ce circuit intégré, le IDT7133S/L, gère les accès simultanés à la mémoire, sans à avoir à utiliser de sémaphores logiciels. Les processeurs peuvent utiliser des adresses d'entrée/sortie pour accéder cette mémoire. L'adresse de base de la mémoire en question peut être définie par des "DIP Switches" situés sur la même carte que la mémoire, telle que défini dans la figure 5.9.

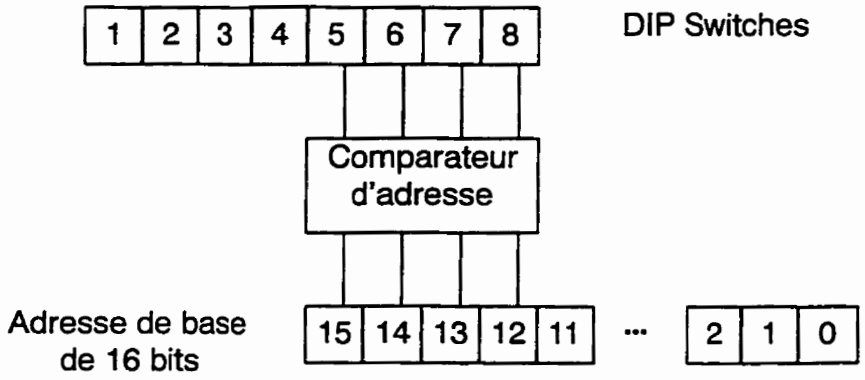


Figure 5.9: Utilisation des "DIP Switches" pour la mémoire à 2 ports

Le tableau 5.6 donne l'organisation de la mémoire DPM. Les noms et déplacements par rapport à l'adresse de base sont donnés pour chaque bloc de 128 mots. Les contraintes de lecture/écriture (colonnes READ/WRITE) ne sont données qu'à titre indicatif de la direction du flux de données puisque aucune contrainte n'est imposée par la DPM.

Le temps nécessaire au STD pour lire un mot de la DPM est d'environ 8 µs (cette valeur a été déterminée de façon empirique).

Tableau 5.6: Contenu de la mémoire DPM

No	Adresse de base DPM	Nom abrégé	Nom du bloc	WRITE	READ	Valeur initiale
0	0_h	LSB	Latest Sample Buffer	MCP	STD	0
1	100_h	DTBE	Data Transfer Buffer for Events	MCP	STD	0
2	200_h	SCRAP	Scrap Buffer (utilisé lorsque la communication est coupée)	MCP	STD	0
3	300_h	ASB	Actual Alarm Status Buffer	MCP	STD	0
4	400_h	AAB	Alarm Acknowledge Buffer	STD	MCP	0
5	500_h	ODB	Output Data Buffer	STD	MCP	0
6	600_h	×	×	×		0
7	700_h	×	×	×	×	0

No	Adresse de base DPM	Nom abrégé	Nom du bloc	WRITE	READ	Valeur initiale
8	800_h	×	×	×	×	0
9	900_h	LPMTO	Logical to Physical Mapping Table for Output Channels	STD	MCP	i (1–1)
10	$A00_h$	ALVTL	Limit Values Table for High Speed Alarms – Lower Limits	STD	MCP	0
11	$B00_h$	ALVTU	Limit Values Table for High Speed Alarms – Upper Limits	STD	MCP	$0FFF_h$
12	$C00_h$	ECLL	Event Condition List – Lower Limits	STD	MCP	0
13	$D00_h$	ECLU	Event Condition List – Upper Limits	STD	MCP	$0FFF_h$
14	$E00_h$	LPMT	Logical to Physical Mapping Table	STD	MCP	i (1–1)
15	$F00_h$	GPB	General Purpose Buffer (Mots de contrôle du STD)	STD	MCP	0
15a	$F80_h$	GPB	General Purpose Buffer (Mots de contrôle du MCP)	MCP	STD	0

De plus, les blocs du GPB (région $F80_h$ de la DPM) sont définis au tableau 5.7.

Tableau 5.7: Contenu de la mémoire DPM, blocs GPB

No	offset GPB	Adresse DPM	Nom abrégé	Nom du bloc	WRITE	READ
0	0	$F00_h$	NBC	Number of Channels	STD	MCP
1	2	$F02_h$	NBETC	Number of Event Triggering Channels	STD	MCP
2	4	$F04_h$	NBFC	Number of Fast Channels	STD	MCP
3	6	$F06_h$	EVENT–LENGTH	Length of Event	STD	MCP

No	offset GPB	Adresse DPM	Nom abrégé	Nom du bloc	WRITE	READ
4	8	$F08_h$	NBSBE	Number of Samples before Event	STD	MCP
5	10	FOA_h	FESTD	Flag Event Buffer Ready	STD MCP	MCP STD
9	12	FOC_h	NBOC	Number of Output Channels	STD	MCP
7	14	FOE_h	DPMSTD	DPM Initialized by STD and read by MCP	STD	MCP
8	16	$F10_h$	BAD SENSOR	Sensor Failure Detected By MCP	MCP	STD
9	128	$F80_h$	ENOC	Event Counter	MCP	STD
10	130	$F82_h$	COMMF	Communication Failure between MCP & RCP	MCP	STD
11	132	$F84_h$	×	×	×	×
12	134	$F86_h$	OVRUN-MCP	Overrun Error	STD	MCP
13	136	$F88_h$	×	×	×	×
14	138	$F8A_h$	EVENT	Event Status	MCP	MCP
15	140	$F8C_h$	Ei	Event Queue Head	MCP	MCP STD
16	142	$F8E_h$	Bi	Event Queue Tail	MCP	STD
17	144	$F90_h$	BiM		MCP	STD

Les GPB contiennent des constantes qui seront utilisées dans la description de l'opération du système.

5.3.3 Combinaison des deux machines, le STD et MCP

Comme nous avons pu le constater, nous sommes en présence d'un système temps réel hybride, c'est-à-dire un système à la fois à contraintes rigides (MCP et RCP) et à la fois à

contraintes souples (MCP, STD et file d'attente). Le diagramme 5.10 nous donne un portrait global de l'architecture.

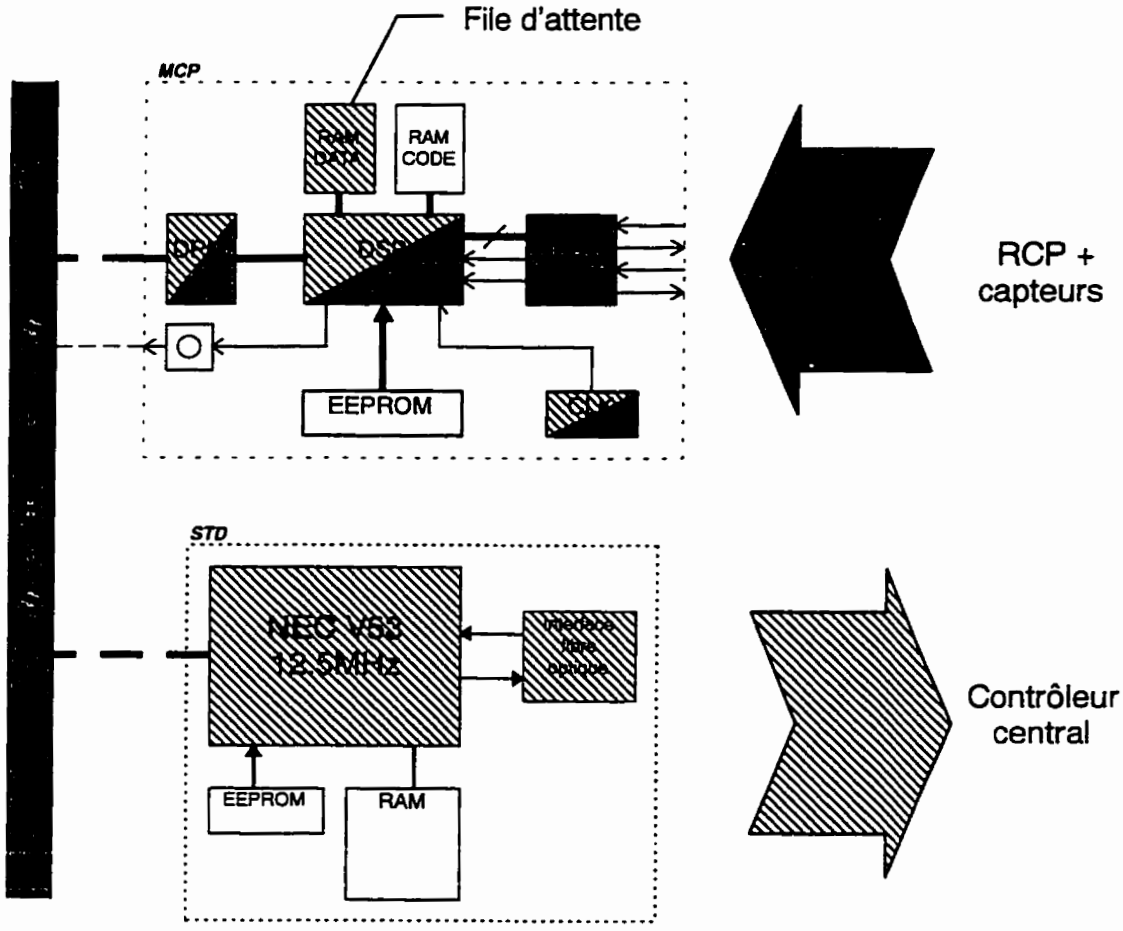


Figure 5.10: Combinaison du STD et MCP

Les zones hachurées sont les sections qui ont trait à la partie temps réel à contraintes souples, et les zones pleines sont celles qui sont utilisées par le système temps réel à contraintes rigides.

CHAPITRE VI
CONTRÔLEUR CENTRAL

Le contrôleur central est un système flexible et évolutif. Il a été conçu de façon à pouvoir s'adapter à la progression de la technologie, et peut être adapté à une foule d'environnements de surveillance. C'est aussi un système a sécurité relative (*"failsafe"*): il a été muni d'une unité d'alimentation auxiliaire, et d'un deuxième disque rigide contenant une copie intégrale du disque en cours d'opération. Finalement, il a été conçu de façon à être simple d'utilisation: les sections qui suivent, expliquant le fonctionnement du logiciel, en font foi.

6.1 Logiciels du contrôleur central

Le système de logiciels du contrôleur repose d'abord sur le système d'exploitation Linux. Puisque les tâches à accomplir sont asynchrones, ce système multitâches, supportant la multiprogrammation a été envisagé. Linux offre de nombreux avantages, dont:

- ***la disponibilité de son code source***: ceci permet de rajouter facilement dans le noyau du système d'exploitation le support pour de nouveaux périphériques. Le contrôleur du poste Guy utilise une carte multiport, permettant de brancher plusieurs dizaines de liens série sur l'ordinateur. Or, cette carte nécessite des pilotes qui tiennent dans le noyau même du système d'exploitation; sans la disponibilité du code et la possibilité de le recompiler, cette carte n'aurait pu être utilisée;
- ***le nombre de pilotes de périphériques et de logiciels disponibles***: Linux est un produit de la génération Internet. Sa diffusion à travers le monde a permis la collaboration de plusieurs individus, afin de créer un logiciel de qualité supportant à peu près

toutes les configurations possibles. Ces programmeurs y travaillent bénévolement, et distribuent gratuitement leurs logiciels;

- ***la stabilité du système***: malgré son jeune âge, Linux est maintenant un produit stable. Son bassin d'utilisateurs permet la découverte rapide des problèmes majeurs;
- ***la compatibilité***: Linux incorpore la norme POSIX du développement des systèmes UNIX. Il supporte à la fois les systèmes SVR4 et BSD. Un code C créé pour SunOS™ ou SCO UNIX™ fonctionne dans plusieurs des cas sans aucune modification majeure;
- ***un système évolutif***: plusieurs programmeurs sont présentement à modifier Linux pour qu'il puisse utiliser le plein potentiel des machines de demain. Avec une architecture à micro-noyau (*microkernel*) et un support pour les ordinateurs à multiprocesseurs, Linux nous permettra lors de l'apparition de nouvelles technologies de changer le contrôleur central pour une machine plus performante, sans qu'aucune modification soit apportée au parc de logiciels existants.

La version 1.2.8 de Linux a été choisie pour sa disponibilité et pour sa stabilité. De plus, l'architecture d'exécutable ELF à été choisie au profit de celle `"a.out"`: celle-ci permet la compatibilité directe avec les binaires en provenance de plusieurs systèmes d'exploitation dont SCO UNIX.

La section 2.3 a fait le tour des principaux autres logiciels qui ont du être implantés pour faire fonctionner le système de télésignalisation des indicateurs de défaut. Nous représenterons dans le tableau 6.1 ces logiciels en explicitant leur utilisation et le nom qui leur a été attribué. Le tableau contient aussi les autres fichiers et programmes utiles au fonctionnement de ces trois logiciels principaux.

Tableau 6.1: Fichiers du contrôleur local

Nom	Catégorie	Utilité
AE_comdisp	répartiteur	Le répartiteur de messages. Répartit les messages aux logiciels et périphériques destinataires.
interpcr	interpréteur	Interprète les requêtes d'usagers. Utilisé pour les communications série.
interpcr2	interpréteur	Interprète les requêtes d'usagers. Utilisé pour les communications ethernet.
inter_sync	vérificateur	Assure le synchronisme du contenu de la base de données et les contrôleurs locaux.
AE_watch_process	utilitaire	Logiciel de chien de garde qui s'assure du bon fonctionnement des quatre précédent logiciels et les repart au besoin.
AE_stop_watching	utilitaire	Logiciel qui tue le chien de garde, sans pour autant arrêter les logiciels principaux.
kill_CC	utilitaire	Logiciel qui permet d'arrêter les principaux logiciels.
/tmp/AE_error.log0	journal	Journal des erreurs de fonctionnement du logiciel AE_comdisp.
/tmp/AE_error.log1	journal	Journal des erreurs de fonctionnement du logiciel inter_sync.
/tmp/AE_error.log2	journal	Journal des erreurs de fonctionnement du logiciel interpcr et interpcr2.
port.dat	configuration	Fichier de configuration contenant les différents modes de communication permettant de leurs paramètres de fonctionnement.
fetch	utilitaire	Point d'accès à la base de données. Permet d'effectuer des requêtes SQL directement à la ligne de commande, pour manipuler et examiner la base de données.
lccom	utilitaire	Utilitaire de communication. Ce logiciel incorpore le protocole de communication du système de surveillance et permet de déverminer et d'interroger directement les différents contrôleurs locaux.

6.2 Maintenance et dépannage

La maintenance du contrôleur local, quoique simple, requiert quelques connaissances sur l'utilisation d'un système UNIX; l'usager devra être en mesure de naviguer à travers les structures de répertoires, d'exécuter des programmes et d'effectuer une certaine gestion des processus.

La première étape à effectuer lorsque le système ne semble fonctionner correctement consiste à ouvrir une session sous le nom de "dbadmin" sur le contrôleur, soit par accès direct ou par la console d'accès modem. Le diagramme 6.1 donne ensuite la procédure à suivre pour remédier au problème; l'état A est l'état initial. Le tableau 6.2 explique les étapes du diagramme de résolution de problème de la figure 6.1.

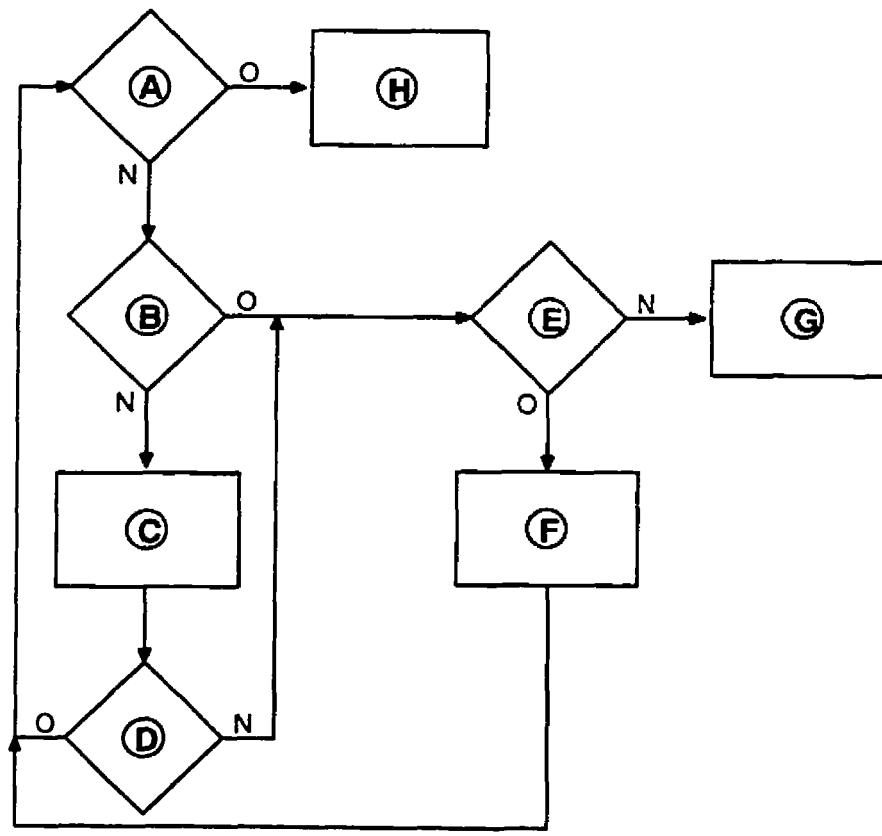


Figure 6.1: Déverminage du contrôleur central

Tableau 6.2: Déverminage du contrôleur central

Objet	Description
A	Est-ce que tout fonctionne?
B	Est-ce que tous les logiciels sont en mémoire et fonctionnels? Pour le savoir, effectuer la commande '`ps -x`' et vérifier s'ils sont présents dans la liste des processus.

C	Repartir les logiciels qui ne fonctionnent pas. Les commandes suivantes permettent de les repartir:
	`AE_comdisp &> /tmp/AE_error.log0 &`
	`inter_sync &> /tmp/AE_error.log1 &`
	`interpcr &> /tmp/AE_error.log2 &`

D	Sont-ils repartis? Vérifier à l'aide de '`ps -x`'.
E	Est-ce que le disque est plein? La commande '`df`' permet d'évaluer le taux d'utilisation des disques.
F	Effacer tout le contenu du répertoire '`/tmp`', et revérifier le taux d'utilisation du disque. Effacer de nouveaux fichiers s'il ne reste pas au moins 10 Mb de libre.
G	Vérifier le fichier si le `port.dat` n'a pas été détruit ou modifié. Si oui, utiliser la copie de sauvegarde. Vérifier aussi les fichiers de journal d'erreurs de chacun des logiciels. Ils peuvent indiquer les causes su mauvais fonctionnement. En dernier recours, repartez le contrôleur central et réinitialisez les contrôleurs locaux à l'aide du logiciel lccom [PV94].

6.3 Implantation du protocole de communications

La compréhension des bases du protocole de communications permet entre autres l'utilisation du logiciel `lccom`, servant au déverminage des contrôleurs locaux. Elle pourra aussi servir aux développeurs, afin de mettre à jour les logiciels des contrôleurs locaux et centraux et de l'interface usager, puisque chacun de ces logiciels utilisent ce protocole.

Le protocole a été conçu pour permettre à un programmeur d'en déchiffrer facilement le contenu. L'approche texte a été retenue plutôt que l'approche binaire: lorsque le program-

meur voit le message, il voit des chaînes de caractères intelligibles, au lieu de voir des informations binaires.

Le protocole de communications de la figure 6.2 est très similaire au modèle de communications OSI de l'ISO (Couloris et Dollimore, 1994).

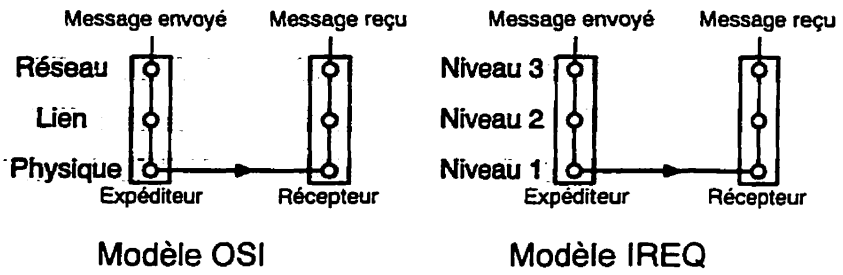


Figure 6.2: Comparaison de l'architecture du protocole avec le protocole OSI

Pour qu'un message soit transmis dans les deux modèles, il doit passer par le niveau réseau, le niveau lien et le niveau physique dans cet ordre avant d'être envoyé sur le support physique de communications. Le message doit effectuer le chemin inverse avant de parvenir au destinataire.

Le niveau 1 et le niveau physique implantent les primitives de communications de bas niveau. Ils permettent l'envoi de bits sur le lien série ou ethernet.

Le niveau 2 et le niveau lien s'occupent de transmettre l'information entre deux ordinateurs directement reliés. Ils s'assurent que le message s'est bien rendu, et ce de façon intacte.

Le niveau 3 et le niveau réseau ont la même fonction: transmettre un message (un "paquet"), d'un ordinateur à un autre. Cependant, contrairement aux protocoles implantés par les niveaux réseau habituels tel X.25 et IP, le niveau 3 n'est utilisable qu'en mode maître/esclave. Ce niveau est responsable du routage de l'information au bon endroit. Il contient entre autres, comme l'illustre la figure 6.3, le nom du destinataire et de l'expéditeur.

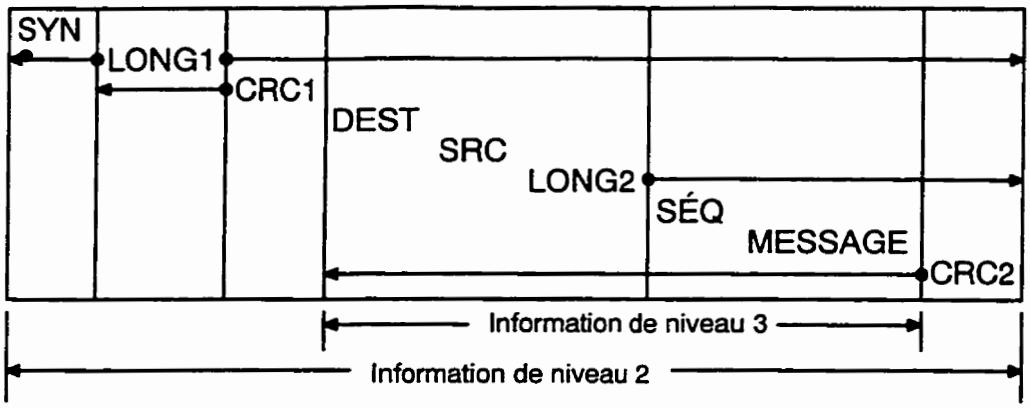


Figure 6.3: Niveau 1 du protocole de communications

Tableau 6.3: Explication du niveau 1 du protocole de communications

Nom	Taille (car.)	Description
SYN	1	Caractère de synchronisation (^V). Permet aux interlocuteurs de synchroniser leurs transmissions.
LONG1	8	Longueur du message en entier, sous forme hexadécimale.
CRC1	4	Nombre du contrôle cyclique par redondance (*"CRC-16"*) sur la taille totale du message sous forme hexadécimale.
DEST	8	Destinataire du message. Cette valeur est utilisée par le répartiteur.
SRC	8	Source du message. Cette valeur est utilisée par le répartiteur.
LONG2	8	Longueur du numéro de séquence et du message, sous forme hexadécimale
SÉQ	2	Numéro de séquence. Compteur séquentiel servant à valider la réponse: la réponse à une requête doit contenir le même numéro. Ce compteur est incrémenté à chaque nouvel envoi de message.
MESSAGE	N.D.	Le message en tant que tel. Le format de message est donné par le texte [Ma95].
CRC2	4	Nombre du contrôle cyclique par redondance sur la section ayant trait au niveau 3 des communications.

Plusieurs messages de niveau 3 sont disponibles pour la gestion du système de télésignalisation. Nous nous limiterons ici à les énumérer et à expliquer brièvement leur utilité. Le détail de l'implantation se retrouve à la référence (Pater et Valliquette, 1994).

Tableau 6.4: Messages de niveau 3 du protocole de communications

Nom	Description	PC↔CC	PC↔LC	CC↔LC
ADDUSER	Ajoute un usager à la banque de données.	✓		
ANSW	Réponse à une requête.	✓	✓	
DELUSER	Efface un usager de la banque de données.	✓		
GETAC	Lecture des limites d'alarmes.	✓		
GETAE	Lecture des alarmes en cours sur le système. Les alarmes qui ont été désactivées n'apparaîtrons pas.	✓		
GETAED	Lecture des alarmes possibles sur le système.	✓		
GETAS	Lecture de toutes les alarmes en cours sur le système.	✓		
GETB	Lecture des noms et numéro de tous les contrôleurs locaux du système.	✓		
GETCAL	Lecture de la calibration des capteurs.	✓		
GETDATE	Lecture de la date inscrite dans le système hôte.	✓	✓	✓
GETDIARY	Lecture du journal de bord du système.	✓		
GETEGS	Demande de données d'événement.	✓		
GETEL	Lecture de la liste d'événements ayant eu lieu sur le système.	✓		
GETH	Demande de l'historique du système ou d'un contrôleur local.	✓		
GETSIGN	Lecture de la liste des signaux.	✓		
GETSOCC	Lecture des valeurs instantannées des différents senseurs.	✓		
GETTGS	Lecture de données de tendance.	✓		

Nom	Description	PC↔CC	PC↔LC	CC↔LC
GETTGSA	Lecture des données de tendance et des dates associées à ces données.	✓		
GETUSER	Lecture de la liste des usagers.	✓		
LOGIN	Requête de branchement.	✓		
LOGOUT	Requête de fin de connexion.	✓		
PUTA	Demande d'accusé de réception pour une alarme.	✓		
PUTAC	Requête d'ajustement des niveaux d'alarmes.	✓		
PUTAED	Requête d'activation ou de désactivation d'une alarme.	✓		
PUTC	Requête d'ajout de commentaire dans l'historique.	✓		
PUTCAL	Requête de modification de la calibration d'un capteur.	✓		
SETDATE	Requête d'ajustement de la date sur le contrôleur local et le contrôleur central.	✓	✓	✓
GETSOLM	Lecture des valeurs minimales, maximales et moyennes de chaque capteur pour la dernière minute.		✓	✓
GETALLC	Lecture des alarmes en cours sur un contrôleur central en particulier.			✓
GETEVALP	Lecture sur un contrôleur local des derniers événements et dernières alarmes.			✓
GETEVFSA	Lecture des informations sur les canaux rapides d'un contrôleur local.			✓
GETINILC	Lecture de la configuration initiale des capteurs après un redémarrage.			✓
GETSOLH	Lecture des valeurs minimales, maximales et moyennes de chaque capteur pour la dernière heure.			✓

Nom	Description	PC↔CC	PC↔LC	CC↔LC
PUTALLC	Requête d'ajustement des paramètres d'alarme sur un contrôleur local.			✓
PUTINILC	Requête d'ajustement des paramètres d'initialisation d'un contrôleur local.			✓

CHAPITRE VII
INTERFACE USAGER

L'interface usager présente le résultat final de tout le système: l'usager ne voit qu'elle, et tout le reste du système demeure une boîte noire qu'il n'a ni à voir ni à comprendre. L'interface doit donc présenter les informations obtenues de façon intelligible, et doit permettre de visualiser l'information de façon conviviale, concise et efficiente; elle a donc été conçue pour les systèmes d'exploitation graphiques biens connus, tels Microsoft Windows et X11R6. L'interface incorpore plusieurs outils qui la rendent facile à utiliser, sans en compromettre sa puissance. Cette section détaille ces outils afin de permettre à un usager de savoir tirer profit du logiciel.

7.1 Description de l'interface

Conçue en C++ à l'aide de la bibliothèque Zinc Application Framework 4.2, la présente version de l'interface fonctionne sur Windows 3.11, Windows 95 et Windows NT. Elle nécessite l'utilisation d'un ordinateur de type PC-486 ou mieux, avec 8 méga octets de mémoire vive et environ 3 méga octets d'espace disque. Un moyen de communication aux contrôleurs centraux est aussi nécessaire. Selon les besoins, un modem, une connexion RS-232-C/EIA-232D ou une connection ethernet peuvent être utilisés. De plus, l'utilisation d'un écran SVGA et d'une souris est fortement recommandée.

La conception du logiciel est couverte à la section 7.3 suivante, mais le choix du langage et de la bibliothèque d'interface a été motivé par les raisons suivantes:

1. ***l'extensibilité***: le langage C++ permet la réutilisation des composantes déjà construites, à travers sa structure "objet" et la hiérarchisation de ces objets à travers l'hé-

ritage. Il est donc facile et rapide d'ajouter de nouvelles fonctions au logiciel, en réutilisant les objets existants ou en les modifiant au besoin;

2. ***la flexibilité***: la bibliothèque d'interface graphique est une bibliothèque multi-plateformes. Elle permet la création d'interfaces fonctionnant sous Windows, SunOS, Solaris, SCO UNIX, O/S 2 et Macintosh: le code de l'interface n'a qu'à être recompilé sur la nouvelle plate-forme pour fonctionner adéquatement. De plus le code entier de la bibliothèque graphique est fourni; ceci nous a permis, après quelques modifications au code, de recompiler la bibliothèque en entier sur le système d'exploitation Linux. Toutefois la bibliothèque ne supporte malheureusement pas les structures de bas niveaux, tels les outils de communications. Pour chaque nouvelle plate-forme utilisée, ces outils devront être ré-écrits partiellement ou même complètement;

3. ***puissance et rapidité d'exécution***: quoique le langage C++ ne soit pas le langage le plus rapide à l'exécution, il est tout de même plus rapide que les langages interprétés. De plus la lutte commerciale que se livrent actuellement les compagnies de compilateurs tel Borland et Microsoft a comme effet de rendre les compilateurs de plus en plus sophistiqués; ceux-ci produisent un code exécutable d'une rapidité souvent supérieure à celui produit par une programmation en code assembleur traditionnelle;

4. ***fiabilité***: l'utilisation du C++ comme langage de programmation augmente quelque peu le temps de production d'un logiciel complet. Cependant elle permet aussi de réduire substanciellement la période de déverminage ("*déboggage*"). La structure de conception du logiciel est beaucoup plus stable, donc plus fiable et moins sujette aux erreurs de programmation. De plus, la bibliothèque d'interface Zinc est suppor-

tée d'un support aux utilisateurs, en cas de difficultés. Les techniciens sont disponibles par téléphone, télécopieur et par courrier électronique.

7.1.1 Boutons d'utilisation générale

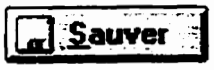
Avant de commencer la description de l'interface, il est important de noter qu'une série de boutons se retrouvent dans presque toutes les fenêtres et boîtes de dialogue de l'interface. Ces boutons ont pour but de faciliter l'accès à certaines fonctions couramment utilisées. Ces boutons d'utilisation générale sont au nombre de six.

Le bouton muni d'un **crochet vert** permet d'accepter les conditions de la fenêtre pour poursuivre et en certain cas de fermer la fenêtre.

Le bouton muni d'une **croix rouge** permet d'annuler l'action en cours et dans certains cas de fermer la fenêtre active.

Le bouton muni d'un **point d'interrogation** donne accès au système d'aide contextuelle de l'interface usager. En appuyant sur ce bouton, l'usager aura l'aide sur la fenêtre courante. Une aide générale et un index d'aide sont disponibles losque ce bouton est utilisé.

Le bouton muni d'une icône d'**imprimante** permet d'imprimer les informations de la fenêtre courante, sous forme de rapport sommaire. Il est aussi possible de modifier la configuration de l'imprimante à partir de ce bouton.



Le bouton avec une icône de **disquette** donne accès aux fonctions de sauvegarde. Il permet de sauvegarder l'information de la fenêtre courante, pour éventuellement la restaurer dans le logiciel ou pour permettre de l'incorporer dans un rapport.

Le bouton muni d'une **flèche effectuant un 360°** permet de rafraîchir les informations dans la fenêtre courante. Dans plusieurs fenêtres, les informations fluctuent selon le temps. Certaines fenêtres ont une variante de ce bouton qui permet le rafraîchissement automatique périodique, l'intervalle de temps étant variable et fixé par l'usager.

7.1.2 Étapes pour l'établissement d'une connexion

Avant d'utiliser l'interface à proprement dit, l'usager doit se brancher sur un contrôleur central, à travers un moyen de communication quelconque. Lorsque le programme est démarré, une fenêtre de sélection du mode de connexion apparaît: cinq modes de connexion sont disponibles.

1. **Directe**: l'ordinateur utilisé est branché directement sur le contrôleur central à travers un câble série et un adapteur "null-modem".
2. **Modem**: l'ordinateur utilise un modem interne ou externe branché sur une ligne permettant d'accéder au contrôleur central. La chaîne d'initialisation du modem doit être connue lors de la première utilisation du logiciel.
3. **Modem & opérateur**: l'ordinateur utilise un modem interne ou externe branché à une ligne téléphonique incapable d'accéder à l'extérieur (dans le cas d'un appel interurbain, par exemple) sans l'intermédiaire de l'opérateur(trice) téléphonique. L'usager choisit cette option et téléphone à l'opérateur lui demandant d'effectuer la connexion. L'interface demandera ensuite à l'usager de confirmer le moment où le modem hôte répondra. Il n'a qu'à raccrocher son téléphone ensuite.
4. **Ethernet**: lorsque le contrôleur central est aussi branché sur le même réseau que l'ordinateur utilisé pour l'interface, ce mode de connexion peut être sélectionné.

5. **Serveur modem:** lorsque l'ordinateur utilisé pour l'interface est branché sur un réseau ethernet, et qu'un serveur de modem est branché sur ce même réseau, cette option peut être utilisée. Il suffit de fournir l'adresse ethernet du serveur, et le reste est pris en charge par l'interface. L'interface supporte pour l'instant les serveurs de modems traditionnels sur stations UNIX.

En plus de ces cinq modes, un mode "démo" est aussi disponible. Ce mode permet de visualiser les différentes fenêtres de l'interface. Cependant ce mode n'est pas d'une grande utilité au quotidien; il permet de démontrer sommairement les possibilités de l'interface. L'illustration de la figure 7.1 présente la fenêtre de sélection du mode de connexion.

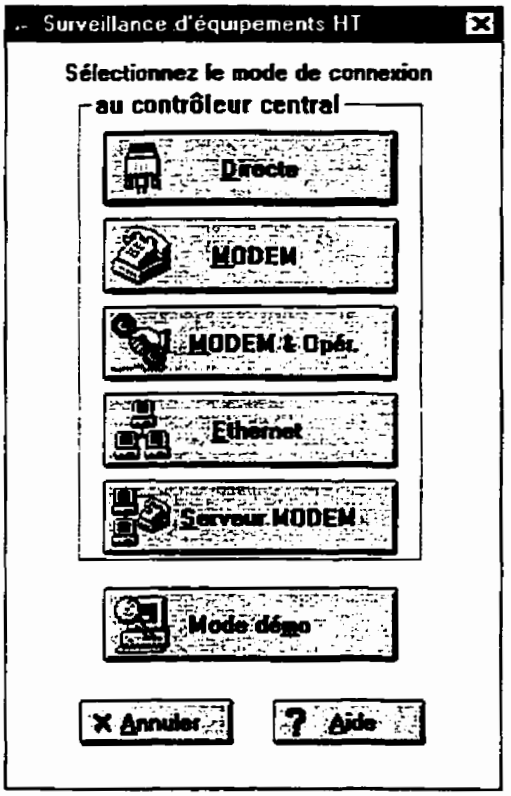


Figure 7.1: Fenêtre de sélection du mode de connexion

L'interface usager permet à un usager de se brancher sur plusieurs contrôleurs centraux: chacun de ces contrôleurs centraux est désigné comme "sous-station", car on retrouve d'ha-

bitude un seul contrôleur central par sous-station électrique surveillée. Cette sous-station est caractérisée par différents paramètres de connexion, soit un numéro de téléphone, une adresse ethernet, un taux de transfert, etc. Ces paramètres ne devront être spécifiés qu'une seule fois, lors de la première utilisation de la sous-station en question, par un administrateur de système. La section suivante couvre la configuration d'une sous-station.

Une fois que l'usager a choisi un mode de connexion à l'aide de la fenêtre de la figure 7.1, la seconde fenêtre qui apparaît (figure 7.2) à l'usager lui permet justement de sélectionner cette sous-station:

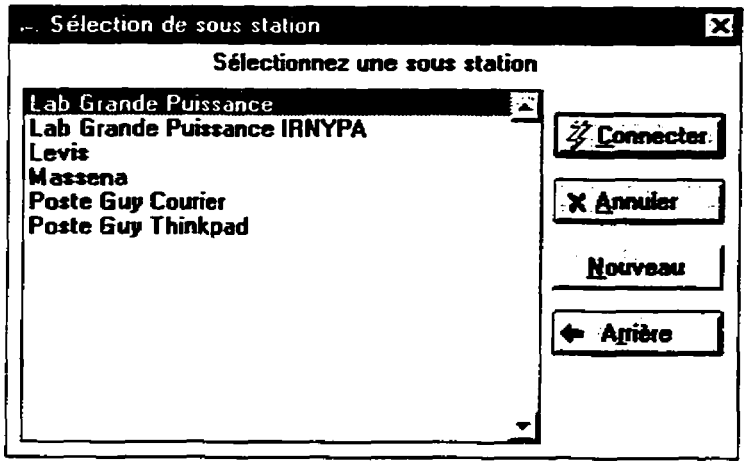


Figure 7.2: Fenêtre de sélection de la sous-station

Une fois la sous-station sélectionnée, l'usager n'a qu'à sélectionner le bouton "Connecter", et attendre que le lien s'effectue avec le contrôleur central. Le temps d'établissement de la connexion varie selon le mode de connexion; une connexion ethernet prendra quelques secondes, selon la charge du réseau, et une connexion modem prendra jusqu'à trente secondes.

Si une erreur survient, à cause d'une ligne occupée pour une connexion modem par exemple, l'usager en est averti aussitôt l'erreur connue. Le bouton "Arrière" est disponible si l'utilisateur décide de changer de mode de connexion. Le bouton "Nouveau" est disponible

pour l'opérateur du système afin qu'il puisse créer, détruire ou la modifier les sous-stations logiques.

Une fois la connexion établie correctement, une validation d'usager a lieu à l'aide de la fenêtre représentée à la figure 7.3. L'utilisateur devra fournir son nom d'usager et son mot de passe: ces informations sont nécessaires pour assurer la gestion des accès selon les utilisateurs (c.f. section 7.1.3).

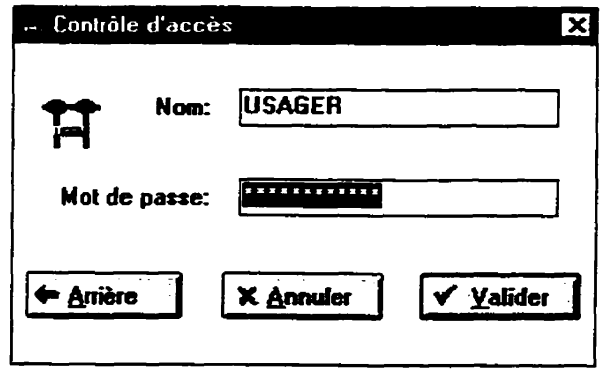


Figure 7.3: Fenêtre de contrôle d'accès

Une fois le mot de passe validé par le contrôleur central, l'usager a accès aux fenêtres qui lui sont disponibles selon son niveau d'accès. Si un usager entre trois fois successivement le mauvais mot de passe, l'interface usager se désactive d'elle-même, et il sera nécessaire de réinstaller l'interface pour y avoir accès de nouveau.

7.1.3 Gestion des priorités d'utilisateur

Comme nous l'avons mentionné à la section précédente, le système de télésignalisation permet la gestion de niveaux d'accès selon les utilisateurs. On peut associer neuf différents niveaux d'accès aux différents usagers: le niveau prioritaire est le numéro 1; le niveau d'accès le moins prioritaire est le numéro 9.

Pour effectuer la gestion, on associe à chacune des fenêtres de l'interface usager une série de cinq chiffres. Chacun de ces chiffres représente une action qui peut être effectuée sur la fenêtre. Certains de ces chiffres peuvent ne pas être utilisés. Prenons l'exemple de la fenêtre de reconnaissance des alarmes, avec un usager de niveau 2 et un chiffre de priorité de "98519". Quatre différentes actions peuvent être effectuées sur la fenêtre:

Tableau 7.1: Exemple des niveaux d'opérateurs pour l'interface usager

Chiffre	Action sur la fenêtre	Exemple
Premier	Accéder à la fenêtre	9
Deuxième	Imprimer la fenêtre	8
Troisième	Reconnaître les alarmes	5
Quatrième	Désactiver les alarmes	1
Cinquième	Inutilisé	9

Pour avoir accès à une des fonctions, un usager doit avoir un niveau de priorité égal ou supérieur au chiffre de l'action. Donc, dans ce cas-ci, l'usager de niveau 2 pourra accéder à la fenêtre, l'imprimer et reconnaître les alarmes. Il ne pourra cependant pas désactiver les alarmes. Un usager de niveau 8 ne pourra qu'accéder à la fenêtre et l'imprimer.

Chacun de ces nombres associés aux différentes fenêtres de l'interface est entièrement configurable. L'administrateur de système peut fixer les valeurs au gré de ses besoins. Les détails de cette configuration sont couverts à la section suivante.

7.1.4 Description de la fenêtre principale

La fenêtre principale est aussi appelée fenêtre parent, puisque toutes les autres fenêtres sont des sous-fenêtres de celle-ci. Son architecture est bâtie comme l'organisation physique du poste: les 21 départs doubles sont répartis en quatre groupes. Chacun de ces groupes, appe-

lés équipements, est la représentation d'un contrôleur local et chacun de ces équipements possède un certain nombre de signaux. L'architecture est représentée par la figure 7.4.

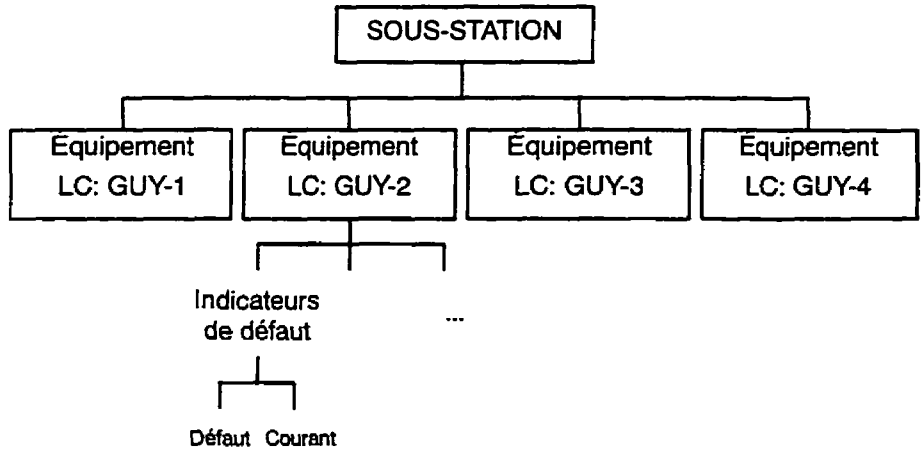


Figure 7.4: Architecture du logiciel d'interface usager vu par l'utilisateur

Chacun des contrôleurs locaux est responsable d'un certain nombre d'indicateurs de défaut. Dans ce cas, 126 indicateurs de défaut produisent 252 signaux. Cette architecture est respectée dans le logiciel, puisqu'il est possible d'examiner la sous-station en entier, un des équipements ou un des signaux. Les alarmes sont affichables selon la sous-station entière ou selon l'équipement. Il existe donc, tel qu'on peut le constater à la figure 7.5, un menu pour chaque élément(i.e. sous-station et équipement).

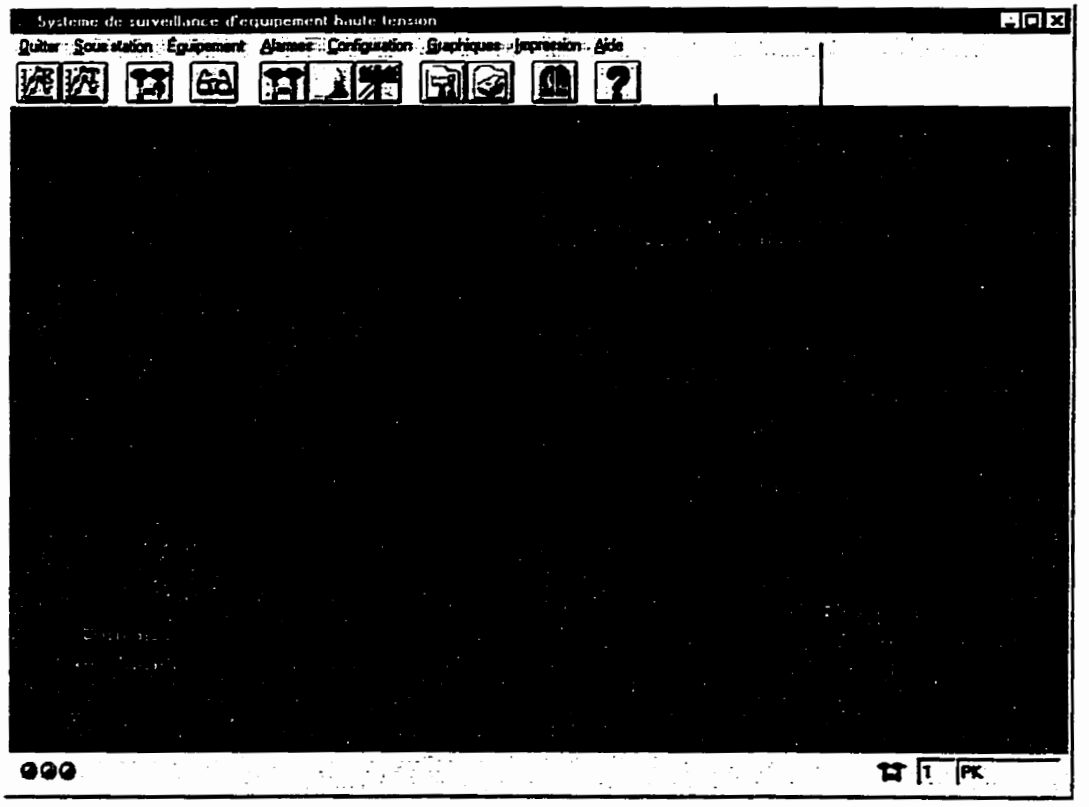


Figure 7.5: Fenêtre principale de l'interface usager

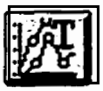
Au premier coup d'oeil à la figure 7.5, on remarque quatre composantes à l'interface:

- ***un menu déroulant*** donnant accès à toutes les fonctions de l'interface;
- ***une barre d'outils*** donnant accès rapidement aux fonctions les plus utilisées de l'interface. Chacun de ces éléments est aussi disponible dans le menu déroulant;

Figure 7.6: Barre d'outils de l'interface usager

On retrouve onze boutons sur la barre d'outils. Voici la description de chacun de ces boutons:

Fait apparaître la fenêtre permettant la création de **graphiques d'événement;**



Fait apparaître la fenêtre permettant la création de **graphiques de tendances** (section 7.1.6.5);

Permet de **sélectionner l'équipement** actuellement sélectionné (section 7.1.6.1);

Affiche la fenêtre de **lecture des capteurs** (section 7.1.6.2);

Permet d'accéder à la fenêtre de **statut d'équipement** (section 7.1.6.1);

Donne accès à la fenêtre des **alarmes** (section 7.1.6.3). L'usager peut à partir de cette fenêtre voir les alarmes actives et leurs seuils;

Donne accès à la fenêtre de **statut de sous-station**;

Permet à l'usager de **configurer l'interface usager** en lui affichant la fenêtre de configuration (section 7.1.6.6);

Donne accès à la fenêtre de configuration d'imprimante, permettant ainsi de **configurer l'imprimante** avant une impression;

Sert à **quitter l'interface** rapidement. L'usager devra confirmer qu'il veut bien quitter l'interface avant que le programme ne termine son exécution;

Donne accès au système d'**aide contextuelle** de l'interface usager. Une aide contextuelle et un index d'aide sont disponibles.

- ***Un indicateur d'équipement sélectionné.*** À chaque équipement est associé un numéro et un nom. Le numéro et de nom de l'équipement sélectionné est affiché à cet endroit en permanence. Les opérations qui auront lieu sur un équipement (à travers le menu "Équipement", par exemple) s'effectueront sur cet équipement;

- ***Un indicateur de statut des communications***. L'indicateur peut être dans un des trois états suivants:


La lumière verte, située à gauche complètement, est allumée. L'interface est en mode d'attente. Aucune communication n'a lieu à ce moment;


La lumière jaune, située au centre, est allumée. L'interface a envoyé une requête, et la réponse a été reçue. L'interface est présentement à traiter les informations reçues afin de les afficher;


La lumière rouge, située à l'extrémité droite, est allumée. L'interface est en attente d'une réponse du contrôleur local. Une requête a été envoyée, et le résultat se fait attendre ou est partiellement arrivé. Il est possible d'interrompre l'attente en appuyant sur la touche <*Esc*>.

En plus de pouvoir accéder aux fonctions à travers la barre d'outils, il est possible d'y accéder à l'aide du menu déroulant. De plus, certaines fonctions utilisées moins fréquemment ne sont accessibles qu'à travers le menu. Nous couvrirons ici la description des différents éléments du menu déroulant.

7.1.5 Description des éléments du menu déroulant

Huit éléments sont disponibles au premier niveau dans le menu déroulant:

1. ***Quitter***: permet de quitter l'interface;
2. ***Sous-station***: donne accès à toutes les fonctions ayant trait à une sous-station. Le statut des alarmes pour la sous-station, l'usure des équipements (lorsque applicable), l'explication des alarmes, l'historique et les commentaires associés aux alarmes sont disponibles;

3. ***Équipement***: à part l'option de sélection de l'équipement courant, toutes les options de ce menu s'opèrent sur l'équipement présentement sélectionné. On peut accéder au statut de l'équipement, à l'explication des alarmes, à l'historique et aux commentaires associés aux alarmes préalablement reconnues. De plus, on peut accéder à la lecture des capteurs et aux signaux calculés de l'équipement sélectionné;

4. ***Alarmes***: ce menu permet d'aller voir et reconnaître les alarmes présentes dans le système, d'activer et de désactiver ces alarmes et de fixer le niveau des alarmes;

5. ***Configuration***: ce menu présente toutes les possibilités de configuration de l'interface usager. On peut y changer son mot de passe, modifier la liste des opérateurs, changer les priorités d'accès selon les niveaux des usagers, modifier les paramètres d'utilisation de l'interface, configurer les communications et les ports modems. Il est aussi possible d'alterner entre le français et l'anglais à partir de ce menu;

6. ***Graphiques***: donne la possibilité à l'usager de créer des graphiques de tendance et d'événement, d'éditer ces graphiques, de les imprimer et d'effectuer un "*Zoom*" avant et arrière;

7. ***Impression***: à travers ce menu, on peut accéder à la configuration de l'imprimante sélectionnée et on peut imprimer un fichier texte;

8. ***Aide***: ce dernier menu présente un index d'aide, une fenêtre d'information sur le présent statut de l'interface, des détails sur le programme à travers une fenêtre "À propos de ce programme", et permet d'extraire une série d'informations à rapporter au développeur en cas de problème.

7.1.6 Description des principales fenêtres

L'interface est composée de plusieurs dizaines de fenêtres, chacune d'entre elles est conçue à ce que les informations soient présentées le plus rapidement et clairement possible. Cependant, certaines d'entre elles peuvent sembler quelque peu rébarbatives, vu la nature complexe des informations à présenter. Cette section est consacrée à la présentation de chacune des fenêtres principales de l'interface.

Les fenêtres les plus utilisées pour le projet pilote du poste Guy sont les fenêtres contenues sous les menus "Équipement" et "Alarmes". Les fenêtres de graphiques seront aussi d'une grande utilité pour l'opération du système à long terme. Ces fenêtres seront présentées ici plus en détail.

7.1.6.1 Fenêtre de sélection de l'équipement courant

Cette fenêtre (figure 7.7) présente le nom et numéro des équipements du système et permet de sélectionner l'équipement sur lequel les opérations du menu "Équipement" seront effectuées.

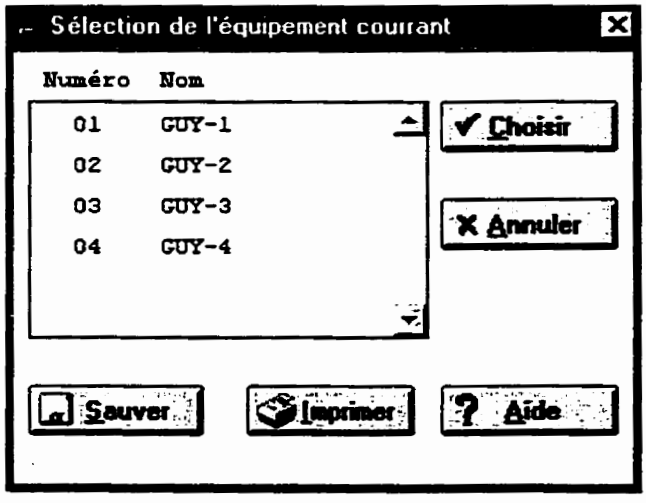


Figure 7.7: Fenêtre de sélection de l'équipement courant

Le numéro et le nom sélectionnés sont ceux qui apparaissent au bas à droite de la fenêtre principale. Le contenu de cette fenêtre peut être imprimé ou sauvegardé à l'aide des boutons correspondants.

7.1.6.2 Fenêtre de lecture des capteurs

Il est possible, grâce à la fenêtre de la figure 7.8, d'avoir une lecture de la valeur minimum, maximum et moyenne de la dernière minute de tous les différents capteurs branchés sur les équipements.

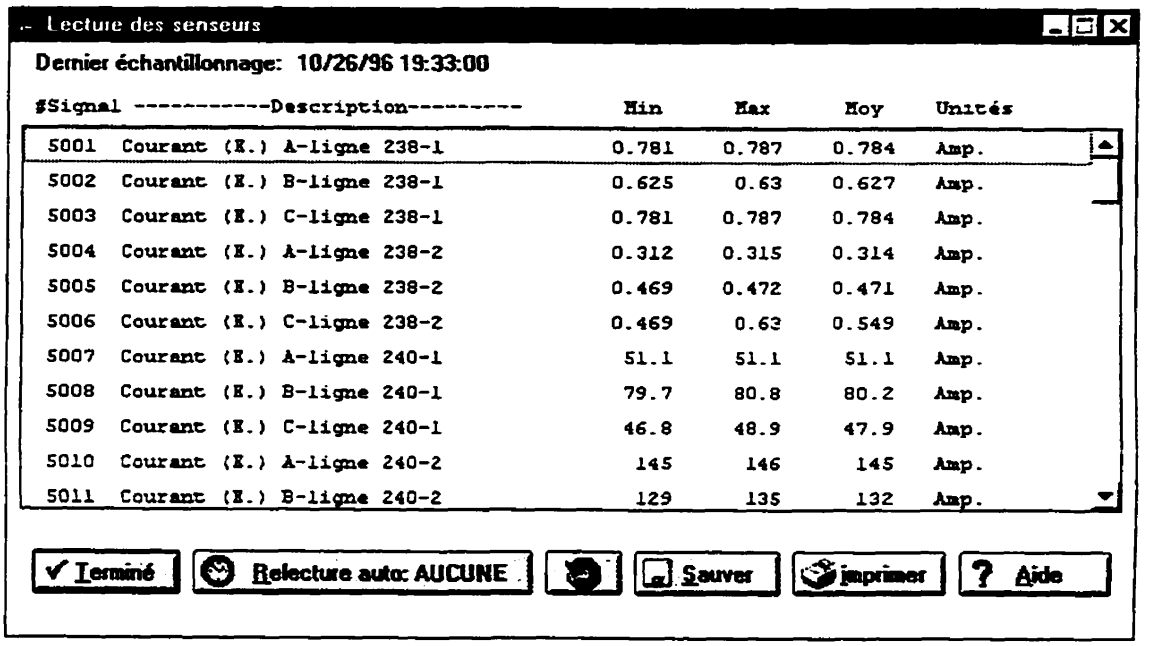


Figure 7.8: Fenêtre de lecture des capteurs

On aperçoit au haut de la fenêtre la date de la dernière mise-à-jour de ces valeurs à partir du contrôleur local vers le contrôleur central. La liste verticale nous donne, dans l'ordre du numéro de signal, sa description, les valeurs minimales, maximales et moyennes de la dernière minute et les unités de ces valeurs. On peut comme à la fenêtre précédente, imprimer ou sauvegarder le contenu de la liste verticale à l'aide des boutons d'impression et de

sauvegarde. Le bouton "Relecture Auto" permet aux informations d'être rafraîchies automatiquement à un intervalle de temps donnée. Lorsque l'usager appuie sur ce bouton, la fenêtre suivante apparaît, lui demandant de fournir un intervalle de temps, en format heure/minute/seconde:

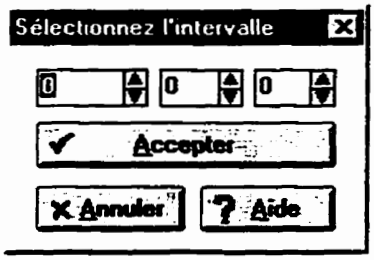


Figure 7.9: Fenêtre de sélection d'un intervalle de temps

Une fois l'intervalle sélectionné à l'aide de la fenêtre de la figure 7.9, l'intervalle apparaîtra sur le bouton et la relecture sera refaite à cet intervalle. Pour annuler la relecture automatique, il suffit de placer l'intervalle à 0 heure, 0 minute 0 seconde.

On doit noter que les informations présentes dans cette fenêtre sont celles de l'équipement sélectionné. Pour obtenir la lecture de capteurs situés sur d'autres équipements, on doit accéder à la fenêtre de sélection de l'équipement courant, sélectionner le nouvel équipement, et attendre la relecture automatique des informations.

7.1.6.3 Fenêtre de reconnaissance des alarmes

La fenêtre de reconnaissance des alarmes (figure 7.10) permet de visualiser à l'aide d'une liste verticale les alarmes présentes sur un équipement. La fenêtre affiche le numéro d'équipement sur lequel l'alarme s'est produit, le numéro de signal impliqué, la description de l'alarme, le type d'alarme, la date et l'heure de d'activation de l'alarme, la date et l'heure fin d'alarme (s'il y a lieu) et indique si l'alarme a été reconnue ou non.

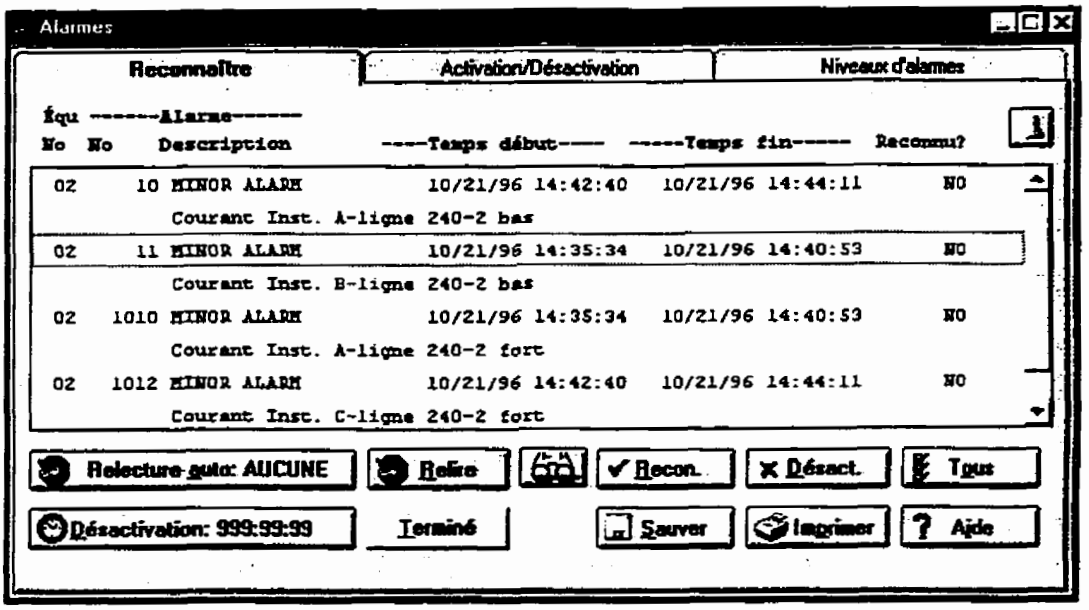


Figure 7.10: Fenêtre de reconnaissance des alarmes

Les alarmes ne disparaissent de la fenêtre de reconnaissance que seulement si elles ont été reconnues et si l'alarme est terminée (i.e. lorsqu'il y a une date de fin). Lorsque l'alarme disparaît, elle s'enregistre dans la liste de l'historique des événements de l'équipement.

En plus de permettre la visualisation des alarmes, cette fenêtre permet aussi d'accuser réception d'une alarme en cours. L'usager peut sélectionner une ou plusieurs alarmes afin d'en témoigner la présence au moyen du bouton de gauche de la souris et d'appuyer sur le bouton "Recon.". Si l'usager en a la permission, l'interface lui demandera alors d'entrer un commentaire associé à la reconnaissance des alarmes sélectionnées. Ce commentaire ira s'inscrire dans l'historique de l'équipement.

Si un opérateur sait qu'une alarme sera redéclenchée périodiquement sur un laps de temps donné, il peut désactiver l'alarme en question pour une période de temps pour éviter que ce signal se répète jusqu'au moment où il aura corrigé la situation. Le bouton "Désactivation" muni d'une horloge sert à déterminer le temps de désactivation voulu. Le bouton "Désact." sert à désactiver les alarmes sélectionnées selon la période de temps indiquée par

le bouton "Désactivation". Ainsi un opérateur pourra éviter de submerger le système d'alarmes non pertinentes. Veuillez prendre note qu'une période de désactivation de 999:99:99 est interprétée comme une désactivation permanente.

Finalement, si l'utilisateur veut observer la valeur présente du signal ayant causé l'alarme, il sélectionne l'alarme et appuie sur le bouton à lunettes. La fenêtre de lecture des capteurs apparaîtra, avec la valeur du signal responsable identifiée clairement sur la fenêtre. Un double "click" avec le bouton de gauche de la souris effectuera le même travail.

7.1.6.4 Fenêtre d'activation et de désactivation des alarmes

Si une maintenance est prévue sur différents équipements, il est possible de désactiver les alarmes affectant ces équipements pour éviter d'inonder le système d'alarmes inutiles. Cette fenêtre (figure 7.11) présente sous forme de liste toutes les alarmes ayant effet sur un équipement donné, et permet d'activer et de désactiver les alarmes sur des périodes de temps déterminé par l'opérateur.

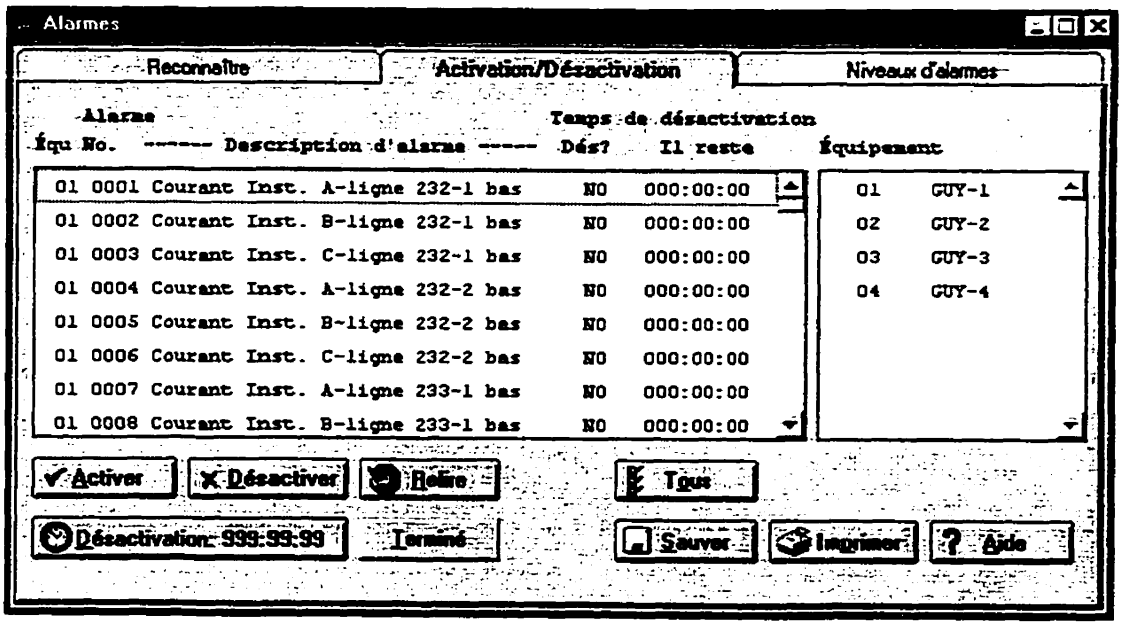


Figure 7.11: Fenêtre d'activation et de désactivation des alarmes

La fenêtre indique le numéro d'équipement sur lequel s'applique l'alarme, le numéro de l'alarme, la description de l'alarme, le statut de l'alarme (désactivé ou non), et le temps de désactivation associé à l'alarme. On remarque une deuxième liste verticale à droite de la fenêtre: pour changer d'équipement, il suffit de sélectionner un des équipements pour avoir accès aux alarmes de l'équipement en question.

Comme pour la fenêtre précédente, on retrouve un bouton pour spécifier le temps de désactivation et un autre pour désactiver l'alarme en question. On remarque aussi la présence d'un autre bouton, celui de l'activation des alarmes qui permet de réactiver une alarme immédiatement.

7.1.6.5 Graphiques de tendance

Un opérateur peut obtenir la tendance de n'importe quel signal sur une plage de temps donné. Un seul ou plusieurs signaux peuvent être superposés sur le même graphique, comportant des signaux de différents équipements ou étalés sur des périodes de temps différentes. La fenêtre de la figure 7.12 permet de créer de tels graphiques.

Pour bien en comprendre son fonctionnement, nous allons balayer la fenêtre de bas en haut et de gauche à droite afin d'expliquer chacune des options de cette fenêtre.

Le premier élément que l'on retrouve est une liste tombante ("*drop down list*") contenant les différentes descriptions de signal. L'usager choisit à l'aide de cette liste le signal qu'il veut faire afficher. Il peut aussi le choisir à partir de son numéro et de la boîte tournante ("spin box") situé à droite de la description du signal.

Si l'opérateur désire faire rafraîchir la liste des signaux parce que, par exemple, il a sélectionné un équipement différent, il peut le faire à l'aide du bouton situé en haut à droite complètement.

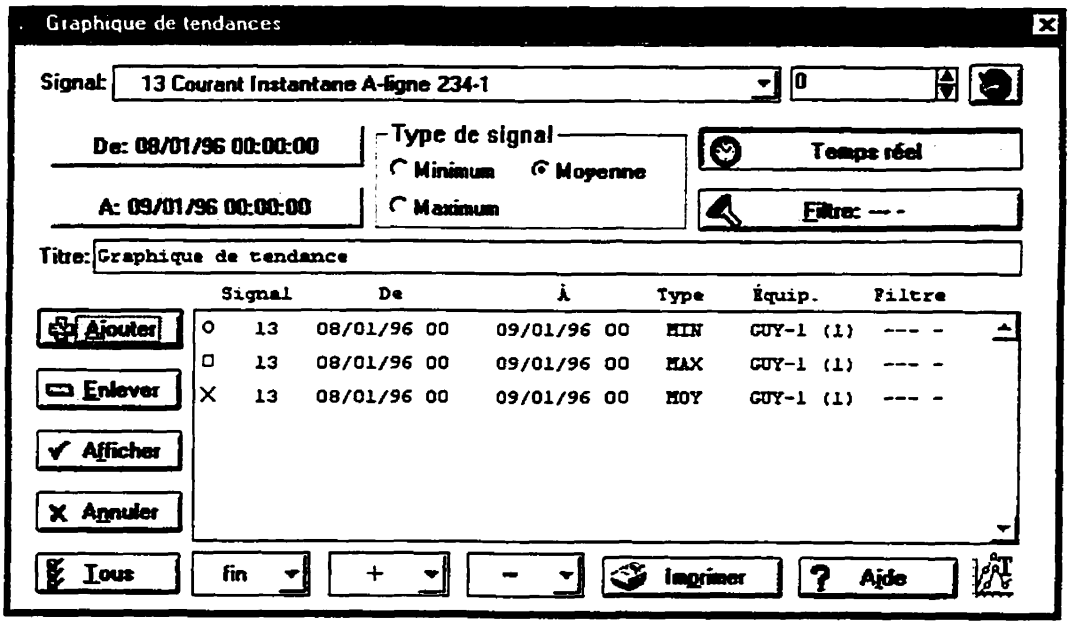


Figure 7.12: Fenêtre de création de graphique de tendances

Sur la ligne suivante, on aperçoit deux boutons avec une date et une heure. Un de ces boutons porte la désignation "De" et l'autre "À". L'usager choisit la plage de temps sur lequel il veut avoir la tendance à partir de ces boutons.

Trois boutons radio désignant le type de signal, minimum, maximum, et moyenne servent à définir le type de signal obtenu. Tout de suite à droite sont situés les boutons "Temps réel" et "Filtre". Le premier bouton permet d'afficher l'échelle de l'abscisse en dates plutôt qu'en nombre d'échantillons, et le second permet de filtrer une partie des données de la courbe pour éliminer les signaux transitoires. Une boîte de texte, étiquetée "Titre" et située sous ces boutons, permet de donner un titre au graphique.

En dessous du titre apparaît l'élément principal de la fenêtre: une liste verticale indiquant toutes les courbes ayant été sélectionnées pour l'affichage. Les boutons situés à la gauche servent à la manipulation des informations dans cette liste verticale.

Pour bien comprendre comment fonctionne cette fenêtre, nous décrivons ici les manipulations nécessaires pour obtenir le contenu de la liste verticale de la figure 7.12.

1. Choix du signal "13 Courant instantané A Ligne 234-2" dans la liste des signaux.
2. Sélection de la plage de temps avec les boutons "De" et "À".
3. Sélection du type de signal minimum.
4. Sélection de la couleur voulue et du marqueur circulaire.
5. Utilisation du bouton "Ajouter". La première courbe apparaît dans la boîte verticale.
6. Sélection du type de signal maximum.
7. Sélection de la couleur voulue et du marqueur carré.
8. Utilisation du bouton "Ajouter". La seconde courbe apparaît dans la boîte verticale.
9. Sélectionner le type moyenne, la couleur voulue et le marqueur "X" et appuyer sur ajouter.

Une fois les manipulations ci-haut exécutées, on aperçoit exactement le contenu de la figure 7.12. Il suffit d'appuyer sur le bouton "Afficher" pour obtenir le résultat présenté à la figure 7.13.

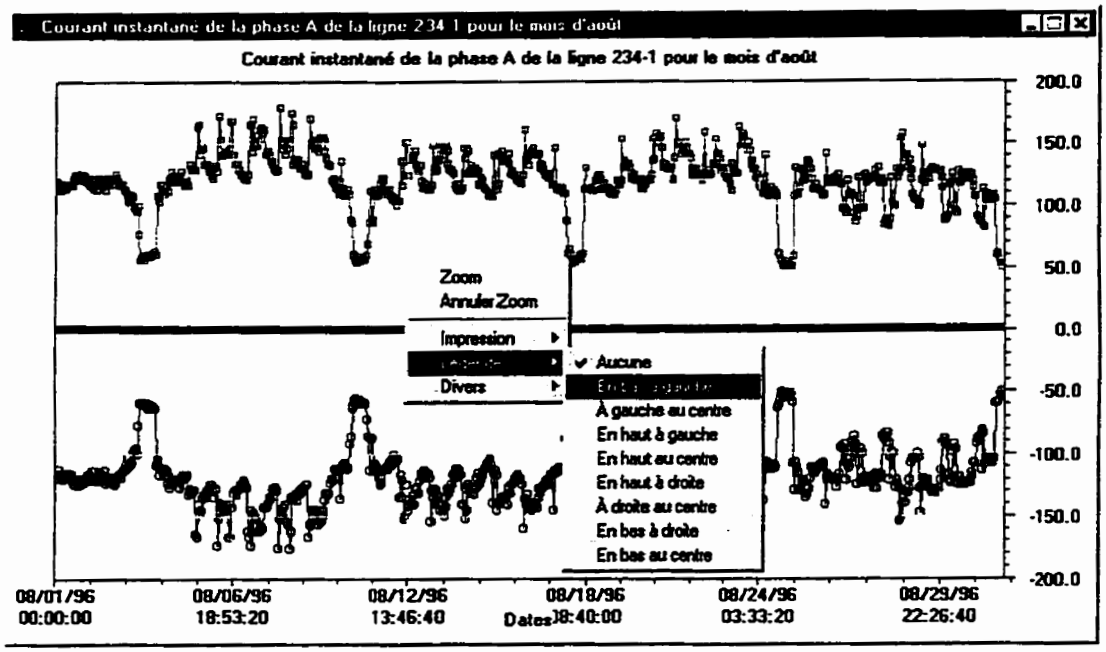


Figure 7.13: Graphique de tendances

On peut apercevoir un menu sur le graphique en question. Ce menu apparaît lorsque l'utilisateur utilise le bouton de droite de la souris sur la fenêtre du graphique. Ce menu permet:

1. d'afficher une légende à l'endroit voulu. La figure 7.14 présente un graphique ayant une légnede;
2. d'effectuer un zoom avant et arrière;
3. d'imprimer le graphique ou de modifier la configuration de l'imprimante;
4. de rafraîchir l'affichage du graphique;
5. de changer les échelles;
6. d'extrapoler de l'information à partir de curseurs.

Lorsque l'usager choisit l'option "Zoom", l'interface demande deux points qui serviront d'extrémités. Ensuite, l'interface réajustera le graphique pour permettre une vue agrandie du graphique.

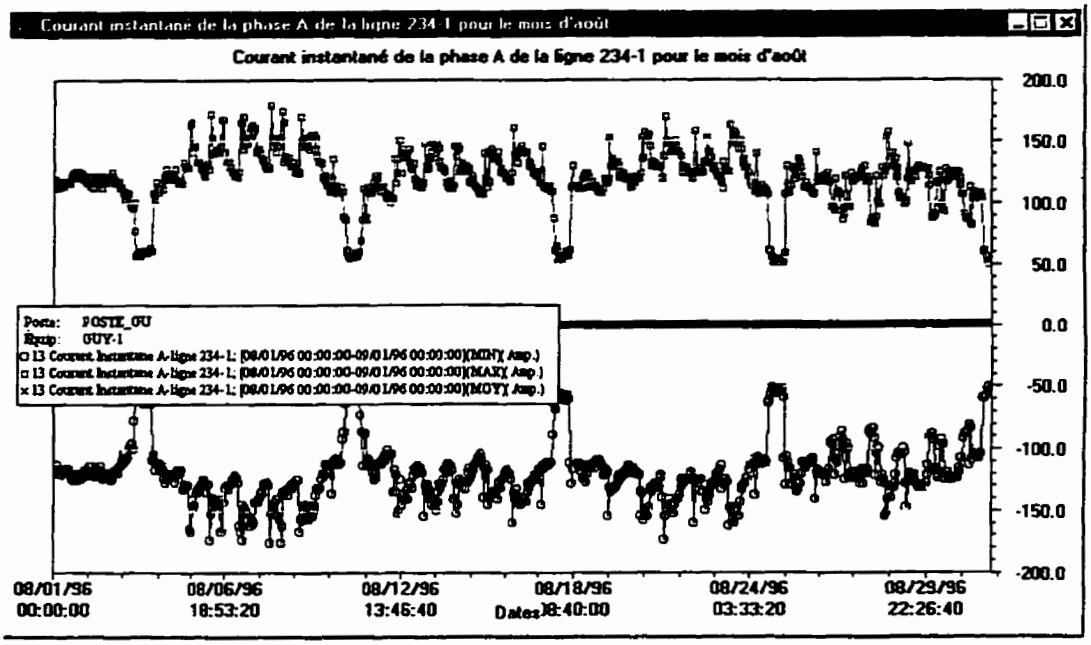


Figure 7.14: Graphique avec légende

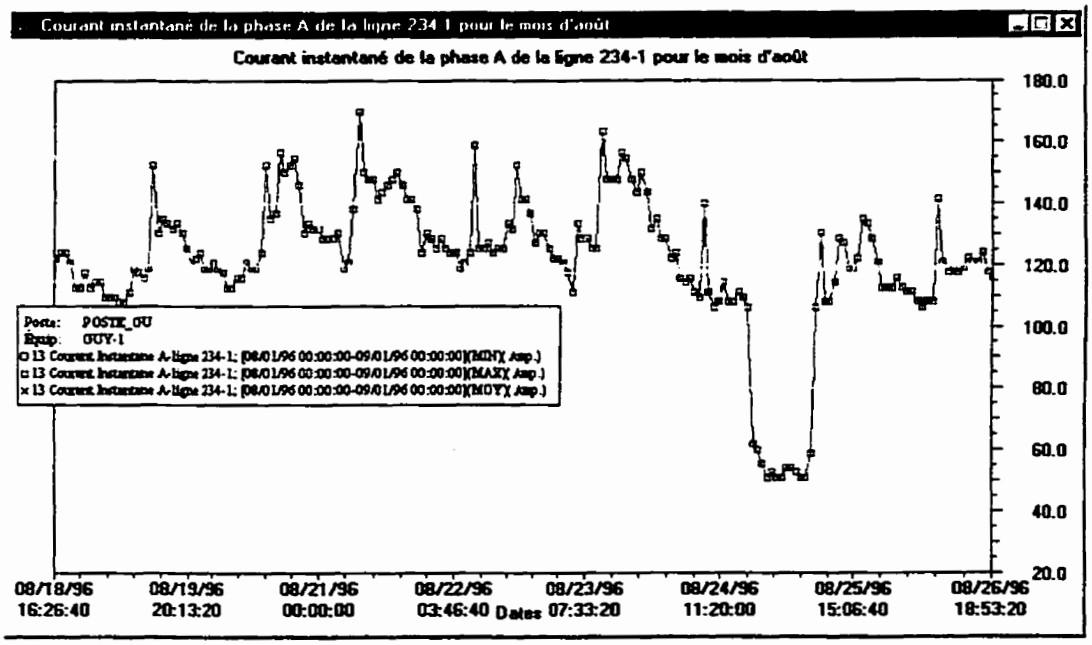


Figure 7.15: Zoom dans les graphiques de tendance

Sur l'exemple 7.15 précédent, l'utilisateur a fait un zoom avant sur les valeurs maximales couvrant du 18 au 26 août (le graphique original couvre tout le mois d'août).

L'option de zoom permet de voir avec plus de détails les données affichées par le graphique. Si, cependant, l'utilisateur a besoin d'encore plus de précision dans l'extrapolation des informations du graphique, il peut utiliser l'option "Extrapoler" disponible à l'aide du bouton de droite de la souris. Dans ce mode, chaque "click" à l'aide du bouton de gauche sur le graphique fait apparaître une croix à l'endroit de la sélection, et les valeurs d'ordonnée et d'abscisse de la sélection sont affichées sur le graphique (figure 7.16).

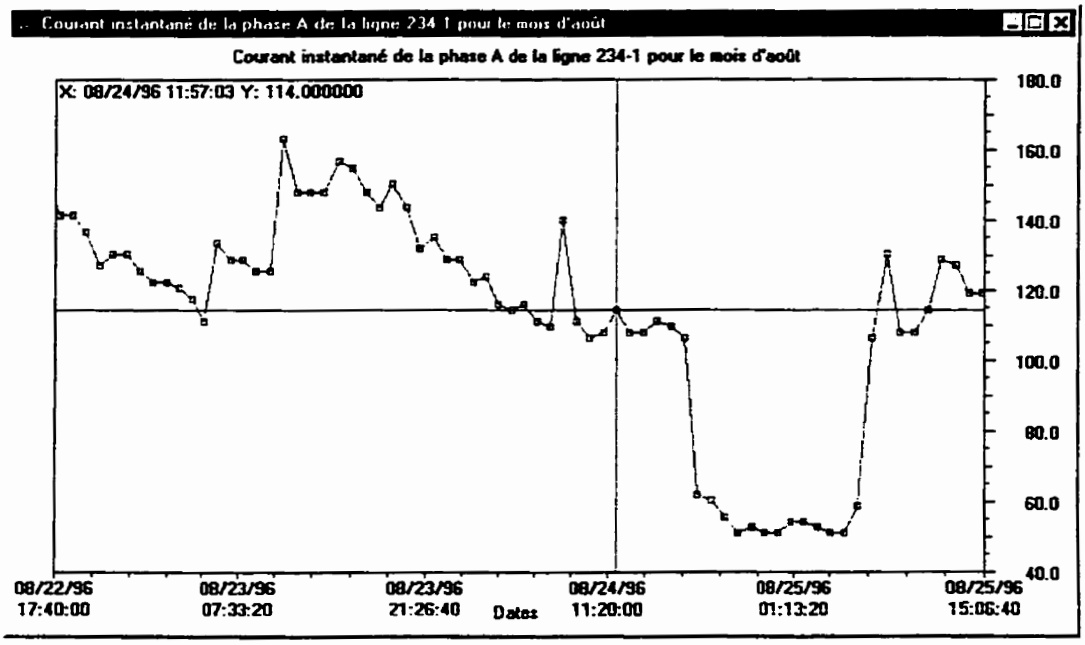


Figure 7.16: Extrapolation des graphiques de tendances

À chaque nouvelle sélection, une nouvelle croix apparaît. De cette façon, le graphique peut devenir rapidement saturé de lignes horizontales et verticales. Pour effacer ses lignes, il suffit de sélectionner l'option "Rafraîchir" dans le menu accessible à partir du bouton de droite de la souris.

Il est aussi possible d'effectuer un certain filtrage sur les courbes pour en éliminer une partie des transitoires du signal. Un algorithme de FFT viendra adoucir la courbe, en fonction d'une valeur, la valeur de filtrage, spécifiée par l'usager. Cette valeur varie entre 0, indi-

quant qu'aucun filtrage n'est effectué, et 256, indiquant une filtration majeure. Lorsque le bouton de filtrage est sélectionné, la fenêtre de la figure 7.17 apparaît.

Elle permet de sélectionner la valeur de filtre voulue, ainsi que l'état du filtre (activé ou non). L'option "Filtrage par type" ne s'applique que dans les systèmes de surveillance de disjoncteurs et n'a pas d'utilité pour le présent projet.

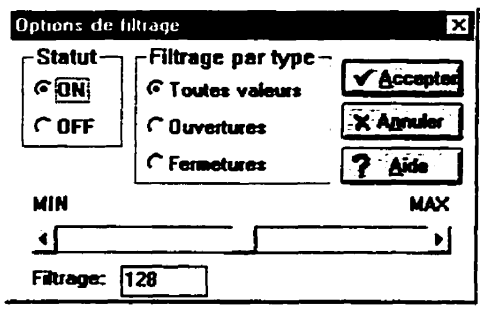


Figure 7.17: Options de filtrage des courbes

L'exemple suivant illustre l'utilisation des filtres de différentes valeurs. La même courbe est représentée quatre fois avec les valeurs de filtrage suivants: 0, 128, 235 et 255.

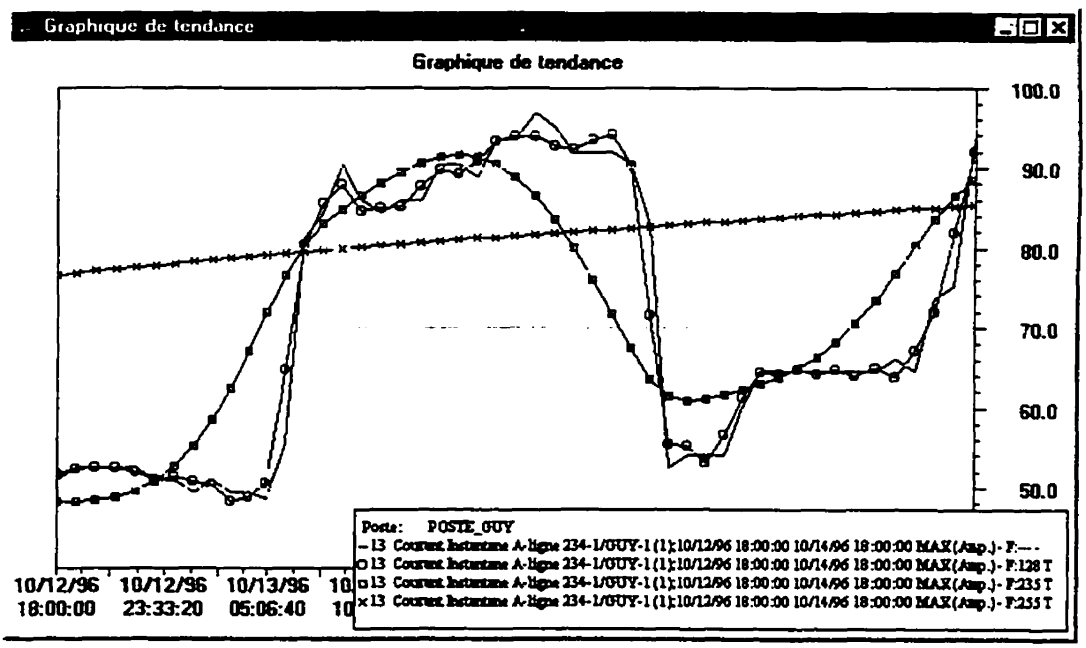


Figure 7.18: Valeurs de filtrage d'un graphique de tendances

7.1.6.6 Fenêtres de configuration de l'interface

En plus des fenêtres de sous-station de la section 7.1.2, plusieurs fenêtres permettent la configuration de l'interface. Quatre volets sont disponibles sur la fenêtre de configuration de l'interface.

1. ***Configuration de mot de passe***: permet à l'usager de changer son propre mot de passe. Permet aussi à un administrateur de système de changer d'identité (i.e. nom d'usager) pour fins de tests.

2. ***Liste des opérateurs***: ce volet (figure 7.19) de la fenêtre de configuration permet d'ajouter, de supprimer et de modifier les comptes d'usagers. Pour créer un usager, on doit fournir un nom d'usager ainsi qu'un mot de passe et niveau de priorité.

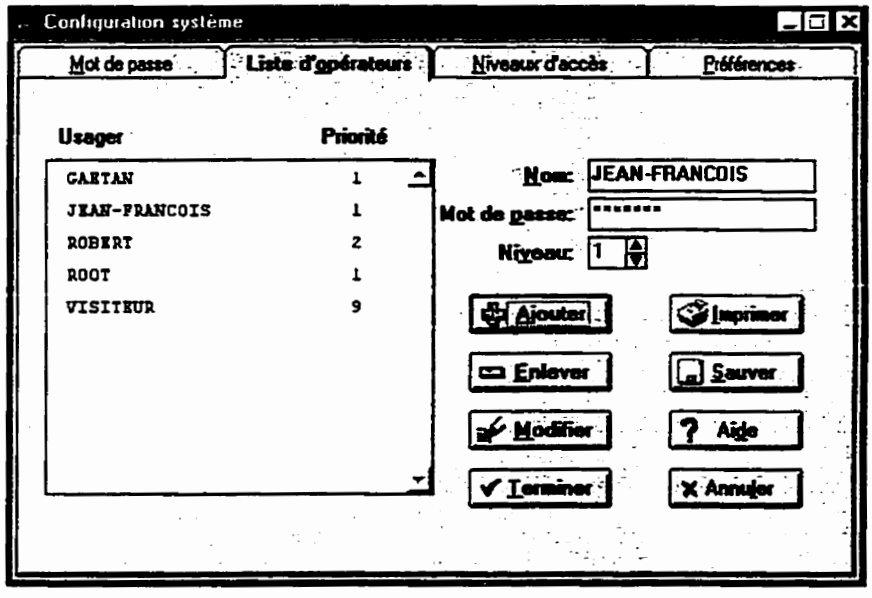


Figure 7.19: Configuration des opérateurs

3. ***Niveaux d'accès***: cette fenêtre (figure 7.20) présente les cinq chiffres de priorités (c.f. section 7.1.3) associés à chacune des fenêtres de l'interface usager et permet de les modifier. L'accès à cette fenêtre doit être réservée à l'administrateur.

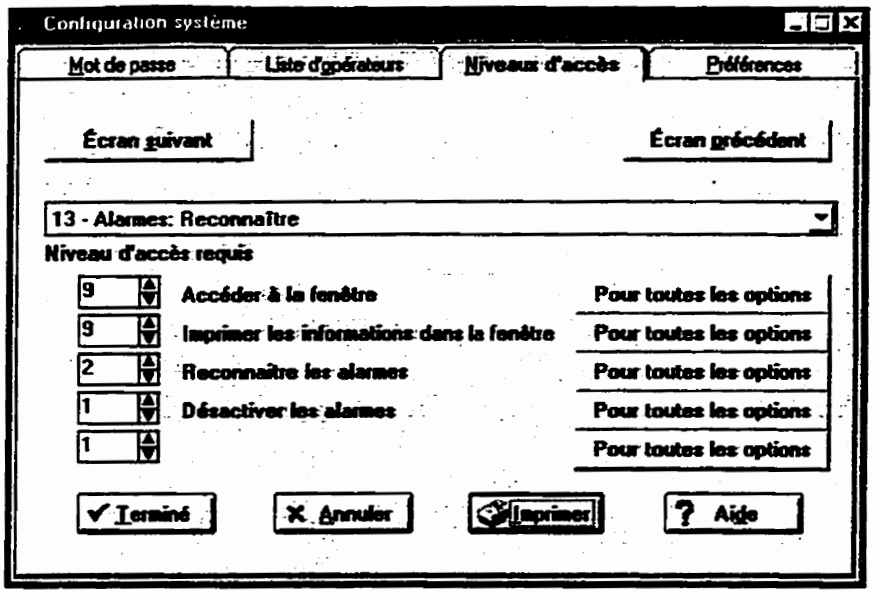


Figure 7.20: Configuration des niveaux d'accès

Les boutons "Pour toutes les options" permettent de fixer le niveau d'accès courrant pour toutes les fenêtres. Par exemple, si on appuie sur le permier de ces boutons, chaque permier chiffre associé aux fenêtres sera identique et égal à la valeur actuelle choisie pour la fenêtre courante.

4. ***Préférences***: cette section permet de configurer diverses options telles que le délai d'attente maximal lors d'une communication ou le nombre de points maximums qui seront utilisés dans un graphique.

7.2 Définition de la topographie des touches

L'utilisation de la souris, quoique très utile, n'est pas essentielle à l'utilisation de l'interface usager. Un usager peut se contenter des touches de raccourcis suivantes.

7.2.1 Définitions pour MS-DOS et Microsoft Windows

Tableau 7.2: Définition des touches pour MS-DOS et Microsoft Windows

Action	Touches	Description
Début de champ	<Ctrl+Home>	Déplace le curseur au début d'un champ ou au début d'une liste.
Fermeture de fenêtre temporaire	<Esc>	Ferme une fenêtre temporaire. Certaines fenêtres sont identifiées comme temporaire (le programmeur leur ajoute le fanion WOAF_TEMPORARY). Le menu temporaire le plus connu est le menu système utilisé dans les fenêtres.
Fermeture de fenêtre	<Shift+F4>	Ferme une fenêtre régulière.
Copier la sélection	<Ctrl+Ins>	Copie la sélection dans le tampon de copie. Cette option n'est valable que pour les champs qui peuvent être édités.
Couper la sélection	<Shift+Del>	Coupe la sélection dans le tampon de copie. Cette option n'est valable que pour les champs qui peuvent être édités.
Effacer	<Del>	Efface le texte sélectionné ou le caractère suivant. La valeur coupée n'est pas stockée dans le tampon de copie. Cette option n'est valable que pour les champs qui peuvent être édités.
Effacer le caractère précédent	<Backspace>	Déplace le curseur un caractère à gauche en effaçant le caractère directement à gauche. Cette option n'est valable que pour les champs qui peuvent être édités.
Effacer mot	<Ctrl+Del>	Positionne le curseur au début du mot à être effacé, et efface le mot et tous les espaces qui suivent. Le curseur reste au même endroit.
Déplacer vers le bas	<↓>	Déplace le curseur ou la sélection (dans une liste verticale, par exemple) vers le champ du bas, si le curseur est placé dans un champ multilignes.
Saut de page vers le bas	<PgDn>	Si le curseur est placé dans un champ multilignes et que le curseur n'est pas placé sur la dernière ligne, le curseur sera déplacé d'une page vers le bas.
Fin de champ	<Ctrl+End>	Déplace le curseur à la fin d'un champ qui peut être édité. Par exemple, dans un objet UIW_TEXT, déplace le curseur sur la dernière ligne du champ.
Fin de ligne	<End>	Déplace le curseur à la fin d'une ligne qui peut être éditée.
Fin de programme	<Alt+F4> ou <Shift+F3> ou <Ctrl+Break> ou <Ctrl+C>	Termine l'application.
Aide contextuelle	<F1>	Affiche l'aide contextuelle associée à la fenêtre courante.
Aide générale	<Alt+F1>	Affiche l'index d'aide du programme. Toutes les rubriques d'aide y sont dispobibles.

Action	Touches	Description
Début de ligne	<Home>	Déplace le curseur au début d'une ligne qui peut être éditée.
Déplacement à gauche	<←>	Déplace le curseur ou la sélection (dans une liste verticale, par exemple) vers le champ de gauche, si le curseur est placé dans un champ multilignes.
Mot de gauche	<Ctrl+←> ou <Alt+←>	Dans un champ de texte éditable, déplace le curseur au début du mot précédent à celui où est placé le curseur.
Marquer le texte	<Shift+←> et <Shift+→>	Dans un champ de texte, débute la sélection de texte à fin de copier, de couper ou de supprimer une section de texte.
Menu	<Alt> ou <F10>	Permet d'accéder au menu déroulant à partir de la fenêtre courante et d'y retourner. Cette combinaison n'est disponible que lorsque la fenêtre principale possède un menu déroulant.
Miniser	<Alt+–> ou <Alt+F9>	Minimise la fenêtre courante. Si la fenêtre est déjà minimisée, cette combinaison de clés la retourne à sa taille originelle. Seules les fenêtres pouvant être minimisées réagissent à cette commande.
Déplacer fenêtre	<Alt+F7>	Permet de déplacer la fenêtre courante. Pour déplacer la fenêtre, il suffit d'utiliser cette combinaison de touches, et d'utiliser les flèches pour déplacer la fenêtre. Une fois positionnée à l'endroit voulu, la touche <Enter> termine le déplacement.
Déplacement de champ	<Tab> ou <F6>(DOS)	Déplace le focus (i.e. la sélection du champ courant) vers le prochain champ disponible. Si l'objet est le dernier de la fenêtre, le focus se déplace sur le premier élément de la fenêtre.
Prochaine fenêtre	<Alt+F6>	Sélectionne la prochaine fenêtre disponible du gestionnaire de fenêtres.
Prochaine fenêtre MDI	<Ctrl+F6>	Sélectionne la prochaine fenêtre MDI disponible du gestionnaire de fenêtres.
Coller sélection	<Shift+Ins>	Colle la sélection située dans le tampon de copie. Cette option n'est valable que pour les champs qui peuvent être édités.
Champ précédent	<Shitf+F6> ou <Shift+Tab>	Déplace le focus (i.e. la sélection du champ courant) vers le champ précédent disponible. Si l'objet est le premier de la fenêtre, le focus se déplace sur le dernier item de la fenêtre.
Redessiner	<F5>	Rafraîchit l'affichage du programme.
Déplacement à droite	<→>	Déplace le curseur ou la sélection (dans une liste verticale, par exemple) vers le champ de droite, si le curseur est placé dans un champ multilignes.
Système	<Alt+Spc>	Sélectionne le menu système (s'il y en a un associé à la fenêtre) de la fenêtre courante.
Système (MDI)	<Ctrl+Spc>	Sélectionne le menu système (s'il y en a un associé à la fenêtre) de la fenêtre MDI courante.
Insertion/écrasement	<Ins>	Bascule entre le mode *insertion* et le mode *écrasement* dans un champ éditable.

Action	Touches	Description
Déplacement vers le haut	<↑>	Déplace le curseur ou la sélection (dans une liste verticale, par exemple) vers le champ du haut, si le curseur est placé dans un champ multilignes.
Saut de page vers le haut	<PgUp>	Si le curseur est placé dans un champ multilignes et que le curseur n'est pas placé sur la première ligne, le curseur sera déplacé d'une page vers le bas.

7.2.2 Définition des touches OSF/MOTIF

Les touches pour cet environnement sont entièrement configurables, et dépendent des caractéristiques du fichier `.Xressources` de l'utilisateur. Cependant, elles seront sûrement configurées d'une manière similaire à la configuration de MS-Windows.

7.3 Implantation de l'interface

L'interface usager a été codée en C++ en utilisant la bibliothèque Zinc Application Framework 4.2 comme système de fenêtrage. Ce système nous permet d'avoir du code d'interface transportable sur diverses plate-formes. Ce chapitre explique l'architecture logicielle de l'interface, tout en y expliquant ses détails d'implantation concrets.

7.3.1 Conventions de codage

Afin d'avoir un code homogène et facilement modifiable, plusieurs conventions de codage ont été adoptées quant aux noms d'identificateurs et à la structure arborescente des fichiers source.

Les fichiers source de l'interface sont situés dans une série de répertoires; chaque répertoire regroupe les fichiers de même utilité. Chacun de ces fichiers source ne contient qu'une seule définition d'objet. Les répertoires et leur utilité sont décrits à la figure suivante. Veuillez noter la brièveté des identificateurs (8 caractères maximum) pour conserver la compatibilité avec les systèmes DOS.

Tableau 7.3: Hiérarchie des fichiers source

Niveau 1	Niveau 2	Description
./		Le fichier principal ("main"), les fichiers de données pour les fenêtres (voir la section 7.3.4 sur l'utilisation des outils de la bibliothèque) et de l'interface sont placés dans ce répertoire.
	Comm	Contient les fichiers permettant la communication de l'interface aux contrôleurs centraux. Les communications série, ethernet et démo sont couverts dans ces fichiers. Puisque Zinc Application Framework ne supporte pas les communications, chaque plate-forme supportée possède ses propres fichiers.

Niveau 1	Niveau 2	Description
	`Graph`	Les fichiers relatifs aux graphiques sont contenus dans ce répertoire. Zinc Application Framework ne supporte pas directement les fichiers. Les objets "Ligne" et "Graphique" ont donc été décrits à partir des primitives de graphisme disponibles.
	`Include`	Tous les fichiers entête de chacun des fichiers source sont contenus dans ce répertoire. Ces fichiers se terminent par l'extension ".hpp".
	`Managers`	L'interface comporte plusieurs gestionnaires, qui sont des instances statiques (i.e. disponibles à tous les objets et qui sont uniques), effectuant plusieurs tâches, telle la gestion des erreurs, la gestion de l'impression, et le stockage sur disque. Ces objets se trouvent dans ce répertoire.
	`Objects`	L'interface manipule aussi des objets représentants des concepts abstraits, n'étant pas reliés directement à l'aspect informatique de l'interface: les objets sous-stations, les chaînes d'initialisation de modem et le temporisateur sont contenus dans ce répertoire. Plusieurs autres objets, tels la chaîne de caractère et le vecteur de bits y sont aussi stockés.

	`Sysdepn`	Les communications dans Zinc Application Framework ne supportent pas tous les aspects de la programmation multi plate-forme. Les aspects n'étant pas couverts par Zinc et ne faisant pas partie des communications se trouvent ici. Les fichiers sources sont une série de
		`#if defined (MS_WINDOWS)...`
		`#elif defined (UNIX) ...`
		Chaque aspect doit être réécrit au complet selon sa plate-forme.

	`Ui_win`	Chacune des fenêtres de l'interface est un objet. Tous les fichiers contenant les descriptions des différentes fenêtres sont situés dans ce répertoire.

Niveau 1	Niveau 2	Description
	`Windows`	Microsoft Windows requiert quelques fichiers pour bien compiler le fichier de définition des ressources, les fichiers d'icônes et le fichier de définition de l'exécutable. Les fichiers spécifiques à cette plateforme sont contenus dans ce répertoire.
	`Unix`	Certains fichiers, tels le fichier de création de l'exécutable, le "Makefile", et d'autres utilitaires propres à la création de l'exécutable UNIX sont placés dans ce répertoire.

Pour rendre le code le plus accessible et le plus compréhensible possible, des conventions ont été fixées quant à la nomenclature des identificateurs utilisés. Chacun des identificateurs est constitué d'un mot ou d'une phrase significative représentant la nature même de l'utilisation de l'identificateur. La langue internationale de l'informatique, l'anglais, est la langue choisie pour les identificateurs.

Tableau 7.4: Conventions de codage selon le type d'identificateur

Types d'identificateurs	Description de la convention
Types et classes	Toutes les lettres dans les noms de types et de classes sont des lettres majuscules. Chaque mot composant un identificateur de ce type ou classe est séparé par le caractère souligné (i.e. '_'). exemple: `CLASSE_QUELCONQUE`
Variables locales	Elles commencent par une minuscule. Les mots formant l'identificateurs sont concaténés, sans qu'un caractère ne les sépare. Chaque mot suivant le permier commence par une lettre majuscule. Exemple: `variableLocaleNumeroUn`

Types d'identificateurs	Description de la convention
Variables globales	Les variables globales suivent les mêmes règles que les variables locales, à l'exception près qu'elles sont précédées du caractère souligné. exemple: `_variableGlobaleNuméroUn`
Noms de fonctions et de méthodes	Les noms de fonctions et de méthodes ont le même format que les variables locales, sauf que la première lettre de l'identificateur qui est une lettre majuscule. exemple: `NomDeFonctionOuMéthode`

En plus des variables créées par le programmeur à l'intérieur des fichiers de code, certaines variables doivent être définies dans l'outil de développement de Zinc (section 7.3.4). Ces variables suivent la convention suivante, notée selon le formalisme de Baccus–Naur (BNF):

[EV_]<*type*>_<*fenêtre*>_[<*sous-fenêtre*>_]*<*description*>

1. [EV_]: ce préfixe est ajouté si l'objet identifié est apte à générer un événement dans le système de gestion des événements (méthode `Event` associée aux fenêtres);
2. <*type*>: identifie le type d'objet associé à l'identificateur. Ce type est un des codes de deux ou trois lettres suivants:

Tableau 7.5: Préfixe des identificateurs utilisés dans le Zinc Designer

Nom du type	Description de l'objet associé
BTN	bouton traditionnel à un ou deux états
FLD	champ d'entrées de données pour du texte
CMB	boîte combo ("*drop down combo*"), interactive ou non
WND	fenêtre MDI
RB	bouton radio
CB	bouton "check box"

Nom du type	Description de l'objet associé
PRM	texte descriptif apparaissant dans une fenêtre
NB	objet composite constitué d'onglets et de fenêtres, du style cahier de notes
GR	groupe servant à isoler des boutons radio d'autres boutons
SPN	"*Spin Box*"
ICN	icône graphique
VTL	liste verticale, avec une barre de défilement verticale à droite

3. <*fenêtre*>: une ou deux lettres décrivant la fenêtre sur laquelle est attaché l'objet associé à l'identificateur;

4. <*sous-fenêtre*>: si l'objet est placé sur une sous-fenêtre, nom de la sous fenêtre associée. L'astérisque, selon la notation BNF, indique qu'il peut y avoir autant de noms de sous-fenêtres qu'il y a effectivement de sous-fenêtres;

5. <*description*>: la description de l'objet identifié en tant que tel.

7.3.2 Choix du langage et de la bibliothèque

Plusieurs bibliothèques multi plate-formes sont offertes sur le marché. Les principaux produits lors du début du projet étaient:

1. *Galaxy*: une bibliothèque très complète. Elle supporte les applications multi plate-formes, et non seulement les interfaces multi plate-formes. Comporte des bibliothèques pour les communications, les graphiques, le système de fenêtrage, et plusieurs autres ressources. Quelques groupes chez Hydro–Québec sont relativement expérimentés avec cette bibliothèque, donc des ressources en cas de problème sont disponibles. La bibliothèque est compilée pour SunOS et Windows NT. Elle semble très

intéressante malgré la courbe d'apprentissage assez élevée. Cependant, son prix la classe hors de portée: de 10 000 $ à 15 000 $ pour une trousse de développement;

2. *XVT*: quelques personnes du Service Appareillage électrique ont eu l'opportunité d'utiliser ce produit, et leurs commentaires étaient peu élogieux. Il semble que cette bibliothèque relève plus d'une bibliothèque pour interface Microsoft Windows à la manière de ObjectWindows de Borland que d'une bibliothèque multi plate-forme. Son prix était de 350 $.

3. *zApp*: à l'époque, produit relativement jeune. Peu de gens avaient eu la chance de travailler avec de produit, et ils en étaient dans leurs premières versions. La fiabilité était donc à établir. Le prix était d'environ 750 $.

4. *Zinc Application Framwork*: support de plusieurs plate-formes pour les interfaces. Ne comprend cependant ni les graphiques, ni les systèmes de communications. Le produit est dans sa quatrième version, et le code source entier de la bibliothèque est disponible. Le service technique téléphonique est disponible sans frais pour les six premiers mois, et gratuit pour une période indéterminée par courrier électronique. Son prix se chiffre entre 500 $ à 3 000 $ dépendamment des plate-formes choisies. Elle a donc été choisie pour ses qualités logicielles, son prix et pour le support aux usagers.

7.3.3 Architecture de Zinc Application Framework

La bibliothèque Zinc Application Framework est constituée de composantes traitant une foule de tâches spécifiques. Cette infrastructure, illustrée à la figure 7.21, fonctionne sur un système à gestion de messages générés par des événements.

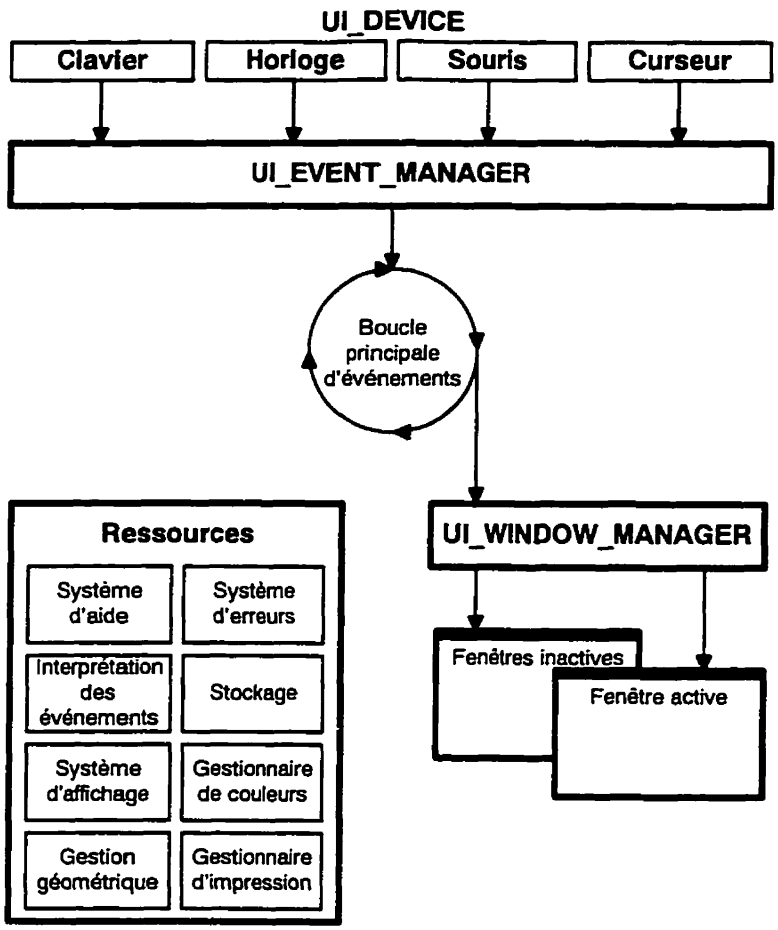


Figure 7.21: Architecture de la bibliothèque Zinc Application Framework

Chacun de ces objets à une tâche spécifique:

- **UI_EVENT_MANAGER**: le gestionnaire d'événements. Il gère le flot des événements à travers le système. Ces événements proviennent de différents dispositifs de l'ordinateur (i.e. "*devices*"). Les événements reçus sont placés dans une file d'attente, jusqu'à ce qu'ils puissent être répartis au gestionnaire de fenêtres. Le gestionnaire traite les événements un à la fois, jusqu'à ce qu'il reçoive le message de fermeture de l'application. Sans entrer dans tous les détails, regardons à titre d'exemple le cas où un usager appuie sur <Alt–F4>, le message de fermeture d'application:

le clavier génère un événement "touche appuyée" avec comme paramètre la valeur de la touche ou de la combinaison de touches, soit <Alt–F4>. Ce message est transmis au gestionnaire d'événements, qui le place en attente, prêt à être ramassé par le gestionnaire de fenêtres quand celui-ci sera libre. Le gestionnaire de fenêtres interprète ce message de fermeture, et active les routines de fin de programme;

- **UI_DEVICE**: tous les dispositifs d'entrée du système sont dérivés de cette classe abstraite, représentée à la figure 7.22. Elle définit le comportement général d'un dispositif d'entrée, sans en spécifier les détails d'implantation. Elle indique, par exemple, qu'un dispositif doit avoir un état, un type, une fonction pour aller chercher l'événement sur le matériel (à la manière d'une boucle infinie sur l'instruction "`getchar`" pour le clavier), une fonction pour traiter les événements, etc. Notons que selon la définition du langage C++, cette classe ne peut être directement instanciée: une autre classe doit être dérivée de celle-ci pour obtenir l'instance.

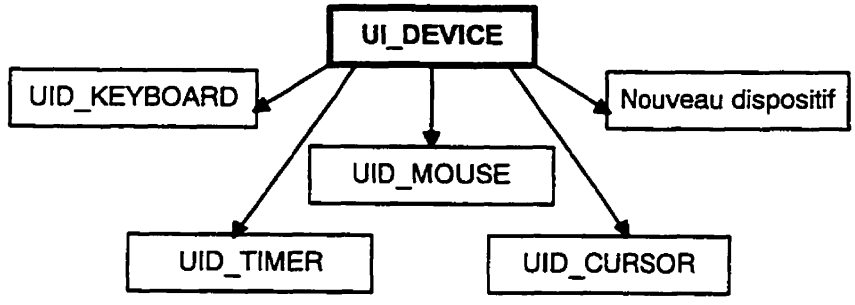


Figure 7.22: Définition de la classe abstraite UI_DEVICE

En plus des dispositifs mis à sa position, il est possible de dériver de nouveaux dispositifs d'entrée, selon les besoins de l'utilisateur. On peut, par exemple, créer un nouveau dispositif comme une carte d'acquisition, un écran tactile ou un stylet.

- **UI_EVENT**: ce sont les événements qui transitent dans le système, selon la direction des flèches de la figure 7.21. Ici, Zinc diffère de plusieurs bibliothèques d'affichage traditionnelles dans son comportement quant à la traduction des événements en pro-

venance du matériel. Les bibliothèques traditionnelles ont souvent tendance à fournir les informations en provenance des dispositifs d'entrée directement dans leurs boucles d'événement. Par exemple, la bibliothèque CXL envoie l'événement L_SELECT dans la boucle d'événements aussitôt que le bouton gauche de la souris est enfoncé. Le programmeur doit alors vérifier dans quel contexte cet événement a eu lieu avant d'agir en conséquence.

Zinc, cependant, effectue déjà l'interprétation de l'événement selon le contexte, libérant ainsi le programmeur d'une partie de la tâche, lui permettant ainsi de se concentrer sur la fonctionnalité de son programme plutôt que sur la gestion des événements. Prenons l'exemple d'un programme ayant une fenêtre, un espace pour du texte et des boutons, tel qu'illustré à la figure 7.23.

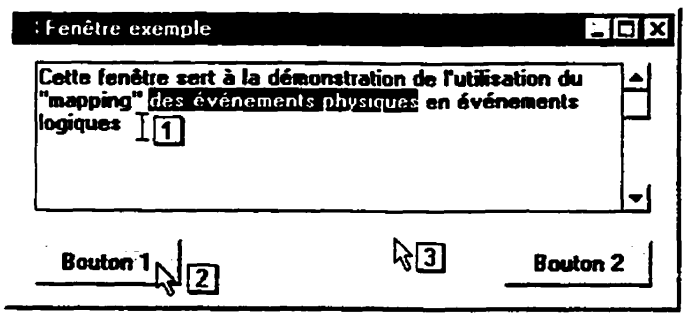


Figure 7.23: Mappage des événements physiques en événements logiques

Lorsqu'un usager appuie sur le bouton de gauche de la souris à l'endroit [1], le curseur se change en curseur de sélection de texte, et l'événement *L_BEGIN_MARK* est envoyé à l'objet texte, indiquant le début d'une sélection de texte. Lorsque l'usager appuie sur le bouton de gauche à l'endroit [2], l'événement *L_SELECT* est envoyé au bouton. Si finalement l'usager appuie à l'endroit [3], l'événement *L_BEGIN_SELECT* est envoyé à la fenêtre. On remarque donc qu'un seul événement génère différents messages dépendamment de l'endroit où la souris est placée:

Zinc effectue un "mapping" des événements dépendamment de l'objet vers lequel il est envoyé. Cette méthode comporte plusieurs avantages:

1. elle nous permet de créer de nouveaux dispositifs d'entrée utilisés avec leurs propres messages d'événements non traduits, tout en utilisant les messages traduits pour les autres dispositifs;
2. cette méthode nous permet aussi la redéfinition des touches de clavier, particulièrement les touches de fonctions, sans avoir à modifier la bibliothèque;
3. la plus grande utilité de cette méthode est de permettre la portabilité du système d'événements sur plusieurs systèmes de fenêtrage. Puisque Zinc permet à chaque objet de réagir différemment, un objet a le loisir de réagir différemment selon le système d'exploitation. On peut alors assigner un comportement selon le type d'objet et selon le système d'exploitation, au lieu de forcer l'objet à réévaluer son comportement selon le contexte de l'environnement de fenêtrage.

- **UI_WINDOW_MANAGER**: le gestionnaire de fenêtrage. Ce gestionnaire détermine la façon dont les fenêtres et les objets de fenêtres réagissent, lorsqu'ils reçoivent des événements en provenance du gestionnaire d'événements.

  La difficulté d'une telle tâche réside dans le fait que les différents systèmes d'exploitation supportés par la bibliothèque ont des architectures qui diffèrent grandement. Certains, dont Microsoft Windows, fonctionnent sur le principe de "bottom up message passing", et certains autres, tel MOTIF, fonctionnent en "top down message passing". De plus des environnements comme DOS n'ont aucun moyen de base qui leur permet de gérer le flot des événements en provenance de dispositifs d'entrée.

  Au lieu de privilégier une des deux architectures et de traduire lorsque le système de fenêtrage ne supporte pas l'architecture choisie, Zinc s'assure que lorsqu'un pro-

gramme tourne dans un système de fenêtrage, qu'il réagisse selon le mode de fenêtrage natif. Dans le cas de systèmes n'ayant pas de moyen de gestion des événements, une infrastrutcure "top down" est implantée par Zinc comme étant l'architecture spécifique.

Le gestionnaire de fenêtres conserve une liste des fenêtres actives, des fenêtres inactives et des fenêtres minimisées. Il détermine aussi leur position et dirige les messages aux bonnes fenêtres. Pour se faire, il possède sa propre boucle d'événements, appelée UI_WINDOW_MANAGER::Event( ). Si par exemple, deux fenêtres se recoupent, comme le montre la figure 7.24, et que des événements en provenance du clavier arrivent au gestionnaire de fenêtre, les messages iront à la fenêtre 1 puisqu'elle a priorité sur la fenêtre 2. Si des événements proviennent de la souris, ils iront à la fenêtre dans laquelle le pointeur de souris est placé. Si le pointeur est placé dans la zone de superposition, les messages iront aussi à la fenêtre 1 à cause de sa priorité sur la fenêtre 2.

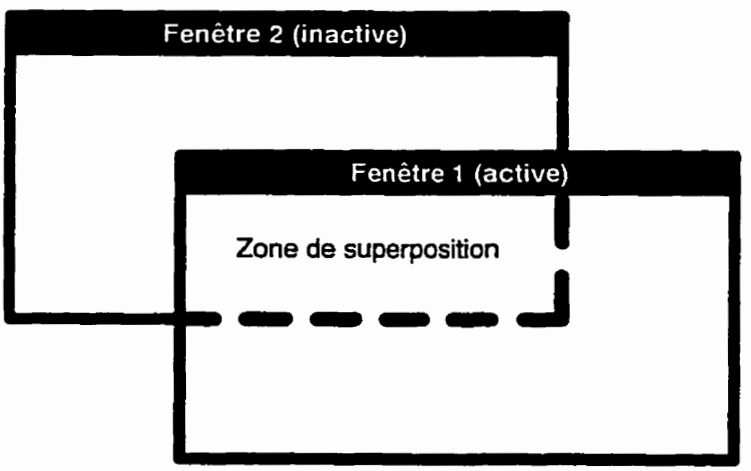


Figure 7.24: Passage de message aux fenêtres selon leur priorité

Notons que tous les objets aptes à recevoir des messages, dont font partie les fenêtres, sont des objets hérités de la classe UI_WINDOW_OBJECT. Ceci veut dire que

les fenêtres et les objets qu'ils contiennent ont plusieurs caractéristiques communes, dont celle d'apparaître au système de fenêtrage comme étant des objets issus de l'environnement. En effet, afin d'avoir une interface performante, Zinc n'émule pas l'apparence et le comportement des objets issus de l'environnement: il utilise carrément les objets de l'environnement, contrairement à la majorité des systèmes de fenêtrage portables. C'est-à-dire qu'au lieu d'accéder à des primitives de graphisme pour créer un bouton sous Microsoft Windows et d'y associer une série de primitives simulant le comportement d'un bouton Windows, Zinc demande directement à Windows d'afficher un bouton ayant les caractéristiques voulues;

- **Système d'affichage**: puisque Zinc fonctionne dans divers environnements, une infrastructure orientée objet lui permettant de faire l'affichage sur tous les systèmes d'exploitation sans modifications au code de l'application a été conçue. La base de ce système d'affichage est supportée par l'objet UI_DISPLAY, à la manière de la classe UI_DEVICE. De cette classe virtuelle sont dérivées les diverses autres classes nécessaires, contenant toutes les primitives d'affichage de bas niveau.

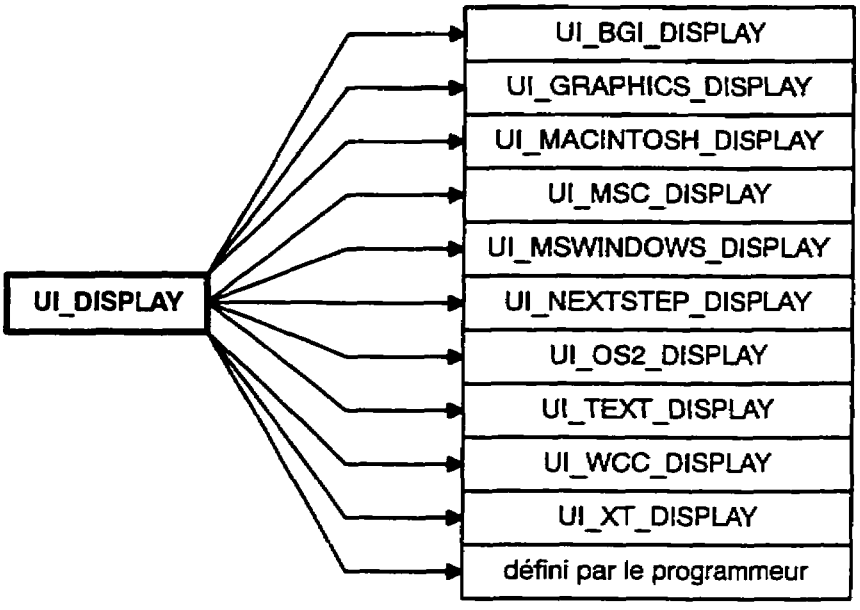


Figure 7.25: Définition de la classe abstraite UI_DISPLAY

Ainsi, le programmeur n'a pas à se soucier de la programmation du système d'affichage. Lors du chargement du programme, Zinc s'occupe de détecter le type d'affichage nécessaire, et d'instancier l'objet UI_DISPLAY approprié.

- **Systèmes d'aide:** cette composante est optionnelle et peut être ignorée du programmeur. Cependant, UI_HELP_SYSTEM nous permet d'avoir un système d'aide avec index automatique transportable d'une plate-forme à l'autre sans qu'une modification n'ait à être effectuée. Ce système d'aide n'est cependant pas aussi sophistiqué que celui fourni par Microsoft Windows, mais sa simplicité est à la fois éloquente et élégante.

- **Système de gestion des erreurs:** Zinc fournit aussi un système de gestion des erreurs, appelé UI_ERROR_SYSTEM. Tout comme la ressource précédente, il est aussi optionnel. Il permet l'affichage automatique d'erreurs conventionnelles telles l'absence d'un fichier nécessaire au fonctionnement du programme ou d'une erreur

d'entrée dans un champ de données. Ce système est toutefois limité et nécéssite l'ajout de primitives d'erreurs propres au programme développé;

- **Stockage**: Zinc fournit des primitives de stockage sous deux formes. La première forme consiste en un objet ZIL_FILE, étant simplement un objet de type fichier traditionnel et transportable sur les diverses architectures. L'objet ZIL_FILE fournit les primitives d'écriture, de lecture, de positionnement, d'ouverture et de fermeture de fichiers. Le second mode s'effectue à travers l'objet ZIL_STORAGE. Cet objet est mieux représenté par un système de fichiers (*"file system"*) de type UNIX dans lequel on retrouve une hiérarchie répertoires/fichiers/données. Tout comme les systèmes de fichiers traditionnels ont leurs opérations de création et de destruction de fichiers et de répertoires, l'objet ZIL_STORAGE possède aussi les siennes. Plusieurs autres primitives sont aussi disponibles pour obtenir la liste des fichiers d'un répertoire, le contenu d'un fichier, etc.

- **Gestionnaire de couleurs**: l'objet UI_PALETTE_MAP permet d'effectuer la correspondance des couleurs du système d'exploitation aux couleurs utilisées par l'application Zinc.

- **Gestionnaire géométrique**: ce gestionnaire dicte l'apparence visuelle des fenêtres lors des changements de tailles des fenêtres ou des écrans et permet de s'assurer que les fenêtres utilisées auront la même apparence dans les différents systèmes d'exploitation utilisés, malgré les différences d'apparences visuelles des objets issus de l'environnement de fenêtrage. Il permet par exemple d'aligner plusieurs objets dans une fenêtre, ou de spécifier le positionnement relatif (en pourcentage) d'un objet dans une fenêtre lors du changement de taille de la fenêtre parent.

- **Gestionnaire d'impression:** ce gestionnaire met à la disposition du programmeur des primitives permettant l'impression du contenu d'un écran ou d'une fenêtre, directement à une imprimante à travers le gestionnaire d'impression du système de fenêtrage installé. En DOS, n'ayant aucun gestionnaire d'impression de base, Zinc permet l'envoi de fichiers PCL directement aux imprimantes. Il permet aussi d'effectuer une impression dans un fichier PostScript.

Comme nous l'avons vu précédemment à la section sur le gestionnaire d'événements, Zinc est un système de fenêtrage à base d'événements, ce qui veut dire que le coeur des programmes écrits avec Zinc sont des boucles tournant en arrière plan, recevant divers événements et les répartissant aux endroits appropriés. Nous discuterons dans les deux sections suivantes de la façon dont se produit la gestion des événements selon le type d'environnement. La configuration suivante servira dans l'explication des deux modes de gestion d'événements. La figure 7.26 présente deux fenêtres; l'une d'elles, la fenêtre active, a comme objets un objet groupe, trois objets bouton-radio à l'intérieur de ce groupe et un objet bouton. La fenêtre inactive n'a aucun objet enfant.

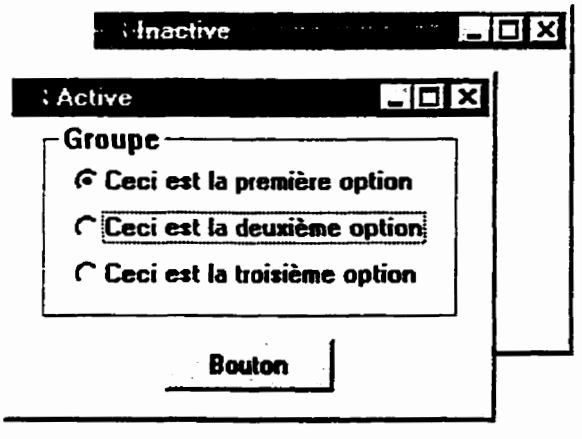


Figure 7.26: Fenêtres servant à l'illustration des deux architectures

Le premier bouton radio est sélectionné alors que le second est actif.

7.3.3.1 L'architecture descendante

Cette implantation est celle qui semble la plus simple et la plus naturelle. Dans cet environnement, le gestionnaire d'événements reçoit les événements des dispositifs d'entrée tels le clavier, la souris et le système de fenêtrage lui-même, pour ensuite les placer dans la file d'attente des événements. La boucle principale d'événements prend les événements dans la file et les répartit à son tour vers gestionnaire de fenêtres. Celui-ci effectue un routage de ces événements à travers sa fonction `Event( )` vers le premier objet enfant. L'objet destinaire détermine alors s'il est en mesure de traiter l'événement. S'il le peut, il effectue alors le traitement requis par l'événement, pour ensuite retourner le contrôle du programme à la boucle principale. S'il ne peut traiter cet événement, il transmettra l'événement à la fenêtre courante. La fenêtre courante aussi possède sa propre fonction `Event( )`, puisqu'elle est dérivée de l'objet UI_WINDOW_OBJECT. À l'aide de cette fonction `Event( )`, elle déterminera si elle est en mesure de traiter l'événement. Sinon, elle tentera de passer le message aux niveaux hiérarchiques inférieurs, et si oui, elle effectuera le travail requis pour retourner le contrôle à la fonction principale. Le tout continue jusqu'à ce qu'un objet puisse accepter l'événement ou que l'événement atteigne un objet n'ayant pas d'objets enfants.

Chacun des objets enfants d'un objet parent est lié par une liste doublement chaînée, à travers les pointeurs `next` et `previous`. De plus, le parent maintient des pointeurs sur le premier objet enfant (`first`), le dernier objet enfant (`last`) et l'objet courant (`current`). Lorsque le message est passé à un étage inférieur, il n'est toujours passé qu'à l'objet courant.

Dans l'exemple de la figure 7.26, si l'usager appuie sur la touche <Barre d'espacement>, pour sélectionner l'élément courant, la transmission des événements suit celle décrite à la figure 7.27.

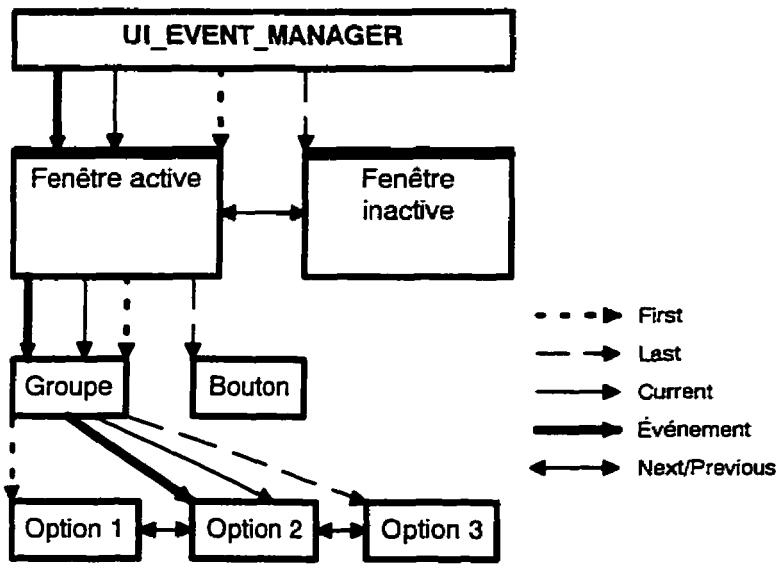


Figure 7.27: Traitement de l'exemple de l'architecture descendante

Le message est envoyé à la fenêtre active. Celle-ci ne sachant répondre à l'événement, le passe au groupe qui est l'objet courant de sa liste d'objets enfant. Si le groupe ne sait comment traiter l'événement, il le transmet au bouton-radio 2, l'objet courant, qui lui traite l'événement et retourne le contrôle à la boucle principale.

7.3.3.2 L'architecture ascendante

L'implantation de l'architecture "bottom up", quoique normalement plus rapide à l'exécution que l'architecture "top down" est quelque peu plus complexe puisqu'il fait intervenir le système de fenêtrage directement. Sa différence se situe au niveau du gestionnaire de fenêtres. Comme dans l'architecture précédente, les événements générés sont envoyés au gestionnaire d'événements, pour être placés dans une file d'attente avant d'être répartis au gestionnaire de fenêtres. Toutefois, dans le gestionnaire d'événements, une sélection

s'opère sur les événements. Le gestionnaire de fenêtres distingue les événements issus du système de fenêtrage et ceux de Zinc ou du programmeur. Ceux qui ne sont pas issus du système suivent le même chemin que les événements dans le mode "top down". S'ils sont spécifiques à l'environnement, ils suivront cependant un chemin complètement différent.

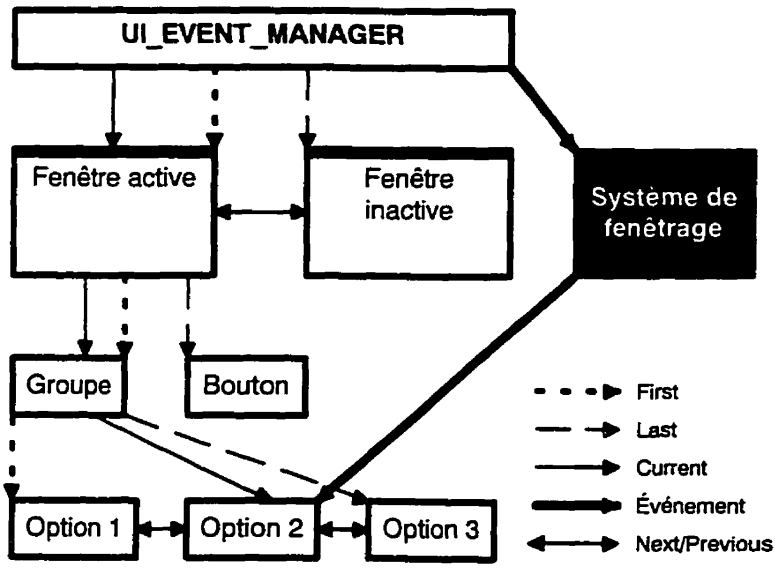


Figure 7.28: Traitement de l'exemple de l'architecture ascendante

Les événements sont alors envoyés directement au système d'exploitation, qui l'envoie directement à l'objet en question. Si cet objet ne peut répondre à l'événement, alors le système route le message au parent de l'objet de bas niveau. Dans notre exemple, le bouton 2 peut répondre au message envoyé. S'il n'avait été apte à réagir au message, l'événement aurait été envoyé au groupe, puis à la fenêtre active, pour finalement être perdu si aucun de ces éléments de la chaîne n'avait su qu'en faire.

7.3.4 Utilisation des outils de la bibliothèque

Plusieurs petits utilitaires sont fournis avec Zinc, mais le logiciel principal est sans contredit l'interface de création de fenêtres, le "Designer". Cet éditeur de ressources permet la création de fenêtres de manière visuelle à l'aide de la souris et permet d'éditer des images et icônes utilisées dans la bibliothèque, de créer le système d'aide contextuel et d'internationaliser une application à partir de gestionnaire de langages et de "locales" (informations de formatage de nombre propre à un pays ou une région comme le format de date, le format monétaire, le format des nombres, etc.).

Tableau 7.6: Utilitaires fournis avec Zinc Application Framework

Nom	Utilité
BROWSE	Permet de lister le contenu des fichiers ZIL_STORAGE.
CHKFILE	Vérifie l'intégrité des fichiers ZIL_STORAGE.
CPPATH	Copie un répertoire d'un fichier de type ZIL_STORAGE à un autre.
GC	Effectue la fonction de ramasse-miettes ("*garbage collector*") sur un fichier ZIL_STORAGE. Quand un enregistrement est supprimé dans un de ces fichiers, ils ne sont que marqués pour la destruction. Cet utilitaire les efface vraiment, réduisant ainsi la taille du fichier.
RRMDIR	Détruit un répertoire et son contenu dans un fichier ZIL_STORAGE.
STRIP	Enlève les informations nécessaires à la compilation et au déverminage dans les versions de production des objets ZIL_STORAGE.
ZDUMP	Permet de voir tout le contenu d'un fichier situé dans le pseudo-système de fichiers ZIL_STORAGE. Affiche l'information sous forme hexadécimale et sous forme de chaîne de caractères.
DESIGNER	Éditeur de ressources. Cette section est consacrée à son utilisation.

La principale utilité du "Designer" est de permettre la création de fenêtres rapidement et sans avoir à coder quoi que ce soit. Pour se faire, les fenêtres sont stockées dans un objet de type ZIL_STORAGE, et sont lues à l'exécution et au besoin.

Pour créer une nouvelle fenêtre complète, il ne suffit que de demander à l'éditeur de créer une fenêtre vierge, et de prendre les objets nécessaires sur la barre d'outils et de les faire glisser sur la nouvelle fenêtre à l'endroit désiré.

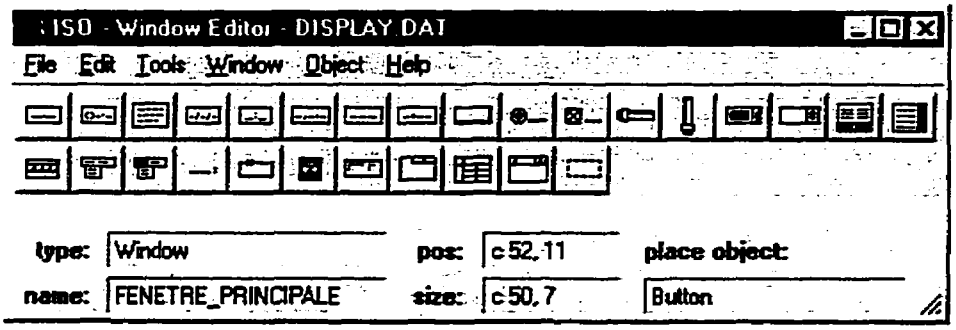
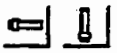


Figure 7.29: Fenêtre principale du Zinc Designer

Pour modifier les objets, il suffit d'appuyer deux fois sur le bouton de gauche de la souris en pointant l'objet à éditer, et une boîte de dialogue apparaît, donnant les caractéristiques de l'objet. On peut ainsi créer une foule d'objets standards qui à l'exécution deviennent des objets issus du système de fenêtrage. La figure 7.30 présente la majorité des objets disponibles; cependant, par manque d'espace sur la fenêtre, les objets "Notebook", fenêtre, table et réglette de déplacement vertical n'ont pas été inclus. On doit aussi noter que certains objets, comme le bouton système ou l'icône de minimisation, ne sont disponibles qu'en double-cliquant la fenêtre pour en éditer son contenu. Le tableau 7.7 résume les divers objets disponibles directement à partir du Designer, elle explique brièvement leur utilité et donne l'image associée à l'objet en question.

Tableau 7.7: Liste et utilité des objets disponibles dans Zinc Designer

Nom	Classe	Sous classe
UIW_STRING	Champs d'entrées	Chaînes quelconques
	*Description*: cet objet permet d'entrer tout forme de texte ASCII sans formatage particulier. Le champ peut aussi effectuer des traductions automatiques sur-le-champ, telles les traductions en lettres minuscules, en lettres majuscules, ou remplacement des caractères entrés par des astérisques, pour les mots de passe. Le texte entré peut être justifié à droite, au centre ou à gauche.	
UIW_FORMATTED_STRING	Champs d'entrées	Chaînes formatées
	*Description*: cet objet permet d'entrer tout forme de texte ASCII, avec un format prédéterminé. Le programmeur doit définir la forme de base, tel "(999) 999-9999". Lorsque l'usager entrera des chiffres, ils se placeront à l'endroit où sont placés les '9'. S'il entre des lettres, celles-ci ne seront pas acceptées dans l'exemple précédent. Il ne pourra non plus effacer les parenthèses, l'espace ni le signe '-'. Un autre exemple d'utilisation serait celui du code postal Canadien: "Z9Z-9Z9".	
UIW_DATE	Champs d'entrées	Chaînes de dates
	*Description*: cet objet permet d'entrer tout forme de texte ASCII, sous forme de date. Le format de la date est dicté par la variable de localité du système de fenêtrage.	
UIW_TIME	Champs d'entrées	Chaînes d'heures
	*Description*: cet objet permet d'entrer tout forme de texte ASCII, sous forme d'heure. Le format de l'heure est dicté par la variable de localité du système de fenêtrage.	
UIW_REAL	Champs d'entrées	Chaînes de nombres réels
	*Description*: cet objet permet d'entrer des chiffres et ponctuation ASCII, représentant des nombres réels à double précision. Les lettres et autres symboles ne sont pas acceptés. Une validation automatique peut être effectuée quant à la plage des données requises par l'usager. Contrairement à l'objet UIW_STRING, le programmeur peut aller chercher la valeur numérique dans avoir à utiliser des fonctions de traductions de chaînes à nombres réels tel `atof( )`, par exemple.	

Nom	Classe	Sous classe
UIW_INTEGER	Champs d'entrées	Chaînes de nombres entiers
	*Description*: cet objet permet d'entrer des chiffres ASCII, représentant des nombres entiers. Aucun symbole autre que les chiffres de 0 à 9 n'est accepté. Une validation automatique peut être effectuée quant à la plage des données requises par l'usager, et comme pour le champ UIW_REAL, le programmeur peut aller chercher la valeur entière sans avoir à traduire l'information à partir d'une chaîne.	
UIW_BIGNUM	Champs d'entrées	Chaînes de nombres quelconques
	*Description*: cet objet permet d'entrer des chiffres divers symboles tels 'e','$' et '%', afin de permettre l'entrée de nombres quelconques. Cette boîte d'entrée accepte différents formats, selon les besoin du programmeur: format décimal, format monétaire, format crédit (utilisant des parenthèses pour symboliser les nombres négatifs), format virgule (pour séparer les milliers avec des virgules) et format pourcentage. Une validation automatique peut être effectuée pour la plage des données requises par l'usager, et comme pour le champ UIW_REAL, le programmeur peut aller chercher la valeur entière sans avoir à traduire l'information à partir d'une chaîne.	
UIW_BUTTON	Boutons	Boutons, boutons radio, boutons "check–box", boutons à un ou deux états
	*Description*: cette classe permet la création de plusieurs types de boutons. D'abord, les boutons normaux, auquel on peut associer du texte et/ou une image imprimée sur le bouton, puis les boutons à deux états (du style "On/Off"), les boutons radio, dont un seul peut être sélectionné à la fois et finalement les boutons de type "check–box" du genre que l'on retrouve sur une liste de vérifications ou de choses à faire.	
UIW_SCROLL_BAR	Réglettes de défilement	
	*Description*: ces objets peuvent être attachés à des fenêtres ou à des boîtes de texte pour en permettre le défilement ou utilisés tel quels. Il en existe deux variantes: une horizontale et une verticale.	
UIW_COMBO_BOX	Liste tombante	
	*Description*: ces objets permettent la sélection d'éléments à travers une liste tombante. Ils ressemblent à un objet UIW_STRING avec une flèche à droite qui permet l'apparition de la liste.	

Nom	Classe	Sous classe
UIW_SPIN_CONTROL	Sélecteur à carrousel	
	*Description*: l'objet "spin control" ressemble à la liste tombante, lorsque sa liste n'est pas apparente. Il a un champ d'entrée de données, et à droite une flèche vers le haut et une vers le bas. L'usager incrémente ou décrémente la valeur affichée à l'aide des flèches. Le contenu de la boîte n'est pas limité aux chiffres: il peut être tout élément de la classe des champs d'entrées.	
UIW_VT_LIST et UIW_HZ_LIST	Listes verticales et horizontales	
	*Description*: ces listes permettent l'affichage de données sous forme de listes desquelles on peut sélectionner l'information et faire défiler l'information. Les listes verticales sont très appropriées pour l'affichage d'une grande quantité d'informations.	
UIW_TEXT	Zone de texte	
	*Description*: cet objet permet à l'usager d'écrire de l'information sous forme de mini traitement de texte, avec fonctions de copie et de collage.	
UIW_TOOL_BAR	Barre d'outils	
	*Description*: les barres d'outils sont des espaces où peuvent être placés des boutons et autres éléments fréquemment utilisés. Ils permettent d'accéder de façon intuitive aux fonctions ou fenêtres fréquemment utilisées d'un logiciel. Ils ne peuvent cependant apparaître que sur la fenêtre principale d'un programme.	
UIW_PULL_DOWN_MENU et UIW_PULL_DOWN_ITEM	Menus	
	*Description*: tout comme les barres d'outils, les menus ne peuvent être utilisés que sur la fenêtre principale. Ils sont les moyens d'accès à toutes les fonctions du logiciel. On peut aussi les utiliser comme menus flottants: ces menus peuvent apparaître à plusieurs endroits, contrairement aux menus conventionnels.	
UIW_PROMPT	Texte d'explication	
	*Description*: ces objets sont des chaînes de texte servant à expliquer l'utilité des objets d'une fenêtre.	
UIW_GROUP	Groupes	

Nom	Classe	Sous classe
	*Description*: cet objet sert surtout à isoler des boutons radio des autres objets. Comme on le sait, seul un bouton radio à la fois peut être sélectionné. Si on veut avoir deux groupes de boutons radio côte à côte, on doit utiliser ces groupes pour les isoler.	
UIW_ICON	Icônes	
	*Description*: ces objets sont principalement utiles comme icônes de minimisation des fenêtres. Ils peuvent toutefois être affichés dans une fenêtre pour en agrémenter le contenu.	
UIW_STATUS_BAR	Barre d'état	
	*Description*: tout comme la barre d'outils, cet objet ne peut se retrouver qu'à un seul endroit dans un programme, et c'est au bas de la fenêtre principale. On peut y installer tous genres d'objets, tels des objets UIW_STRING, pour afficher de l'information pertinente pour le programme.	
UIW_TABLE	Tableur	
	*Description*: il existe dans la bibliothèque un objet tableur similaire aux tableurs commerciaux, permettant l'entrée de données, leur affichage et les calculs. On peut aussi placer des titres verticaux et horizontaux afin de regrouper l'information des colonnes ou lignes en champs ayant une relation.	
UIW_WINDOW	fenêtres	fenêtre simple
	*Description*: il est possible à partir de l'interface de création de fenêtres de créer directement une fenêtre enfant à une fenêtre parent.	
UIW_NOTEBOOK	fenêtres	cahier de notes
	*Description*: les cahiers de notes sont en fait une série de fenêtres que l'on peut atteindre en sélectionnant l'onglet de la fenêtre voulue. On retrouve souvent ce type d'objet dans les panneaux de configuration de système de fenêtrages.	
UIW_IMAGE	Images	
	Description: Zinc permet l'affichage d'images propres au système d'expolitation à l'intérieur d'une fenêtre. Par exemple, des formats Bitmaps (BMP) peuvent être affichés sous Microsoft Windows et des formats Pixmaps (XPM) sont utilisables sous MOTIF.	

Nom	Classe	Sous classe
UIW_MINI-MIZE_BUTTON et UIW_MAXI-MIZE_BUTTON	objets de support	Boutons de minimisation de maximisation
	Description: ces premiers objets font partie d'une classe de support. Ils ne peuvent pas être rajoutés de la même façon que les objets précédents. Ils sont rajoutés automatiquement lorsque l'usager modifie les paramètres de la fenêtre sur laquelle il travaille. Tous les membres de la classe support sont optionnels. Ces boutons servent à minimiser et à maximiser la fenêtre qui les contient.	
UIW_SYS-TEM_BUTTON	objets de support	
	Description: le bouton système, lui aussi un objet de support, permet d'accéder à certaines fonctions mises à la disposition de l'usager via le système de fenêtrage. Ce bouton permet d'accéder à un menu contenant ces fonctions. Le contenu de ce menu varie quelque peu d'un système de fenêtrage à un autre, mais on y retrouve le plus souvent des fonctions permettant de déplacer, de redimensionner et de fermer la fenêtre.	
UIW_TITLE	objets de support	
	Description: chaque fenêtre est pourvue d'un titre affichée dans une barre en haut de la fenêtre. Cet objet définit le texte de ce titre.	
UIW_BORDER	objets de support	
	Description: si cet objet est présent sur la fenêtre, elle aura, selon le système de fenêtrage utilisé, une bordure plus épaisse qu'à l'habitude. L'usager pourra à l'aide de la souris modifier la taille de la fenêtre.	
UI_GEOME-TRY_MANAGER	objets de support	
	Description: afin d'avoir des fenêtres d'apparence identique d'un système de fenêtrage à un autre, il est possible de spécifier le positionnement relatif d'un objet par rapport à un autre. Ceci est nécessaire puisque les objets, tels les boutons, n'ont pas la même apparence d'un système de fenêtrage à un autre. Aussitôt que le programmeur spécifie des contraintes de positionnement relatif pour un objet, l'objet de support UI_GEOMETRY_MANAGER apparaît dans la liste des objets de la fenêtre.	

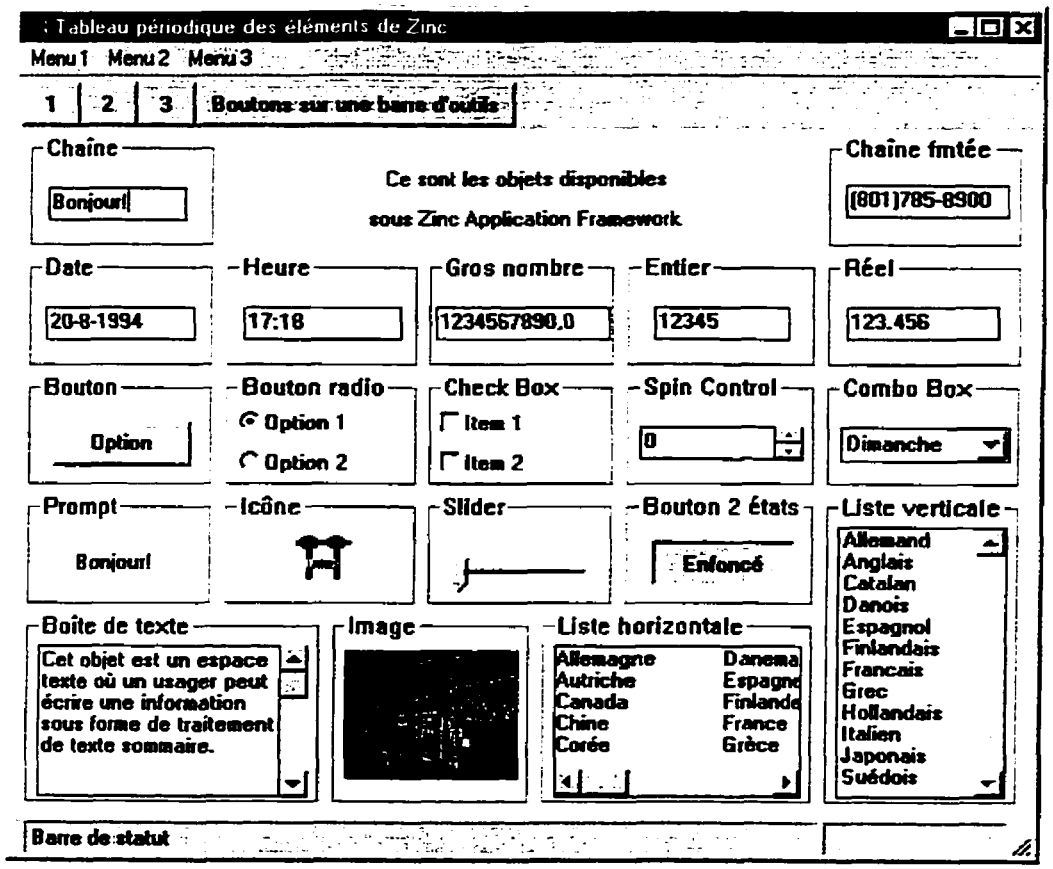


Figure 7.30: Objets dans la bibliothèque Zinc Application Framework

7.3.5 Programmation avec la bibliothèque Zinc

Nous avons vu qu'il est possible de créer des fenêtres et leurs composants à partir de l'éditeur de ressources Zinc Designer. Toutefois, il est aussi possible de créer à l'aide de lignes de code C++ les mêmes fenêtres, ou de rajouter des éléments aux fenêtres créées par l'éditeur de ressources avec du code. Nous verrons ici les rudiments de la programmation avec Zinc, soit la création d'un programme principal et la création d'une fenêtre parent et d'une fenêtre enfant. Veuillez noter qu'une certaine notion de programmation C++ est nécessaire à la compréhension de cette section.

Comme pour plusieurs bibliothèques de fenêtrage, la procédure `main( )` n'est pas la composante ni la plus imposante, ni la plus importante du programme. Cependant, il existe deux façons de créer cette fonction avec la bibliothèque:

1. Configuration manuelle: cette méthode permet d'utiliser la fonction `main( )` native à l'environnement de l'exécutable. Elle force aussi l'usager à créer des instances des diverses ressources dont il a besoin, comme le clavier, la souris, l'affichage, le gestionnaire d'événements, etc. Cette méthode est de beaucoup plus fastidieuse que la suivante, mais elle offre la flexibilité de remplacer les ressources de bases par des ressources qui auraient été modifiées selon les besoins du programmeur. Illustrons ceci par un exemple tiré du monde Microsoft Windows:

```
#include <ui_win.hpp>
int PASCAL WinMain( HANDLE hInstance, HANDLE hPrevInstance,
      LPSTR lpszCmdLine, int nCmdShow )
{
      // Initialisation du système d'affichage
      UI_DISPLAY *display = new UI_MS_WINDOWS_DISPLAY(
            hInstance, hPrevInstance, nCmdShow );

      /* Création du gestionnaire d'événements et des */
      /* dispositifs qui y seront rattachés           */
      UI_EVENT_MANAGER *eventManager
            = new UI_EVENT_MANAGER( display );
      *eventManager
            + new UID_KEYBOARD
            + new UID_MOUSE
            + new UID_CURSOR;

      // Création du gestionnaire de fenêtres
      UI_WINDOW_MANAGER *windowManager = new UI_WINDOW_MANAGER(
            display, eventManager );

      /* On retrouverait ici la création des fenêtres et leur */
      /* gestion, soit le corps même du programme             */

      // Terminé, on ramasse les miettes...
      delete windowManager;
      delete eventManager;
      delete display;
      return( 0 );
}
```

2. Configuration automatique:

```
#include <ui_win.hpp>
int UI_APPLICATION::Main( void )
{
      /* On retrouverait ici la création des fenêtres et leur */
      /* gestion, soit le corps même du programme             */

      return( 0 );
}
```

Comme on le remarque, le programmeur a avantage à utiliser la seconde méthode, puisqu'en plus d'être plus simple, elle est transférable d'un système d'exploitation à un autre. Dans le cas du premier exemple, le programmeur devrait faire une autre entête au `main( )` correspondant à celle appropriée pour son système d'exploitation.

L'utilisation de UI_APPLICATION simplifie la routine `main( )` en créant automatiquement une instance du gestionnaire d'affichage approprié au système de fenêtrage et toutes les autres ressources nécessaires. Cependant, certaines ressources ne sont pas créées automatiquement, puisqu'elles sont optionnelles. C'est le cas par exemple du système de gestion d'erreurs, de gestion d'aide et de gestion des messages (le gestionnaire de messages permet de stocker des chaînes de caractères spécifiques à la langue d'utilisation). De plus, la création du gestionnaire de stockage est une condition *sine qua non* à l'utilisation des fenêtres crées par le Zinc Designer. Pour produire de tels objets, le programmeur doit rajouter à son code les lignes suivantes:

```
/* Ce code doit être situé tout de suite après la */
/* création du gestionnaire de fenêtres           */

// Copie du nom du fichier de données dans une variable
ZIL_INTERNATIONAL::strcpy( fd, "fichier_designer.dat" );

// Création du gestionnaire d'erreurs
UI_WINDOW_OBJECT::errorSystem = new UI_ERROR_SYSTEM;

// Création du gestionnaire d'aide
if ( !ZIL_STORAGE::ValidName( fd ) ) {
      UI_WINDOW_OBJECT::errorSystem->ReportError(
            windowManager,
```

```
                    WOS_NO_STATUS,
                    "Erreur d'ouverture du fichier Designer" );
                    delete UI_WINDOW_OBJECT;
                    return( 1 );
}

// Création du gestionnaire de stockage
UI_WINDOW_OBJECT::defaultStorage =
      new ZIL_STORAGE_READ_ONLY( fd );

// Création du gestionnaire de messages
messageTable = new ZIL_LANGUAGE( "nom_de_table_de_mesgs",
      UI_WINDOW_OBJECT::defaultStorage );

// Création du gestionnaire d'aide
UI_WINDOW_OBJECT::helpSystem = new UI_HELP_SYSTEM( fd );

/* Cette section doit se retrouver à la fin de la */
/* procédure Main, dans la section de nettoyage   */
delete UI_WINDOW_OBJECT::helpSystem;
delete UI_WINDOW_OBJECT::messageTable;
delete UI_WINDOW_OBJECT::defaultStorage;
delete UI_WINDOW_OBJECT::errorSystem;
```

Les variables `helpSystem`, `messageTable`, `defaultStorage` et `errorSystem` sont des variables membre héritées de UI_WINDOW_OBJECT. L'exemple précédent génère le système d'erreurs en permier. Il doit apparaître en premier, puisqu'il servira à avertir l'usager en cas d'erreur lors de la création des gestionnaires suivants. Ensuite, le gestionnaire de stockage est créé, puis le gestionnaire de messages et le gestionnaire d'aide.

Jusqu'à présent, nous avons vu comment créer la procédure principale et les ressources nécessaires au fonctionnement du programme, mais aucun des exemples ne comporte de fenêtres. Nous verrons ici comment utiliser les fenêtres créées à partir du Designer, comment les modifier à l'aide de lignes de code C++ et comment créer des fenêtres à partir de code, sans passer par le Designer. Nous examinerons aussi comment est implantée la routine de gestion des événements `Event( )`, et son utilisation pour interpréter les messages en provenance des objets de la fenêtre.

La première étape dans la création d'une fenêtre est la dérivation d'une classe propre au programme héritant ses propriétés de la classe UIW_WINDOW. Toutes les fenêtres de l'interface usager sont créées à partir d'objets qui ont hérité leurs propriétés de UIW_WINDOW. Il est possible d'instancier une fenêtre sans avoir à créer une nouvelle classe héritière de UIW_WINDOW, mais une telle pratique n'est justifiable que pour des fenêtres temporaires et d'utilisation restreinte. La génération d'une telle fenêtre s'effectue de la manière suivante:

```
UIW_WINDOW *fenetre = UIW_WINDOW::Generic(2, 2, 40, 6, "Titre")
*window +
      new UIW_PROMPT(0, 0, 0, 0, "Bonjour!");
```

L'exemple précédent crée une fenêtre à partir d'une matrice de fenêtre générique, à la position (x,y) = (2,2), de largeur 40 unités et de hauteur 6 unités, avec un titre. La seconde ligne crée un objet UIW_PROMPT, et l'attache à la position (0,0) de la fenêtre. On remarque l'utilisation de l'opérateur '+' pour ajouter un élément à la fenêtre. En effet les opérateurs '+' et '–' sont redéfinis pour la presque majorité des objets de Zinc et servent à ajouter ou à soustraire un élément d'une fenêtre, du gestionnaire de fenêtres ou du gestionnaire d'événements.

La création d'un objet dérivé de la classe UIW_WINDOW permet d'avoir une fenêtre simple d'utilisation, facile à comprendre et permet d'avoir un code clair et modifiable. La classe suivante peut servir de guide de création de fenêtre:

```
// fichier .hpp
class FENETRE_GENERIQUE : public UIW_WINDOW
{
public:
      FENETRE_GENERIQUE( char );   // Constructeur
      -FENETRE_GENERIQUE( void );  // Destructeur
      // Gestionnaire d'événements de la fenêtre
      virtual EVENT_TYPE Event( const UI_EVENT &event );

      // Placer ici les méthodes accessibles de l'extérieur

private:
```

```
        /* Placer ici les membres (données) et les méthodes      */
        /* qui ne sont disponibles qu'à l'intérieur de la classe */
}

// Fichier source .cpp
/* Définition du constructeur */
FENETRE_GENERIQUE::FENETRE_GENERIQUE( char *nom )
        : UIW_WINDOW( nom, UI_WINDOW_OBJECT::defaultStorage )
{
        /* Initialisation des paramètres de la fenêtre */
}
/* Définition du destructeur */
FENETRE_GENERIQUE::-FENETRE_GENERIQUE( void )
{
        /* Désinitialisation des paramètres de la fenêtre */
}
/* Boucle d'événements de la fenêtre */
EVENT_TYPE FENETRE_GENERIQUE::Event( const UI_EVENT &event )
{
        EVENT_TYPE ccode = event.type;

        switch( ccode ) {
                case XYZ:
                        /* Traiter les événements qui sont */
                        /* dirigés vers cette fenêtre      */
                        break;
                default:
                        /* Cette fenêtre ne peut interpréter    */
                        /* l'événement manuellement: utiliser   */
                        /* la procédure par défaut d'UIW_WINDOW */
                        ccode = UIW_WINDOW::Event( event );
}
```

L'exemple donné fonctionne pour des fenêtres créées à partir du Designer. Pour créer une classe à partir de code uniquement, le constructeur de la fenêtre doit être différent et écrit selon le modèle suivant:

```
// fichier .hpp
class FENETRE_GENERIQUE : public UIW_WINDOW
{
public:
        // Constructeur
        FENETRE_GENERIQUE( int, int, int, int, WO_FLAGS );
        -FENETRE_GENERIQUE( void );  // Destructeur
        // Gestionnaire d'événements de la fenêtre
        virtual EVENT_TYPE Event( const UI_EVENT &event );

        // Placer ici les méthodes accessibles de l'extérieur
```

```
private:
      /* Placer ici les membres (données) et les méthodes      */
      /* qui ne sont disponibles qu'à l'intérieur de la classe */
}
// Fichier source .cpp
/* Définition du constructeur */
FENETRE_GENERIQUE::FENETRE_GENERIQUE( int gauche, int haut,
      int largeur, int hauteur, WO_FLAGS woFlags )
      : UIW_WINDOW( gauche, haut, largeur, hauteur, woFlags )
{
      /* Création des objets à l'intérieur de la fenêtre */
      *this
            + // placer ici le premier objet à rajouter
            + ...
      /* Initialisation des paramètres de la fenêtre */
}
```

Maintenant que les fenêtres ont été créées, elles doivent être rattachées au gestionnaire de fenêtres. Tout comme il existe plusieurs types de fenêtres, il existe plusieurs façons de rattacher les fenêtres au gestionnaire de fenêtres:

1. *Fenêtre normale ou fenêtre parent*: ces fenêtres ne sont pas des fenêtres MDI, c'est-à-dire qu'aucune fenêtre enfant n'y est rattachée. L'ajout et l'utilisation d'une telle fenêtre sont relativement simple. Il suffit, une fois la création de la fenêtre terminée, de la rattacher directement au gestionnaire de fenêtres de la manière suivante:

   ```
   FENETRE_GENERIQUE *fenêtre
         = new FENETRE_GENERIQUE( "FENETRE_GENERIQUE" );
   *windowManager + fenêtre;
   UI_APPLICATION::Control( );
   ```

   Notons ici que la fenêtre utilisée était de type créée par de Designer, et que la chaîne de caractères fournie en paramètre est le nom de la fenêtre comme défini dans le Designer. La première ligne crée l'instance, la deuxième l'attache au gestionnaire de fenêtres, et la troisième appelle la boucle principale gestionnaire d'événements du programme, afin de pouvoir utiliser la fenêtre en question.

   Pour fermer une fenêtre, il suffit de lui envoyer le message S_CLOSE à l'aide du gestionnaire d'événements.

2. *Fenêtre enfant (ou fenêtre MDI)*: les fenêtres MDI sont un peu plus complexes. Tout d'abord, le parent et l'enfant doivent avoir dans leur membre woAdvancedFlags le fanion WOAF_MDI_OBJECT activé, préférablement à partir du constructeur de la fenêtre:

```
woAdvancedFlags |= WOAF_MDI_OBJECT;
```

Ensuite,

```
UI_WINDOW_OBJECT *object = ZIL_NULLP( UI_WINDOW_OBJECT );
UI_REGION update;

object = Last( );
if ( object )
      update = object->OSRegion;

FENETRE_GENERIQUE fenetre = new FENETRE_GENERIQUE( "nom" );

Add( fenetre );
Add( fenetre );

if ( object )
      Event( UI_EVENT( S_DISPLAY_INACTIVE, 0, update ) );

Event( UI_EVENT( S_DISPLAY_ACTIVE, 0, fenetre->OSRegion ) );
```

La première ligne est une façon élégante d'effectuer une assignation de la valeur NULL transtypée (i.e. "*castée*") en objet UI_WINDOW_OBJECT. Les deux autres lignes sauvegardent la position de la dernière fenêtre placée dans le gestionnaire de fenêtres. Ensuite, l'opération d'ajout est appelée deux fois à cause d'un problème de la bibliothèque Zinc (si une seule opération est effectuée, la fenêtre apparaît, mais la fonction `Current( )`, permettant d'aller chercher un pointeur sur la fenêtre courante, ne fonctionne pas). Les lignes suivantes rafraîchissent la zone d'affichage que recouvre la fenêtre.

Pour fermer une fenêtre MDI, il suffit de lui enyoyer le message S_CLOSE + S_MDICHILD_EVENT.

Lorsque les fenêtres sont enlevées du gestionnaire de fenêtres, il n'est pas nécessaire de les détruire à l'aide de la fonction `delete( )`, même si elles ont été créées à partir de l'opérateur `new( )`: le destructeur de la fenêtre prend en charge la tâche de libérer les ressources mémoire, à mois que le programmeur spécifie à la fenêtre de ne pas se détruire automatiquement.

Une fois les fenêtres créées, attachées à la fenêtre et leur comportement dicté avec la méthode `Event( )`, le programmeur n'a qu'à savoir comment extraire et placer l'information dans les fenêtres. La majorité des opérations qu'il aura à effectuer auront lieu avec la fonction `Get( )`, qui permet d'aller chercher un pointeur sur un objet de la fenêtre, et avec la fonction `Information( )`, qui permet l'extraction de l'information comme telle. Un exemple sera la meilleure façon d'expliquer et d'illustrer leur utilisation:

```
...
/* Aller chercher un pointeur sur l'objet désiré à partir du   */
/* nom qui lui a été attribué par le programmeur dans Designer */
UIW_STRING *chaine = (UIW_STRING *)Get( "NOM_D_OBJET" );

char *texteDeLaChaine;

chaine->Information( I_GET_TEXT, &texteDeLaChaine );
...
chaine->Information( I_SET_TEXT, "Chaîne à afficher" );
...
```

La première étape consiste à aller chercher le pointeur sur l'objet voulu, avec le nom qui lui a été attribué par le programmeur dans Zinc Designer, et de la fonction `Get( )`. La seconde étape consiste à utiliser la fonction `Information( )` pour extraire ou placer la valeur voulue dans l'objet. Il est à noter que chaque objet hérité de UI_WINDOW_OBJECT a sa méthode "Information" redéfinie et a ses propres messages d'informa-

tion (du genre I_GET_TEXT). Par exemple, le message I_GET_TEXT n'est pas valable pour l'objet UIW_INTEGER, qui utilise I_GET_VALUE pour la lecture de son contenu.

7.3.6 Compilation des programmes sous les diverses plate-formes

Chaque plate-forme a ses propres fichiers de bibliothèques, et chacun son mode de compilation particulier. Quoique la compilation soit généralement simple, l'utilisateur doit cependant faire attention aux points suivants:

1. *Sous Microsoft Windows avec Borland C++*: les bibliothèques nécessaires sont: `WIN_ZIL.LIB`, `WSERVICE.LIB` et `WDIRECT.LIB`. La première bibliothèque est la bibliothèque principale pour Windows 16 bits, et les deux secondes sont des bibliothèques qui sont créées à la compilation du Zinc Designer.

   Le programme *doit* utiliser le modèle de compilation "Large". La bibliothèque "Runtime" doit aussi être sélectionnée pour l'édition de liens, et le tout doit être uni de façon statique. En ce qui concerne les options de compilation, les étapes suivantes doivent être effectuées: dans le menu "Options/Project/C++ Options/Exception handling RTTI" de l'interface Borland, les options "Enable exceptions" et "Enable runtime type information" doivent être déselectionnés, car ils entreront en conflit avec le gestionnaire d'erreurs de Zinc. La taille du tas et de la pile du programme ("*heap*" et "*stack*") doivent être de 19200 octets; cette opération est effectuée dans le fichier `.def` utilisé par l'application.

2. *Sous SunOS et Linux*: les bibliothèques nécessaires s'appellent `lib_mtf_zil.a`, `lib_direct.a` et `lib_service.a`. De plus, les bibliothèques MOTIF `libXm.a`, `libXt.a` et `libX11.a` sont nécessaires. L'usager n'a qu'à se faire

un fichier `makefile` compilant tous les fichiers source et effectuant l'édition des liens de tous les fichiers objets et de toutes les bibliothèques.

7.3.7 Description et hiérarchie des objets de l'interface usager

Tout comme les objets de la bibliothèque possèdent une hiérarchie grâce à l'héritage, les objets utilisés par l'interface usager suivent aussi une hiérarchie qui leur est propre. On distingue principalement quatre familles d'objets.

1. *Les objets fenêtres*: ces objets ont une hiérarchie assez horizontale, c'est-à-dire que l'héritage n'est utilisé que de façon limitée. Les fenêtres de l'interface usager sont toutes dérivées directement de la classe UIW_WINDOW. Voici la hiérarchie des fenêtres de l'interface:

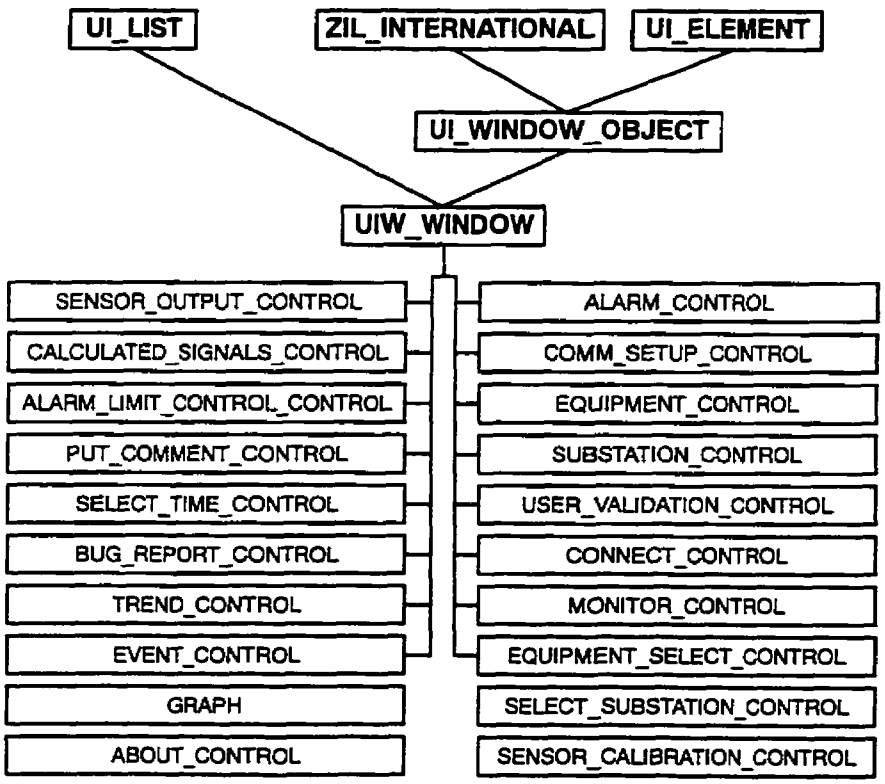


Figure 7.31: Hiérarchie des classes utilisées pour les fenêtres de l'interface

Les noms de classes en gras dans la figure 7.31 sont des objets fournis par Zinc. Les autres objets sont des objets propres à l'interface usager. Toutes ces classes ont différentes propriétés et utilités:

- **UI_LIST**: utilisé pour la gestion des listes. Implante, entre autres, l'opérateur `Add( )` (ou la redéfinition de l'opérateur '+') et `Subtract( )` (ou la redéfinition de l'opérateur '−') permettant d'ajouter un objet à une liste aisément. Cette classe est aussi responsable des pointeurs `first`, `last` et `current` vus aux sections 7.3.7.1 et 7.3.3.2;
- **UI_ELEMENT**: responsable de la fonction Information, pour interroger un objet. Aussi responsable des pointeurs `next` et `previous` des sections 7.3.7.2 et 7.3.3.2;
- **ZIL_INTERNATIONAL**: s'occupe de l'internationalisation des objets. Tient compte de la localité et de la langue d'utilisation. Possède une foule de méthodes permettant le traitement de chaînes, du genre `strcpy` et `atoi`, de caractères internationaux;
- **UI_WINDOW_OBJECT**: l'objet de base pour tous les objets que l'on peut afficher. Toutes les classes ayant comme préfixe les lettres UIW, dont fait partie UIW_WINDOW, dérivent leur information de cette classe. Cette classe est responsable du gestionnaire d'événements et de fenêtrage de chaque objet. Elle est aussi responsable de l'apparence de l'objet à l'écran;
- **UIW_WINDOW**: cette classe est la première, dans cette hiérarchie, qui n'est pas une classe purement virtuelle (c'est à dire qu'elle peut être instanciée, ou exister directement). Elle définit les fonctions de défilement à travers les barres

de défilement horizontales et verticales, et s'occupe des fonctions de modification de taille et de déplacement de la fenêtre;

- SENSOR_OUTPUT_CONTROL: cette fenêtre prend charge l'affichage des valeurs des capteurs. Elle permet à l'usager d'effectuer une relecture périodique des valeurs affichées;
- CALCULATED_SIGNALS_CONTROL: permet à l'usager d'aller chercher la liste des signaux calculés pour un événement particulier;
- ALARM_LIMIT_MODIFICATION_CONTROL: lorsqu'un usager demande l'édition d'une condition d'alarme, cette fenêtre lui permet d'éditer les différentes valeurs et de les envoyer au contrôleur central;
- PUT_COMMENT_CONTROL: lorsqu'un usager veut accuser réception d'une alarme, le système lui demande d'associer un commentaire à l'alarme reconnue, à l'aide de cette fenêtre. La fenêtre comporte un objet UIW_TEXT pour l'entrée d'un commentaire à l'aide d'un traitement de texte sommaire. L'objet permet aussi de valider le commentaire: certains caractères, tels les caractères ASCII supérieurs à 0xFF et le caractère '|', ne sont pas acceptés dans les commentaires;
- SELECT_TIME_CONTROL: l'interface usager fait appel plusieurs fois à cette fenêtre pour obtenir de l'usager une valeur de temps;
- BUG_REPORT_CONTROL: si l'interface rencontre un problème, elle affiche une fenêtre donnant une foule d'informations système et des informations sur l'erreur qui s'est produite. Ces informations peuvent ensuite être transmis aux développeurs, pour qu'ils puissent analyser les conditions d'erreur et y remédier;

- TREND_CONTROL: l'interface permet d'afficher des graphiques de tendance. La présente fenêtre dispense des services de création de tels graphiques, en spécifiant entre autres la plage de temps sur laquelle devra s'écouler chacun des signaux à afficher dans le graphique, leur filtrage et leur apparence;
- EVENT_CONTROL: de même que pour les graphiques de tendances, les graphiques d'événements ont aussi besoin d'une interface donnant la chance à l'usager de spécifier ses besoins;
- GRAPH: cet objet fenêtre est un cas particulier de fenêtre. Ils affichent des informations sous forme de graphiques sur lesquels on peut effectuer une foule d'opérations. Le point 3. définit cet objet avec plus de précision;
- ABOUT_CONTROL: il existe dans l'interface usager une fenêtre affichant de l'information générale sur le programme comme sa version, le mode de connexion présent, et les paramètres de connexion. La fenêtre ABOUT_CONTROL sert à l'affichage de cette information;
- ALARM_CONTROL: prend charge des différentes fonctions reliées aux alarmes. Gère l'affichage des alarmes, leur reconnaissance, la désactivation et l'activation, tout comme la modification des seuils d'alarmes;
- COMM_SETUP_CONTROL: le mode et les paramètres de communications doivent être spécifiés afin de pouvoir se brancher. Cette fenêtre comporte deux volets: une section pour la spécification et l'édition des paramètres de communications à une sous-station, et une section pour la configuration de modems;
- EQUIPMENT_CONTROL: plusieurs informations peuvent être affichées sur l'équipement sélectionné dans l'interface. La présente fenêtre affiche des informations sur le statut, l'usure, les alarmes et l'historique de l'équipement;

- SUBSTATION_CONTROL: cette fenêtre ressemble beaucoup à la fenêtre précédente, à la différence qu'elle affiche l'information sur toute la sous-station au lieu de se spécialiser sur un équipement particulier;
- USER_VALIDATION_CONTROL: cette fenêtre permet la validation de l'usager en lui demandant son nom d'usager et son mot de passe;
- CONNECT_CONTROL: cette fenêtre est la première fenêtre qui apparaît lors du démarrage de l'interface. Elle demande à l'usager le mode de communication qu'il voudra utiliser;
- MONITOR_CONTROL: la fenêtre principale de l'interface du système de télésignalisation. Elle possède un menu déroulant, une barre d'outils et une barre d'état. Toutes les autres fenêtres sont des fenêtres enfant MDI de cette fenêtre;
- EQUIPMENT_SELECT_CONTROL: cette fenêtre est très simple. Elle affiche les équipements disponibles sur cette sous-station, et demande à l'usager d'en choisir un comme équipement courant;
- SELECT_SUBSTATION_CONTROL: l'usager peut définir plusieurs configurations de sous-stations. Cette fenêtre lui demande laquelle des configurations il veut utiliser, et génère l'objet de communication approprié. Le constructeur de l'objet de communication effectue alors les premières étapes de la communication;
- SENSOR_CALIBRATION_CONTROL: cette fenêtre permet de modifier la calibration des capteurs du système de télésignalisation.

2. *Les objets de communications*: les objets de communication font une utilisation plus approfondie des notions d'héritage que les objets de la section précédente. Comme

Zinc ne supporte pas les communications, aucun objet de communication n'hérite des propriétés des objets de Zinc:

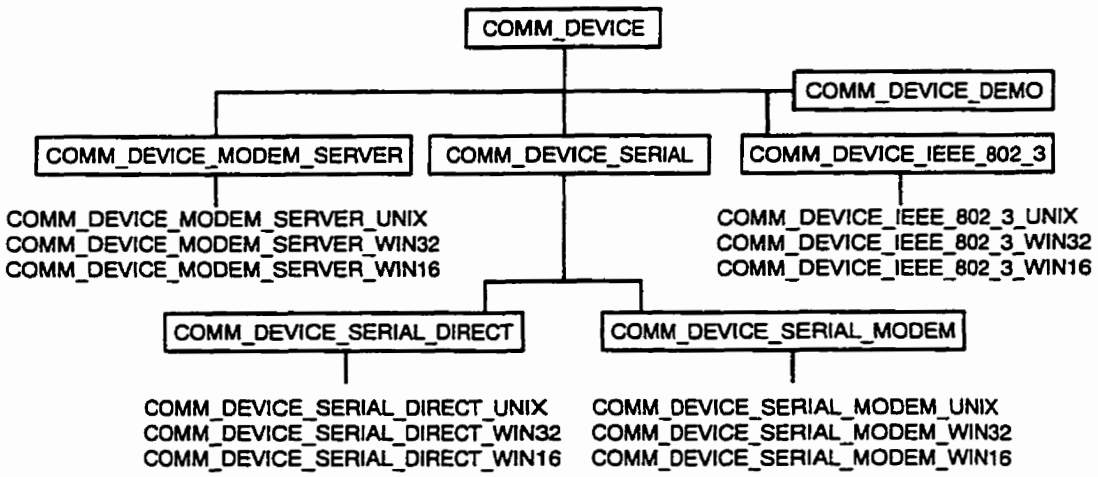


Figure 7.32: Hiérarchie des classes utilisées pour la communication

Les classes purement virtuelles sont les classes encadrées. Les autres peuvent être instanciées, selon le système d'exploitation utilisé. Cette architecture permet de rajouter facilement de nouveaux modes de communications, tout comme l'ajout de nouvelles plate-formes. On doit noter que l'implantation de chaque classe doit être effectuée selon le type de communication nécessaire. Voici la description des différentes classes et de leur mode de fonctionnement:

- COMM_DEVICE: la classe parent de toutes les autres. Définit virtuellement les méthodes d'ouverture et de fermeture du périphérique de communications, et la méthode d'envoi de requêtes. Notons que les méthodes virtuelles ne sont par définition que des entêtes de fonctions qui devront absolument être définies chez un héritier pour que l'enfant puisse réellement être généré. La classe définit aussi (réellement cette fois, et non en méthode purement virtuelle), les fonctions de gestion des erreurs;

- COMM_DEVICE_DEMO: cette classe est assez particulière car elle définit toutes les fonctions nécéssaires à l'instanciation de la classe, mais que ces méthodes sont en fait vides, et ne retournent jamais d'erreurs ni d'informations. Elle sert pour le mode de démonstration de l'interface;
- COMM_DEVICE_SERIAL: cette classe définit les méthodes qui sont à la fois communes aux modems et aux communications directes, et qui ne requièrent pas d'appels aux fonctions de bas niveaux spécifiques pour les communications. Elle est aussi purement virtuelle, car elle ne définit que la méthode d'envoi de requêtes, et laisse le soin à ses héritiers de définir les méthodes d'ouverture et de fermeture de périphérique de communication;
- COMM_DEVICE_SERIAL_DIRECT: définit les paramètres d'ouverture du port de communications série, comme la vitesse de communications, la parité et le numéro de port série. Purement virtuelle, cette classe ne contient pas les méthodes d'exploitation du port série;
- COMM_DEVICE_SERIAL_MODEM: effectue les mêmes fonctions que la classe précédente, en plus de spécifier les diverses chaînes de configuration de modem et le numéro de téléphone;
- COMM_DEVICE_SERIAL_DIRECT_[UNIX/WIN16/WIN32]: cette classe fournit les primitives d'envoi de chaînes de caractères et de réception de chaînes de caractères utilisées par la méthode d'envoi de requêtes. Elle définit aussi les méthodes de configuration du port série et celles de la gestion des dépassements de temps imparti (i.e. "*timeouts*");
- COMM_DEVICE_SERIAL_MODEM_[UNIX/WIN16/WIN32]: définit les mêmes méthodes que les classes précédentes, en spécifiant aussi les méthodes

servant à l'initialisation du modem à l'aide des chaînes de configuration et l'établissement de la connexion à l'aide du modem;

- COMM_DEVICE_MODEM_SERVER: cette classe aurait pu bénéficier de l'héritage des objets COMM_DEVICE_SERIAL_MODEM et COMM_DEVICE_IEEE_802_3, mais l'architecture aurait été plus difficile à maintenir. L'auteur a donc décidé de créer une nouvelle classe sous COMM_DEVICE pour ce mode de communication. Elle est un croisement entre ces deux classes, avec la particularité de l'établissement de la connexion à travers le serveur de modem;
- COMM_DEVICE_MODEM_SERVER_[UNIX/WIN16/WIN32]: implante la procédure d'établissement de connexion et d'envoi et réception de chaînes de caractères sur un système utilisant un serveur de modems.

3. *Les objets pour les graphiques*: trois classes interviennent dans la conception de graphiques. Deux de ces classes, les classes GRAPH et LINE, sont spécifiques aux graphiques et ne peuvent être utilisées ailleurs. La troisième est la classe BIT-MAP_VECTOR, ou vecteur de bits, disponibles à d'autres applications dans le logiciel:

   - LINE: cette classe est relativement simple et contient tout ce qui peut caractériser une ligne dans un graphique de tendances ou d'événements. Elle ne contient, à part son constructeur, aucune méthode. En fait, la majorité de son contenu est constitué d'objets membres (données) qui définissent les coordonnées de chaque point la constituant, de sa palette de couleur, de son nom, du type de marqueur servant à identifier les points et sa valeur minimale et maximale. Elle contient aussi un vecteur de bits ayant autant de bits que de points de données.

Ce vecteur, dont chaque bit est mappé sur un point de données, indique si le point en question doit être affiché ou non. Ce vecteur permet d'avoir des courbes non continues, indiquant ainsi un manque de données à l'intérieur d'une courbe;

- GRAPH: cet objet est beaucoup plus complexe que le précédent. Il s'occupe de l'affichage de chacune des courbes qu'il contient, d'afficher automatiquement les échelles sous forme de points ou de dates, de déterminer automatiquement les échelles et divisions, d'effectuer des zooms avant et arrière, de gérer les titres et d'effectuer une foule d'autres opérations sur les données.

Certaines choses sont à retenir de l'objet GRAPH, dont la méthode utilisée pour la détermination automatique des divisions d'échelles. Elle s'effectue à partir des valeurs minimales et maximales de toutes les données que l'on retrouve sur le graphique. L'algorithme est le suivant:

Détermination automatique des divisions d'échelles de graphiques

valeurs à précision double:
plage, l10, norme, tics
valeurs à précision double passées en paramètres:
min, max

plage ← |min – max|

; si la plage est de 0, dx/dt = 0. On obtiendra donc une erreur
; à cause de $\log_{10}(0)$. 5 a été choisi aléatoirement
si (plage = 0) alors plage = 5

l10 ← $\log_{10}$(plage)

norme = $10^{l10-\left\{\begin{matrix} l10\geq 0 \rightarrow \lfloor l10 \rfloor \\ l10<0 \rightarrow \lfloor l10 \rfloor - 1 \end{matrix}\right\}}$

si norme ≤ 2 alors

$$tics = 0{,}2 \times 10^{\left\lfloor n10 - \left\{ \begin{matrix} n10 \geq 0 \rightarrow \lfloor n10 \rfloor \\ n10 < 0 \rightarrow \lfloor n10 \rfloor - 1 \end{matrix} \right\} \right\rfloor}$$

sinon si norme ≤ 5 alors

$$tics = 0{,}5 \times 10^{\left\lfloor n10 - \left\{ \begin{matrix} n10 \geq 0 \rightarrow \lfloor n10 \rfloor \\ n10 < 0 \rightarrow \lfloor n10 \rfloor - 1 \end{matrix} \right\} \right\rfloor}$$

sinon

$$tics = 1{,}0 \times 10^{\left\lfloor n10 - \left\{ \begin{matrix} n10 \geq 0 \rightarrow \lfloor n10 \rfloor \\ n10 < 0 \rightarrow \lfloor n10 \rfloor - 1 \end{matrix} \right\} \right\rfloor}$$

retourner tics comme valeur de division d'échelle

Il est important de comprendre comment les courbes sont affichées dans la fenêtre. L'algorithme qui suit détaille la méthode employée pour cet affichage. Cet algorithme est implanté dans la méthode DrawItem, qui elle est appelée quand le graphique doit être dessiné ou rafraîchi.

Affichage des lignes de graphique

```
pour toutes les lignes du graphique
        pour tous les points de la ligne sauf le dernier
                si on doit afficher le présent point alors
                        calculer les coordonnées du présent point
                        si on doit afficher le point suivant alors
                                calculer les coordonnées du point suivant
                                tirer la ligne entre les deux points
                        fin si
                        afficher le marqueur de point selon son type
                fin si
        fin pour
        si on doit afficher le dernier point alors
                calculer les coordonnées du présent point
                afficher le marqueur de point selon son type
        fin si
fin pour
```

Le zoom avant et le zoom arrière sont implantés à l'aide des valeurs minimales et maximales. Lorsqu'il n'y a pas de zoom, le graphique utilise la valeur maximale et minimale de toutes les courbes calculées à la création du graphique pour

définir les échelles. Lorsqu'il y a zoom, les valeurs minimales et maximales sont substituées par les valeurs minimales et maximales de la boîte de zoom.

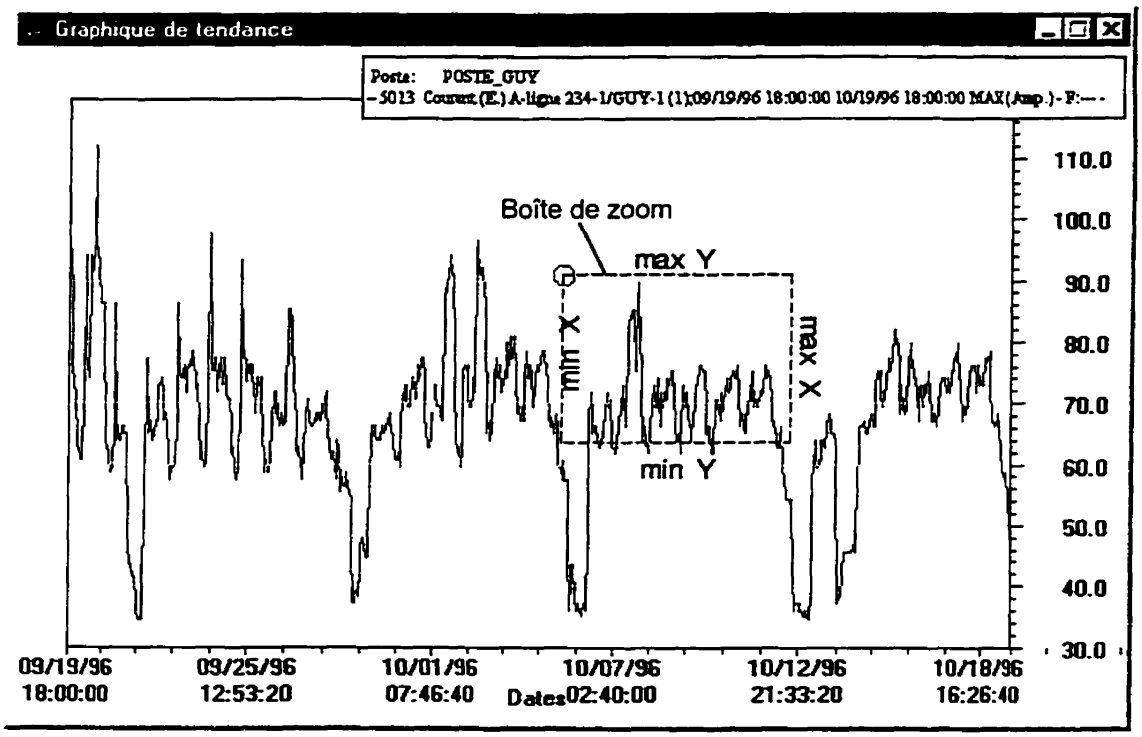


Figure 7.33: Utilisation de la boîte de zoom dans les graphiques

Lors du zoom, les couples (min X, min Y) et (max X, max Y) deviennent les valeurs minimales pour l'objet. En regardant l'algorithme AUCUN LIEN , on remarque que tous les points sont passés un à un pour être affichés, même si ces points sont hors de la vue de l'usager (i.e. à l'extérieur de la boîte de zoom); de ces manipulations mathématiques résultent en une certaine perte de temps. Dans une version ultérieure, cet algorithme d'affichage pourrait être amélioré.

4. *Les autres objets*: les autres objets se classent en deux familles distinctes. Les gestionnaires qui font partie de la première famille sont majoritairement des instances statiques (il n'y a pas deux instances d'une instance statique: elle est unique et

aucune autre instance du même type est présente dans le programme au même moment) servant à gérer certaines conditions et services. On retrouve entre autres:

- PRINT_MANAGER: s'occupe de l'impression des graphiques et du texte de l'interface usager. Envoie les informations au gestionnaire d'impression de l'interface de fenêtrage;
- ANSWER_PARSER: offre des services de formatage de chaînes de caractères. En effet chaque commande envoyée de l'interface usager vers le contrôleur central génère une réponse. Le contenu de cette réponse varie évidemment selon la requête envoyée en premier lieu. Toutefois l'information n'arrive pas nécessairement dans un format utilisable immédiatement par l'interface usager. Les services offerts par ce gestionnaire permettent de formater une réponse selon le type de message envoyé;
- ERROR_MANAGER: effectue la gestion des erreurs pouvant survenir dans l'interface à cause de problèmes de programmation ou d'erreur de l'usager et l'affichage de ces erreurs;
- MONITOR_STORAGE: gère le système de stockage d'information de l'interface usager. Ce système de stockage contient entre autres les différentes configurations de connexion, la configuration des modems et les paramètres d'opération de l'interface;
- INIT_MANAGER: l'interface usager met à la disposition de l'usager un fichier texte contenant des informations de configuration de logiciel, à la manière des fichiers `.INI` de Microsoft Windows. Cet objet s'occupe de lire et d'interpréter ce fichier au chargement de l'interface;

- LANGUAGE_MANAGER: l'interface peut supporter diverses langues d'opération. Pour l'instant, seul le support de l'anglais et du français sont prévus, mais il est possible d'en supporter d'autres. Ce module s'occupe d'aiguiller les requêtes d'affichage de fenêtres aux fichiers correspondant à la langue désirée;
- PROGRAM_STATUS_INFORMATION: ce gestionnaire contient une foule d'informations nécessaires au fonctionnement de l'interface et utilisées par presque tous les autres modules. Il contient entre autres le nom et le numéro de l'équipement sélectionné.

En plus des gestionnaires, il existe plusieurs autres classes représentant des objets réels ou des concepts. Ces classes sont:

- STRING et HSTRING: l'interface usager manipule énormément de chaînes de caractères. Quand elle reçoit une réponse à une requête, c'est sous forme de chaîne de caractères qu'elle les reçoit. Cette classe a été conçue afin de simplifier l'utilisation de telles chaînes et de tenter de limiter les accès aux fonctions d'allocation de mémoire directement par le programmeur, qui causent souvent beaucoup de problèmes. La classe permet aussi d'accéder des fonctions de copie et de recherche beaucoup plus facilement qu'à l'habitude. Voici un exemple qui illustre bien la méthode conventionnelle et la méthode utilisant les objets chaînes de caractères. Dans cet exemple, on crée une chaîne, puis on copie de l'information à l'intérieur et on lui fait la concaténation d'une autre chaîne:

```
// Utilisation de la méthode traditionnelle
char *chaine, *ptr;
char chaineFinale[] = ", comment allez-vous?\0";

chaine = new char[ strlen("Bonjour") + 1 ];
strcpy( chaine, "Bonjour\0" );
ptr = new char[ strlen( chaine ) + strlen( chaineFinale ) + 1 ];
strcpy( ptr, chaine );
strcat( ptr, chaineFinale );
```

```
delete[] chaine;
chaine = ptr;
delete[] chaine;

//Utilisation de STRING
STRING chaine = "Bonjour";
char chaineFinale[] = ", comment allez-vous?\0";

chaine += chaineFinale;
```

Il est fréquent qu'un programmeur oublie le '+1' utilisé lors de l'allocation de mémoire, pour allouer de l'espace pour le caractère de terminaison '\0'. Tout est effectué automatiquement avec l'objet STRING. Notons que dans la section d'utilisation traditionnelle, aucune gestion d'erreur n'est effectuée. Si le programme manque de mémoire, l'usager ne s'en aperçoit pas et le pire arrive... Comme l'objet STRING possède sa propre gestion d'erreurs, il fait affaire avec le gestionnaire ERROR_MANAGER, ce qui permet d'avertir l'usager en cas de problème d'espace mémoire;

- BITMAP_VECTOR: cette classe, comme nous l'avons mentionné à la section graphiques, permet la création de vecteurs de bits, sur lesquels on peut effectuer plusieurs opérations d'interrogations et d'assignation;
- EXTENDED_TIMER: cette classe est celle d'un temporisateur, envoyant un message E_TIMER à la boucle d'événements `Event( )` de la fenêtre auquelle il est rattaché. Le message peut être envoyé indéfiniment ou un nombre de fois prédéterminés, et à des intervalles de temps programmables;
- MODEM_CONFIG_STRING: cet objet devrait en fait avoir un nom pluriel, car il est un réceptacle pour toutes les chaînes de configuration de modem, soit la chaîne d'initialisation, la chaîne d'attention, la chaîne de commande (du genre "+++", etc.). Des primitives ont aussi été conçues pour l'écriture et la lec-

ture automatique de son contenu dans le fichier de stockage de l'interface usager;

- SUBSTATION: cet objet aussi devrait avoir un autre nom, soit celui de *configuration* de sous station, puisqu'il contient toute l'information de configuration d'une connexion vers un contrôleur central.

7.3.8 Difficultés d'implantation selon la plate-forme

Certaines difficultés ont été rencontrées lors de la création de l'interface, mais la principale difficulté a été et demeure la gestion de la mémoire dans les objets STRING. L'interface requiert l'accès à des chaînes de plus de 64 Ko: UNIX ou Windows 32 bits ne posent aucun problème. Cependant pour utiliser des blocs de mémoire contigus de plus de 64Ko dans l'environnement Windows 16 bits, le programmeur doit utiliser des chaînes de type "huge" qui sont très lourdes à manipuler. Le programme doit à *chaque* accès normaliser l'adresse du segment de mémoire où se trouve l'information. Des mesures ont été effectuées quant à la rapidité d'exécution de manipulations de pointeurs permettant l'accès à moins de 32 Ko de mémoire (type "far") et de pointeurs permettant l'accès à des zones mémoire de plus de 64 Ko: les résultats sont étonnants. Les pointeurs "far" sont au moins dix fois plus rapides que les pointeurs "huge".

Pour diminuer l'impact de ces transformations d'adresses de pointeurs, deux classes de chaînes ont été créées: une supportant les pointeurs "huge" (HSTRING) et une autre supportant les pointeurs "far" (STRING). Les chaînes HSTRING ne sont utilisées que lorsqu'il y a vraiment possibilité de débordement dans plus grand que 32 Ko de mémoire. Toutefois, cette solution n'est pas aussi élégante qu'elle devrait l'être. Il serait possible de créer une nouvelle classe de chaîne ayant une gestion de mémoire basée sur une structure de système

de fichiers, avec un index où l'on trouve un pointeur sur le bloc de 32 Ko et un autre pointant à la position de l'information à l'intérieur du bloc de 32 Ko.

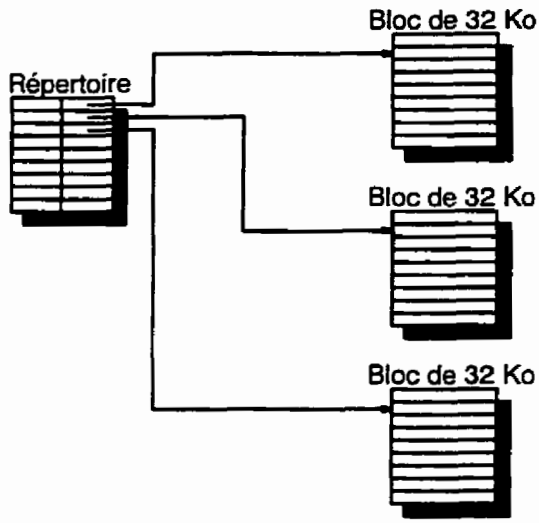


Figure 7.34: Solution au problème de mémoire sous Windows 16 bits

En plus des problèmes de gestion de mémoire, la bibliothèque Zinc a posé quelques problèmes: depuis le début du projet il y a deux ans, la bibliothèque est passée de la version 4.0 à la version 4.2 en cinq étapes. Depuis ce temps la majorité des problèmes causant des "bugs" ont été résolus.

7.4 Développements futurs

En date du 19 octobre 1996, le code de l'interface usager comporte 25 127 lignes (codes et commentaires) ou 816 897 octets répartis en 125 fichiers. Le code évolue de jour en jour, et de nouvelles modifications y sont apportées constamment. Plusieurs améliorations sont encore possibles quant à la performance du système. Le système de gestion des fenêtres et l'algorithme d'affichage des lignes sur les graphiques peuvent être optimisés en fonction de la vitesse.

En plus ce ces modifications souhaitables, un projet est en cours pour l'intégration du contrôleur central et de l'interface usager sur une même plate-forme. Ce projet entraînera les modifications et ajustements suivants:

- fonctionnement de l'interface usager dans l'environnement X de Linux;
- ajout d'un mode de communication par canal de transmission FIFO ("*named pipe*");
- ajout de nouvelles fonctions pour permettre la configuration de la base de données du contrôleur central à partir de l'interface usager.

CONCLUSION

Le but principal de ce projet était d'implanter un système de télésignalisation d'indicateurs de défaut sur un réseau de distribution souterrain 25 kV afin de réduire le temps de service d'un défaut. La réduction du temps de service permet à Hydro-Québec d'offrir un service de distribution de meilleure qualité, tout en lui permettant de réduire les pertes monétaires encourues à cause de l'interruption de la vente d'énergie.

Afin de mettre sur pied ce système, nous avons premièrement examiné les composantes matérielles et logicielles du système de surveillance de disjoncteurs haute tension. Nous avons pu alors déterminer quelles étaient les modifications à lui apporter afin qu'il puisse être adapté à la télésignalisation de défauts. Une étude exhaustive des capteurs a ensuite été mise sur pied, afin d'identifier l'indicateur de défaut répondant le plus aux besoins de la télésignalisation.

Suite à l'étude des indicateurs de défaut, il a été déterminé que le système conserverait, en plus de l'indication de défaut, la valeur de la charge instantanée des lignes sous surveillance; le système pourrait alors être aussi utilisé pour la gestion de la charge.

Une fois l'architecture de base établie à partir des équipements existants, les modifications nécessaires ont été apportées au logiciel et au matériel, pour créer le prototype de laboratoire. Une série de tests de validation ont alors été effectués sur le prototype, avant l'implantation en réseau.

Présentement, les systèmes d'acquisition et la base de données sont en place, et 5 des 41 lignes ont leurs indicateurs de défaut d'installés. La formation des opérateurs a présentement lieu, et l'ensemble du système devrait être entièrement opérationnel d'ici l'été 1997;

l'installation des indicateurs de défaut requiert le retrait d'exploitation d'une ligne, ce qui ne peut se faire l'hiver à cause de la charge globale du réseau.

Aucun défaut ne s'est manifesté depuis l'installation du système au poste Guy, soit le mois de août 1996; l'utilité de la détection de défaut n'a donc pu être évaluée, et les 0,5 M$ investis n'ont pu être justifiés dans la pratique pour le moment. La validation se confirmera lorsque des défauts se manifesteront sur les lignes surveillées. Cependant, les valeurs de courant ont servi maintes fois à évaluer la charge des lignes surveillées.

L'implantation du premier prototype en réseau a fait naître plusieurs idées: un système de télésignalisation d'indicateurs de défauts aériens, utilisant la signalisation par ondes radio et la technologie de puces neurones, est en train de voir le jour. La mise au point de plusieurs aides au système de télésignalisation, tels des outils statistiques sur le nombre de défauts et sur les charges moyennes des différentes lignes et un système expert identifiant automatiquement à partir d'une définition de topologie de défaut l'endroit du défaut sont maintenant considérés.

Le système est appelé à prendre de plus en plus d'importance dans les opérations quotidiennes de gestion du réseau. À l'ère des réseaux intelligents, la qualité et l'automatisation ont une place de choix.

Les tendances mondiales à rechercher la qualité vont aussi faire du système de télésignalisation un outil clé. De plus en plus, Hydro-Québec cherchera à améliorer la qualité de son service de distribution et de transport et la qualité de son produit: des développements sont présentement en cours pour remplacer l'indice de continuité par un indice tenant compte de la charge de chacun des clients et le secteur de la qualité de l'onde prend de plus en plus d'essor. Le système de télésignalisation d'indicateurs de défaut s'inscrit donc dans une

démarche qualité entreprise par Hydro-Québec, et jusqu'à présent, les résultats sont très prometteurs.

RÉFÉRENCES

BENNET, R., SCHWABE, R., et al. (novembre 1993). Demonstration and Field Evaluation of a Condition-Monitoring System for SF6 High-Voltage Circuit Breakers. Document présenté à l'EPRI Substation Equipment Diagnostics Conference, EPRI, Nouvelle Orléans, États-Unis.

COULORIS, G., DOLLIMORE, J. (1994). Distributed Systems Concepts and Design, Deuxième édition, Addison-Wesley, Reading, Massachusetts, États-Unis, 70–71.

DAIGNEAULT, G. (1994). Rapport d'étape – Analyse et sélection de capteurs pour la télésignalisation des indicateurs de défaut du réseau de distribution souterrain. Rapport IREQ-95-080, Institut de recherche d'Hydro-Québec, Varrennes, Canada.

Groupe RCRSD – Révision de la Conception du Réseau Souterrain de Distribution (1991). Rapport RCRSD–91–019, Hydro–Québec, Montréal, Canada.

Groupe RCRSD – Révision de la Conception du Réseau Souterrain de Distribution (1991). Rapport RCRSD–91–022, Hydro–Québec, Montréal, Canada.

LANDRY, M., MERCIER, A., DAIGNEAULT, G. et al. (novembre 1994). Experience and Field Evaluation of a Condition-Monitoring System for High-Voltage Circuit Breakers – An Update. Document présenté à l'EPRI Substation Equipment Diagnostics Conference, EPRI, Nouvelle Orléans, États-Unis.

MARTIN, J. (1967). Design of Real-Time Computer Systems, Prentice-Hall, Englewood Cliffs, New-Jersey, États-Unis.

PATER, R., VALIQUETTE, D. (1994). LCCOM: Programme de communication avec le contrôleur local. Rapport IREQ 94–112, Institut de recherche d'Hydro-Québec, Varrennes, Canada.

ANNEXE I: ANALYSE DU SYSTÈME TEMPS RÉEL

Cette section présente une analyse du contrôleur local utilisé pour le système de surveillance de disjoncteurs haute tension. Elle permet d'établir le taux d'utilisation du système, et sa charge. Quoiqu'il y ait des disparités entre le fonctionnement du système de télésignalisation des indicateurs de défaut et le système de surveillance de disjoncteurs haute tension, elle demeure pertinente, puisqu'il faudra éventuellement effectuer cette étude sur le système de télésignalisation pour en évaluer la charge. L'analyse porte d'abord sur les signaux à utiliser pour l'étude temps réel, puis s'attarde sur les signaux recueillis pour fin d'analyse. Ensuite les concepts de base de la modélisation d'un système sont présentés. Finalement, les données recueillies sont appliquées au modèle afin d'en tirer une conclusion.

I.1 Étude des signaux à utiliser pour la modélisation

Comme nous l'avons vu à la section , on retrouve quatre types de données dans ce système. On doit donc discriminer quelles sont les données utilisées pour le temps réel à contraintes douces. De ces quatre types de signaux, seuls les événements sont placés dans une file d'attente en cas de besoin pour y êtres traités plus tard par le système.

Tableau I.1: Les données du système et leur priorité

Données	Utilité	Contraintes de temps imposées sur les données
Rapides	Créer des événements.	Prioritaire: ces données doivent être conservées dans une file d'attente pour créer les événements. Leur taux d'arrivée est constant à 1 kHz. Ils font partie du système temps réel à contraintes rigides. On ne peut les placer dans une file d'attente.
	Calculer des moyennes, minimums et maximums pour les valeurs de dernière heure et dernière minute.	Non prioritaire: on permet au système de négliger ces données lorsqu'il traite un événement.
Lents	Créer des événements.	Non prioritaire: ces signaux changent très peu aux heures.
	Calculer des moyennes, minimums et maximums pour les valeurs de dernière heure et dernière minute.	Non prioritaire: on permet au système de négliger ces données lorsqu'il traite un événement.
Alarmes	Avertir un usager d'un problème de fonctionnement.	Non prioritaire: comme les alarmes ne servent pas à contrôler aucun processus et ne servent qu'à avertir un usager, les contraintes de temps ici sont négligées: si l'alarme arrive à l'usager quelques secondes voir quelques minutes plus tard, il n'y a aucun effet. Ces alarmes sont conçues pour détecter des défaillances à long terme. De plus, la taille de la structure pouvant accueillir toutes les alarmes possibles est de taille fixe et proportionnelle aux nombres de canaux échantillonnés
Événements	Permettre à un usager devoir le comportement de certains signaux dans le temps lors d'une opération de l'équipement.	Prioritaire: on ne veut perdre aucune donnée lors d'un événement. Le taux d'arrivée des événements est aléatoire. On doit donc les conserver dans une file d'attente, lorsqu'on doit effectuer le traitement des signaux rapides.

La modélisation de la file d'attente s'effectuera exclusivement sur les événements puisqu'ils sont les seuls à utiliser les files d'attentes.

I.2 Arrivées des événements

Pour notre étude, nous avons pris un échantillon de 1500 événements, soit la quasi-totalité des événements qui ont eu lieu sur le disjoncteur PK/PKV installé au laboratoire Grande Puissance de l'Institut de recherche d'Hydro–Québec à Varennes depuis son installation en 1993.

De cette combinaison date/heure, nous avons pu étudier la distribution du nombre d'arrivées par unité de temps. L'histogramme du taux d'arrivée des événements par seconde est montré à la figure I.1.

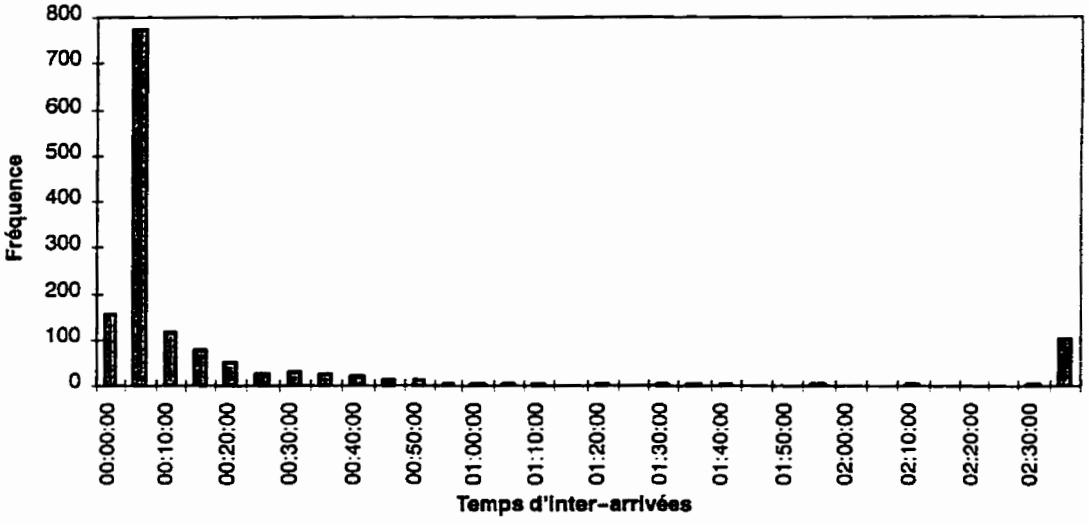


Figure I.1: Temps d'inter–arrivée des données originelles

On remarque qu'une partie importante des inter–arrivées dépasse 02:30 heures. Toutefois, nous ne tiendrons pas compte de ces temps, puisqu'ils sont dans la majorité du temps, causés par des pannes du contrôleur central. Nous nous intéressons davantage aux inter-arrivées de la région des 00:05:00, qui semblent correspondre à la vraie moyenne. Nous ne conserverons que les temps d'inter-arrivées inférieurs à une heure. Cette modification des données peut à premier abord sembler suspecte, car il est certain qu'il n'y a pas d'événements

physiques (i.e. ouvertures/fermetures de disjoncteurs) à toutes les heures, et qu'il peut s'écouler de grandes périodes de temps entre un événement et un second.

Cependant, dans le cadre de l'étude des capacités du système, il est préférable d'être prudent et de surcharger "artificiellement" le système que d'en surestimer les capacités en améliorant les données. Les nouvelles données, desquelles on a éliminé les temps supérieurs à une heure, sont présentées dans le graphique de la figure I.2.

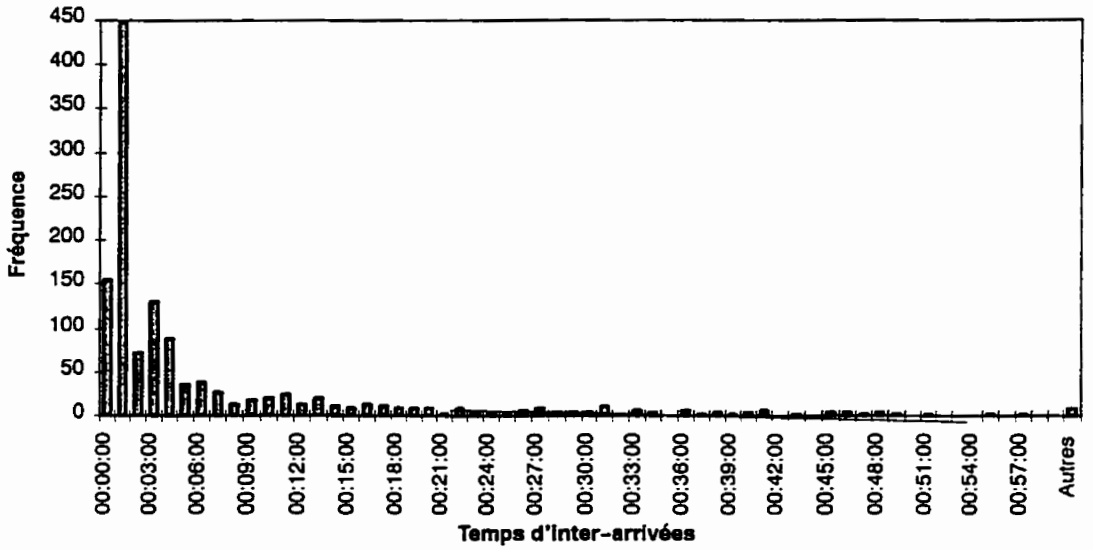


Figure I.2: Temps d'inter-arrivée des données ajustées

La nouvelle moyenne et écart type des temps d'inter–arrivées sont les suivantes:

$$\mu = 6{:}59{,}3 \text{ min}$$

$$\sigma = 14{:}35{,}0 \text{ min}$$

on peut donc dire que si on a une arrivée aux 6 minutes 59 secondes 3 centièmes (ou bien aux 419,3 secondes) en moyenne, que l'on a :

$$\overline{n} = \left[\frac{1}{419,3\,s}\right] = 0,002385\,Hz$$

On a donc $2{,}385 \times 10^{-3}$ arrivées par secondes, ou 8.586 arrivées à l'heure, et le temps moyen d'inter-arrivée $\overline{t_a} = 0{,}116772$ (i.e. 8.568^{-1}) h.

I.3 Modélisation du système

I.3.1 Éléments de base d'un système temps réel à contraintes douces

D'une façon générale, les systèmes temps réel à contraintes douces utilisent une série de files d'attente leur permettant d'accuser un certain retard, tout en conservant un fonctionnement acceptable du système. La figure I.3 illustre un système temps réel typique.

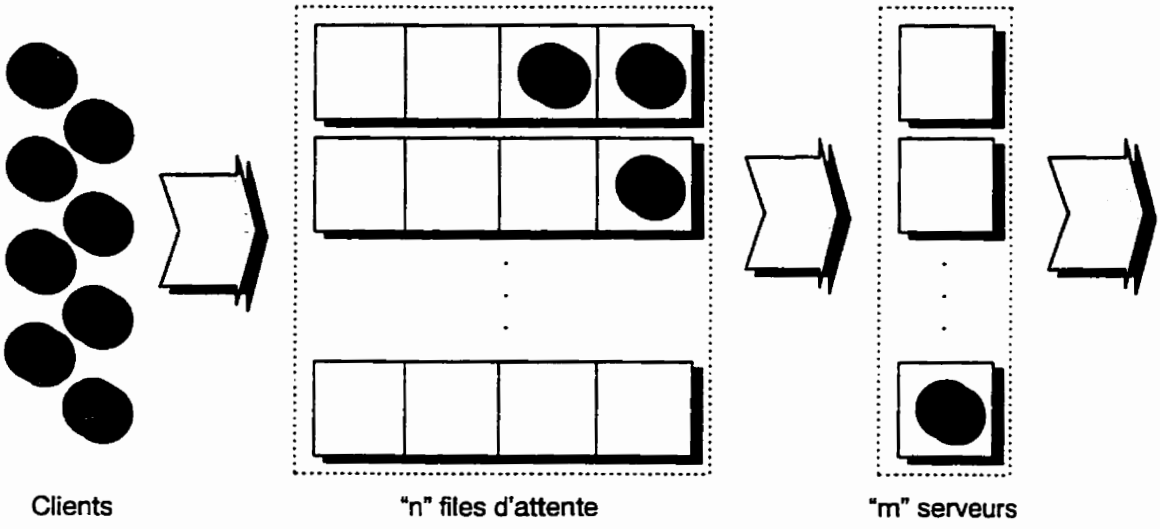


Figure I.3: Fonctionnement général d'un système temps réel à contraintes douces

Une population de clients passe à travers "n" files d'attente à un taux d'arrivée quelconque. Ils sont ensuite desservis à un certain rythme par "m" serveurs identiques.

I.3.2 Type de système

Le système temps réel du contrpoleur local est système à un seul serveur. Le tout ressemble au système suivant:

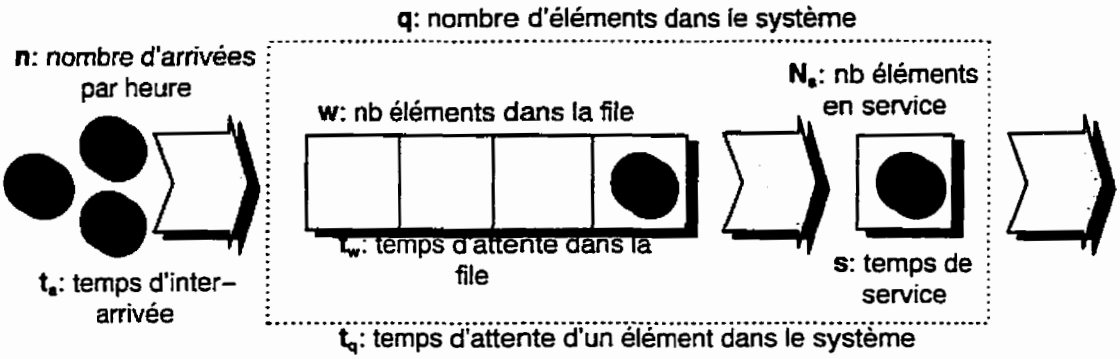


Figure I.4: Configuration du système temps réel étudié

Les variables utilisées pour modéliser le système sont donc:

- n: nombre d'arrivées à l'heure dans le système;
- t_a: temps d'inter-arrivée des événements;
- w: nombre d'événements présents dans la MCPRAM (le tampon circulaire);
- t_w: temps d'attente dans la file;
- N_S: nombre d'éléments en service dans le serveur STD. Cette valeur est constante et toujours égale à un;
- s: temps de service d'un événement par le STD;
- q: nombre d'événements présents à la fois dans le système en entier;
- t_q: temps d'attente total d'un événement dans le système (temps d'attente + temps de service).

De plus, comme nous l'avons mentionné à la section I.2,

- $\bar{t}_a$ = 0.116772 heure entre chaque arrivée en moyenne;
- $\bar{n}$ = 8,568 arrivées par heure en moyenne.

On doit aussi connaître le temps de service et son écart type. À cette fin, nous avons pris des mesures sur le système. Comme à chaque fois qu'un événement se produit le STD note la date et l'heure au début d'un événement, la moitié de nos mesures s'effectue automatiquement. Il ne suffit que de rajouter une procédure qui note l'heure lorsqu'elle calcule le CRC de la structure d'événement. Avec ces lectures, nous avons pu déterminer le temps de service moyen ainsi que l'écart type:

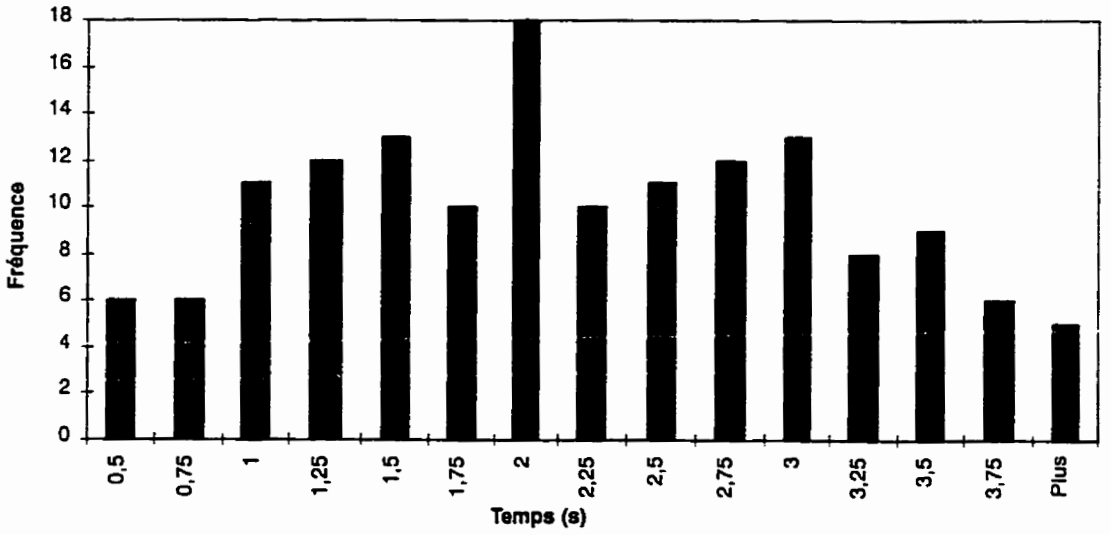


Figure I.5: Temps de service extraits du système

De ces données, on a pu tirer le temps de service moyen ainsi que l'écart type:

- $\bar{s}$ = 2.08 secondes = 5.78×10^{-4} h par événement traité;
- $\overline{\sigma_s}$ = 0,94 secondes = 2.61×10^{-4} h par événement traité.

I.3.3 Facteur d'utilisation

Avec les données obtenues jusqu'à présent, nous sommes en mesure de calculer le facteur d'utilisation ρ. Il est caractérisé par la formule suivante:

$$\rho = \frac{temps\ utilisé}{temps\ disponible} \tag{I.1.}$$

Toutefois, nous ne possédons pas directement cette information. On peut se servir de l'équation $\rho = \overline{n \cdot s}$ puisque nous sommes dans un état stable. On obtient ainsi un facteur d'utilisation très petit:

$$\rho = 8{,}568 \frac{événement}{h} \times 5{,}78 \times 10^{-4} \frac{h}{événement} = 0{,}00495 = 0{,}495\%$$

Puisqu'on a maintenant ρ, on peut donc calculer le temps moyen d'attente d'un événement dans la file d'attente.

I.3.4 Probabilité d'avoir une file d'attente de taille supérieure à X

Étudions le système avec les distributions d'Erlang. Tout d'abord, on doit déterminer quel sera le coefficient de la distribution. Ce coefficient nous est donné avec la formule suivante:

$$E = \left(\frac{\bar{s}}{\sigma_s}\right)^2 \tag{I.2.}$$

On retrouvera donc un coefficient d'Erlang équivalent à $\left(\frac{5{,}78 \times 10^{-4}}{2{,}61 \times 10^{-4}}\right)^2 = 4{,}9$ qui une fois arrondi à la meilleure valeur donne **5**. Le graphique suivant donne un aperçu de la distribution Erlang 5:

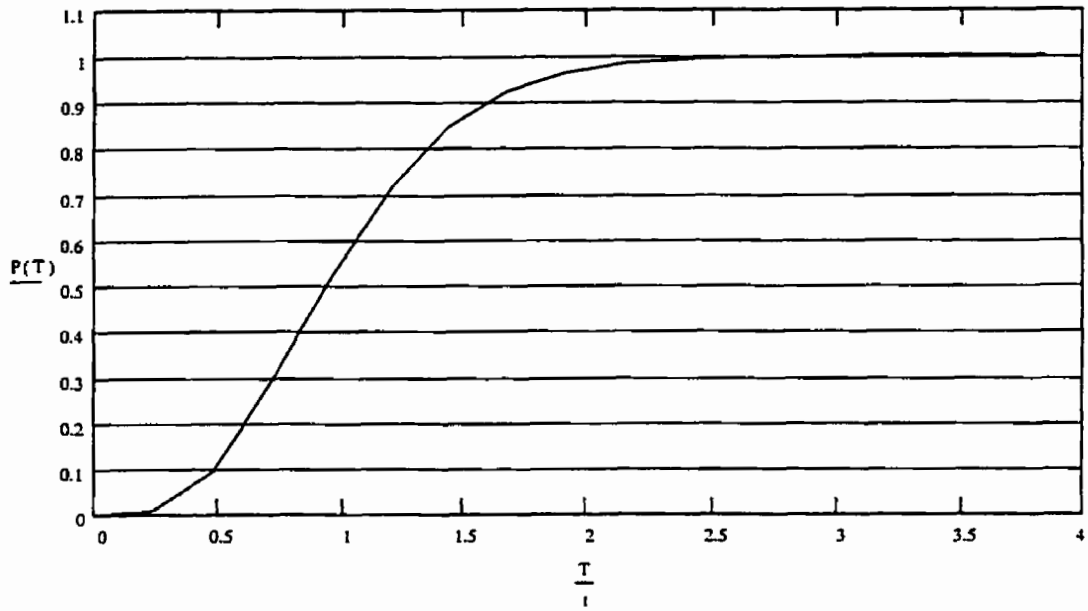


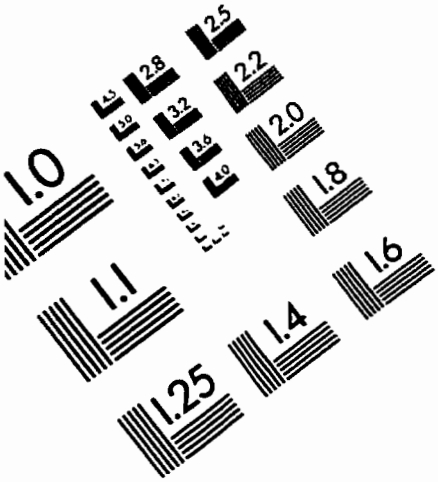
Figure I.6: Distribution Erlang 5 pour le système de surveillance

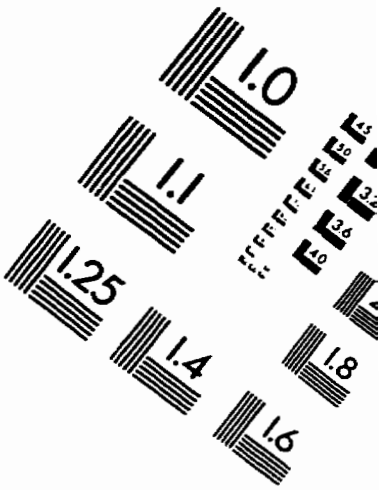
Cette fonction est en fait un cas particulier de la fonction gamma calculée à l'aide de la formule suivante:

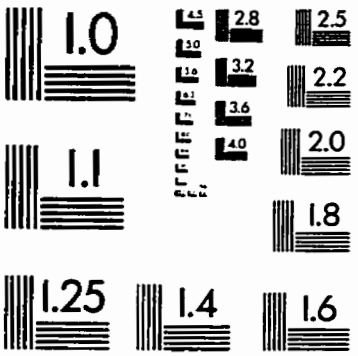
$$P(t \leq T) = \left\{ \frac{\int_0^T \left(\frac{E \cdot t}{\bar{t}}\right) \cdot e^{-\frac{E \cdot t}{\bar{t}}} \cdot \frac{E}{\bar{t}} dt}{\int_0^\infty \left(\frac{E \cdot t}{\bar{t}}\right) \cdot e^{-\frac{E \cdot t}{\bar{t}}} \cdot \frac{E}{\bar{t}} dt} \right\} \qquad \text{(I.3.)}$$

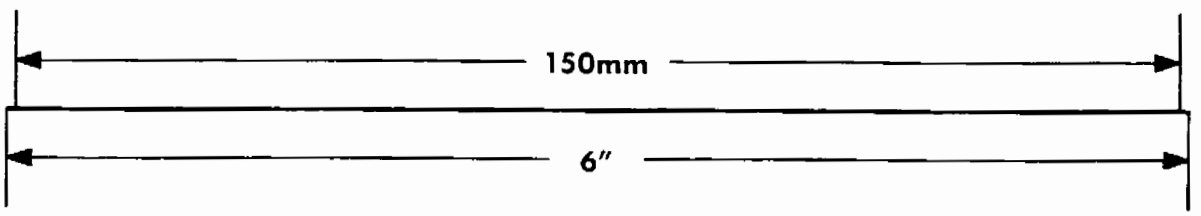
où E est toujours un entier positif.

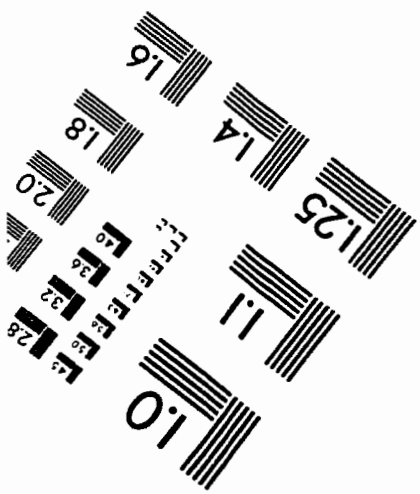
IMAGE EVALUATION TEST TARGET (QA-3)

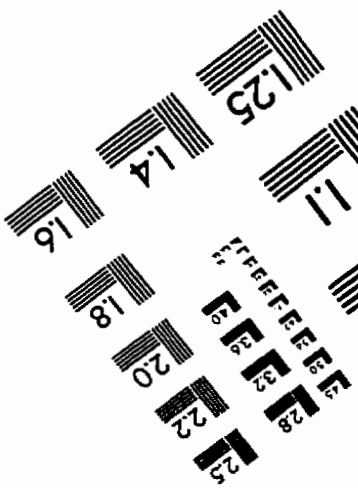












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989