THESE

Présentée à

L'UNIVERSITE DE METZ

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE METZ

EN ELECTRONIQUE

pour le sujet de recherche

Etude et réalisation du logiciel d'exploitation d'un Equipement de Communication pour Applications Distribuées (E.C.A.D).

par

CHARRIERE Claude

Présentée le 15 Décembre 1992 devant la commission d'examen:

Président du jury:	M. TOSSER-ROUSSEY Professeur à l'Ecole Nationale d'Ingénieurs de METZ
Rapporteurs:	M. LE BIHAN Professeur à l'Ecole Nationale d'Ingénieurs de BREST
	M. YVROUD Directeur de Recherche à l'Ecole Nationale Supérieure d'Electricité et de Mécanique de NANCY
	M. FAURE Professeur à l'I.U.T du HAVRE en génie éléctrique.
Examinateurs	M. LAURENT Professeur à l'Université de METZ
	M. ROUSSEL Professeur à l'I.U.T de TROYES
	M. PERRINE Professeur associé à l'Université de METZ et Directeur de la Direction Opérationnelle de la région Est de France Télécom.
Invités:	M. FLAENDER Directeur d'I&T COM (Innovations et Technologies en Communication)
	M. KOPP Ingénieur d'étude au Centre d'études en Radicommunications et Radiodiffusion de T.D.F.

Remerciements.

Ce mémoire présente un ensemble de travaux d'étude et de développement, réalisés au Centre d'Etude et de Recherche de Lorraine (CERLOR) de Télédiffusion de France, et au laboratoire Mécatronique Industrielle de l'Université deMetz.

J'exprime ma reconnaissance à Mr D.FLAENDER, directeur du CERLOR de m'avoir accueilli dans ce centre de recherche et d'avoir accepté d'être membre du jury ce cette soutenance.

J'exprime à Monsieur le professeur A.TOSSER-ROUSSEY professeur à l'Université de Metz ,et directeur du laboratoire Mécatonique Industrielle ma trés profonde gratitude pour la confiance qu'il m'a accordée et pour les conseils qu'il m'a prodigués tout au long de mes travaux.

Je remercie Ms LECLUSE ,LEBIHAN, YVROUD d'avoir accepté d'étudier avec minutie l'ensemble de mes travaux pour établir les rapports de thèse.

Que tous les membres du jury: Ms ROUSSEL, LAURENT, PERRINE soient également remerciés pour le temps qu'ils ont consacré à la lecture de ce rapport et les maux de tête subits pour comprendre ce long et fastidieux charabia.

Un certain nombre de personnes du CERLOR dont Mrs P.PIGNON, A.REFIK m'ont apporté de précieux conseils et leurs encouragements et je les en remercie vivement.

Ces trois ans que j'ai passé au CERLOR, ont été pour moi une expérience très enrichissante tant par les domaines techniques abordés ou approchés que par les relations personnelles avec l'ensemble des personnes du centre qu'ils soient administratifs ou techniques.

Sommaire

IV L'EXECUTIF MULTITACHE DU TERMINAL

V MODELES ET MECANISMES DE GESTION MEMOIRE

VI AUTRES ELEMENTS LOGICIELS DE L'EXECUTIF

IX ANNEXES

Présentation des travaux effectués

Ce mémoire présente les résultats de mes travaux d'étude, de conception et enfin de développement du logiciel d'exploitation du terminal E.C.A.D, poursuivis depuis trois ans. Mon espoir dans la rédaction de ce texte est d'exposer le plus clairement possible l'objet de ce logiciel sous tous ses aspects statiques et dynamiques et enfin de répondre aux deux questions; Pourquoi? et Comment?

Le but poursuivi est de réaliser un logiciel capable de proposer à un utilisateur quelconque des services de type domotique ou immotique, à partir d'une base matérielle spécifique.

Ce mémoire est structuré en huit chapitres, dont le premier, aprés avoir exposé le cadre général de développement de la domotique dans lequel se place le projet E.C.A.D, définit les contraintes techniques et fonctionnelles du terminal.

Le chapitre suivant expose les caractéristiques techniques matérielles du terminal et notamment celles des principaux composants.

Le troisième chapitre expose les principales propriétés et structures des systèmes d'exploitation ayant servi de référence à mes travaux de conception, puis décrit de façon générale (structure, mise en place, fonctionnement linéaire) le logiciel d'exploitation que je développe.

Le chapitre IV définit de façon précise les éléments constitutifs et communs des systèmes d'exploitation: les processus, la synchronisation des tâches, les stratégies et algorithmes d'ordonnancement. S'appuyant sur cette description, j'y expose en détail les différentes options et les mécanismes mis en place ainsi que les raisons de mes choix de conception pour constituer le noyau du logiciel, et notamment la fonction d'ordonnancement des processus.

La gestion de la mémoire constitue le deuxième grand mécanisme de l'exécutif et son importance nécessite un exposé dans le chapitre suivant (chapitre V) des différentes méthodes (disponibles) de gestion de ce périphérique ainsi que les mécanismes conçus et mis en place au niveau du terminal pour utiliser de façon optimum une ressource matérielle limitée en taille (gestion dynamique compacte et contigüe des enregistrements en mémoire).

Le traitement logiciel et l'organisation des interruptions permettant à l'équipement d'interagir avec son environnement sont décrits dans le chapitre VI. Sont décrits aussi les pilotes de périphériques et l'ensemble logiciel de gestion du réseau (sur le plan général puis particulier dans le cas du réseau Médiabus).

L'interpréteur de commande et le mécanisme de gestion de l'écran constituent le chapitre VII, suivi par une description rapide des processus utilisateurs mis en place et constituant les possibilités d'application de l'ensemble.

Enfin le huitième chapitre, apporte un certain nombre de conclusions et de propositions sur des développements ultérieurs.

Les annexes, renferment des éléments bibliographiques, un glossaire, les caractéristiques principales de quelques systèmes d'exploitation, et enfin la description détaillée des variables et fonctions utilisées dans ce logiciel.

I INTRODUCTION

I.1: La domotique

Aprés l'apparition dans les années 70 de l'informatique, et la percée de celle-ci comme industrie et comme activité économique (fourniture de services), une autre révolution se prépare en douceur et concerne l'ensemble du cadre de vie ainsi que les activités professionnelles, sociales et de loisirs de l'homme. Profitant de nouveaux moyens de communication développés cette derniére décennie, l'idée de donner à l'homme le moyen de consulter, commander, utiliser des services de n'importe quel lieu géographique (cabine téléphonique), à partir d'un équipement spécialisé a vu le jour. Le Minitel est la première illustration de cette volonté. Cette tendance à la communication et à l'automatisation des équipements se poursuit actuellement sous diverses formes telles que l'urbatique, l'immotique, la domotique. Ces trois termes recouvrent des systèmes de gestion automatisés faisant appel aux techniques de communication, d'automatisme, d'informatique, de gestion de bases de données s'appliquant à différents niveaux de gestion des équipements d'une ville.

L'urbatique concerne tous les équipements placés sur la voie publique et porte sur la gestion adaptative de ceux-ci. Ainsi la régulation des feux de signalisation au niveau d'une ville est effectuée en fonction de l'évolution du trafic routier observé, des habitudes de déplacements enregistrés préalablement, et des incidents de circulation (accidents, travaux programmés). L'affichage dynamique d'informations (routière, culturelles) en fonction du secteur géographique est un autre exemple de ce marché de la communication.

L'immotique est à court terme l'application la plus rentable et la plus rapidement mise en place du fait de son application à des ensembles collectifs qu'ils soient à usage d'habitation ou de travail. Les exemples d'application de l'immotique sont nombreux en France et en Europe (siège social de Bouyges, Institut du monde arabe). Ces systèmes régulent le milieu de vie (température, hygrométrie), les éléments d'accès (zones réservées), et les ensembles de déplacement interne (ascenseur, escalier mécanique).

La domotique a pour espace d'intervention l'habitat individuel par des systèmes de régulation de chauffage, des sources d'énergie, des systèmes de détection d'intrusion et d'accidents domestiques (incendie, inondation, fuite de gaz). Le marché de la domotique est peu développé car l'aspect normalisation technique n'est pas réalisé et parce qu'il dépend aussi du nombre et de l'attractivité des services mis à la disposition du grand public.

Les efforts entrepris en recherche, développement et normalisation dans ces trois domaines, sont encouragés par la communauté européenne sous forme de grands projets de type ESPRIT et font intervenir des groupes industriels de l'électricité (Legrand), de

l'électronique (Philips, Clemessy), ainsi que des universités. Ces trois types d'activités visent à créer un espace de vie intelligent pour l'homme aussi bien dans le cadre de son travail que de sa formation, de ses loisirs, et de sa communication. Ces projets visent aussi sur le plan économique au renforcement de la compétitivité des entreprises européennes des secteurs concernés, et à la conservation de la maîtrise de ces technologies.

I.2: Le terminal domotique (E.C.A.D)

Philosophie du terminal

Le projet de réalisation d'un terminal domotique s'inscrit dans celui de constitution d'un réseau collectif domotique.

La domotique s'envisage, de façon générale, dans le cadre d'une maison individuelle ou d'un appartement, pour automatiser, contrôler interactivement le fonctionnement de divers équipements de l'habitat. Les fonctionnalités classiques portent sur la régulation du chauffage des pièces (selon les consignes de l'utilisateur), sur des automatismes domestiques (ouverture modulée de stores), sur l'aspect sécurité par la détection d'intrusion, sur le relevé de consommation de fluides, d'énergie, etc.

TDF, dont la spécialité est d'acheminer les images TV du lieu de conception, de celles-ci, au foyer de l'utilisateur, conçoit et gère des réseaux de télédistribution. Il a été naturellement conduit à envisager d'utiliser le support matériel existant pour ouvrir un nouvel espace à la domotique.

Le phénomène incitatif résulte d'une part du désir de développer et au minimum de rentabiliser les réseaux de télédistribution actuels, et d'autre part de la volonté de promouvoir le développement de la domotique par son ouverture à l'extérieur du foyer domotisable (immotique ou urbatique).

A partir de l'opportunité de disposer d'un réseau de télédistribution existant, reliant un grand nombre d'abonnés, on utilise le câble coaxial (dont les limites de possibilité de transport ne sont pas encore atteintes) pour véhiculer de nouvelles informations alphanumériques.

L'ouverture de la domotique sur l'extérieur par des réseaux suppose que ceux-ci mettent à disposition des fonctionnalités et services de type collectif non disponibles au niveau de l'habitat individuel isolé. Cette forme ouverte de la domotique entre alors en concurrence avec le minitel utilisant lui, le réseau téléphonique.

De façon pratique, après redéfinition du plan de fréquences utilisé par le réseau câblé de la ville de Metz, il est nécessaire de construire des équipements de transmission des

informations de texte dans les deux sens (serveur <-> utilisateur) sur le réseau. Il est nécessaire d'implanter sur le réseau un (ou plusieurs) organe de régulation et de gestion des messages sur le réseau, et des terminaux qui peuvent êtres des automates de contrôle ou de mesure (compteur de consommation d'eau, capteur et régulateur de température) ou même de véritables terminaux à usage des utilisateurs. La première opération consistait à relier par le réseau un ensemble de logements regroupés dans la ZUP (zone d'urbanisation prioritaire) de Borny (METZ), pour le compte d'organismes ou entreprises tels que l'U.E.M (Usine d'Electricité de Metz), et de l'Office de HLM qui désirait l'utiliser pour assurer la gestion de ses équipements collectifs (ascenseurs) et de ses logements .

Dans un premier temps, les services utilisateurs n'ont été que peu développés, et en conséquence ne nécessitaient pas des terminaux utilisateurs très sophistiqués. La domobox est dans le cadre du projet d'équipement collectif le premier terminal utilisateur par lequel celui-ci peut bénéficier de services domotiques (programmation de la température). la domobox est un équipement microinformatique constitué d'un clavier et d'un afficheur permettant de visualiser des messages (émis et reçu). Cette première opération grandeur nature et limitée en taille permettait aussi de résoudre des problèmes non appréhendés auparavant.

Par la suite une passerelle utilisant le minitel comme terminal a été développée constituant ainsi un deuxième moyen d'accès aux services proposés.

Pour intéresser l'utilisateur, il faut disposer simultanément de deux éléments: un ensemble de services domotiques attractifs offrant un plus incontestable (par rapport au minitel) et un terminal graphique interactif d'utilisation aisée et instinctive. Le support et l'infrastructure de transmission existant devaient être simplement aménagés et adaptés. Dans la même optique d'utilisation d'un moyen technique existant et de son adaptation, le terminal devait utiliser l'appareil de télévision comme interface de communication visuelle avec l'utilisateur. De cette façon on évite d'implanter un nouveau terminal spécialisé; ce qui constitue la première innovation de ce terminal (fig n°I.1).

Terminal de contrôle et de visualisation

Ce terminal en étant connecté à deux espaces, l'espace collectif et l'espace de l'habitat individuel, par l'intermédiaire d'interfaces spécialisées occupe une position particulière de point central (ou noeud) de communication entre deux mondes. Par ce terminal, l'utilisateur peut contrôler et programmer des équipements de la maison (régulation pièce par pièce du chauffage) mais aussi communiquer avec un autre utilisateur en utilisant un service de messagerie via le réseau collectif domotique.

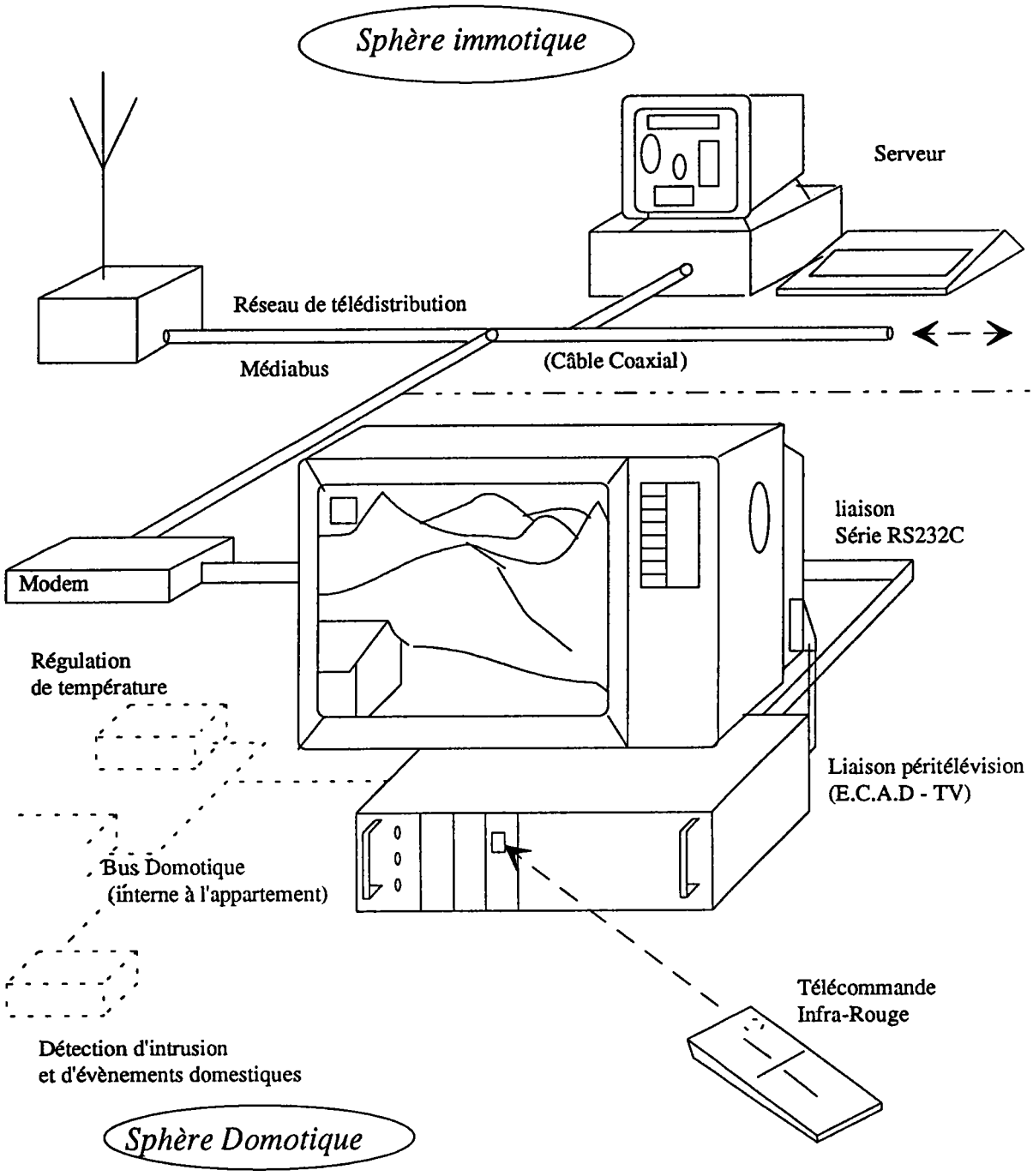


Fig n°: I.1: Environnement Immotique/Domotique du terminal (E.C.A.D)

Des services internes et externes complémentaires

Une autre ambition de ce terminal est d'être un terminal autonome proposant des services en propre (services internes résidents) ou liés au lieu de vie, disponibles de par la conception matérielle et logicielle du terminal (programmation d'alarmes, horloge permanente, etc). Ces services ne seront probablement pas les plus intéressants, mais viendront en complément de services extérieurs. L'existence de ce terminal n'est pas liée à la présence obligatoire d'un réseau télévisuel domotisé. Par contre son existence est conditionnée par un certain niveau d'équipement de l'appartement ou de la maison.

De multiples interfaces

La structure matérielle et logicielle du terminal doit être suffisamment modulaire pour offrir un bon niveau de souplesse d'évolution, d'installation, de configuration. Cette souplesse peut se traduire par le choix des interfaces et des protocoles de communication, le choix des capteurs et actionneurs en fonction des désirs des utilisateurs, des contraintes d'installation matérielles et de l'aspect financier. Pour relier par exemple un capteur de présence à l'unité centrale qui constitue le terminal, les moyens de communication utilisables sont variés: liaison infra-rouge, liaison radio-fréquence, fibre optique, paire torsadée, câble coaxial. Les moyens de communication ne sont pas les seuls éléments de modularité. On pourra concevoir des équipements et des cartes de traitement spécialisées (commande de moteurs d' ouverture/fermeture des volets, reconnaissance des personnes pour des zones protégées) sans pour cela que le logiciel de l'équipement ne devienne rapidement obsolète. Ces interfaces et services sont encore à concevoir.

Un automate programmable

L'utilisateur devra pouvoir contrôler le fonctionnement temporel de divers équipements de l'appartement ou de la maison. Ces dernières années ont vu fleurir les mécanismes de temporisation pour la mise en marche différée (machine à laver), pour l'arrêt programmé automatique (four), mais ces équipements sont restés sans organisation de fonctionnement global. L'intérêt de tels systèmes est qu'ils permettent des économies d'énergie (fonctionnement en heures creuses).

L'introduction de la domotique devrait faire évoluer certains équipements ménagers, non pas sur le plan de leurs fonctionnalités et sur la qualité de leur travail, mais sur la possibilité de commande manuelle ou programmée, intégrée ou déportée, sur un terminal.

Une interface banalisée

Laisser l'utilisateur en face de multiples terminaux spécialisés en fonction des services accessibles est l'une des raisons d'un rejet certain de ces nouvelles applications. Un seul type de terminal doit être utilisé et être capable de s'interfacer à différents ensembles de gestion ou de communication. Un appartement pourra comporter plusieurs terminaux d'aspects différents (tube cathodique, écran LCD).

L'ergonomie

S'adressant (en principe) à tous les utilisateurs, la commande des différents équipements doit être d'une ergonomie très poussée permettant un apprentissage et une utilisation aisée (pas de permis domotique). Les équipements devront par eux-mêmes offrir un niveau de sécurité très important, tant sur le plan de l'accès et de la conservation des informations que sur le plan fiabilité du matériel.

I.2: Les contraintes matérielles

Les contraintes de réalisation

Nous venons de voir les fonctionnalités que nous voulons voir proposer par ce terminal et qui en quelque sorte constituent la finalité de cet équipement. Cet aspect est complémentaire de celui ayant trait plus particulièrement aux caractéristiques techniques telles que rapidité, fiabilité, économie, strandardisation, protection. Cette liste n'est pas exhaustive.

La rapidité est une notion toute relative puisqu'elle s'envisage tant au niveau de la fréquence d'horloge associée au processeur (différente de la fréquence d'horloge du système multitâche) qu'au niveau du processeur avec son architecture, sa technologie, la largeur de son bus de données. La vitesse doit être adaptée à l'application. Une vitesse trop élevée entraîne des problèmes de rayonnements électromagnétiques et donc de conception des circuits pour les limiter. L'amélioration de la vitesse de traitement, de manipulation et de transfert des données est fortement limitée au niveau de l'interface avec les périphériques (temps de lecture/écriture, vitesse d'émission/réception).

Les performances d'un système se mesurent en terme de temps de traitement et de possibilité de réaliser des fonctions de faibles et fortes complexités, selon des règles de fonctionnement définies.

L'économie sur le plan industriel pour un produit fini doit intégrer le prix des composants et de leur conditionnement, le prix de la main d'oeuvre pour la réalisation, les tests et la maintenance et les coûts de développement. Pour réaliser une étude complète dans le cadre d'une comptabilité analytique, il est nécessaire de prendre en compte le temps passé en étude de faisabilité, de choix techniques, à la réalisation et au test d'un ou plusieurs prototypes; ensuite de prendre en compte le coût des éventuelles modifications ou adaptations sans oublier de chiffrer les coûts annexes (administration, transport, maintenance).

La fiabilité s'envisage sur plusieurs plans dont celui d'un taux de panne le plus réduit possible, mais aussi sur la capacité du système à détecter ses dysfonctionnements matériels et logiciels, et pour ces derniers à y apporter une solution sans l'aide d'un intervenant extérieur, ou pour le moins à poursuivre un fonctionnement global correct (hors dysfonctionnement local). Les solutions logicielles se traduisent essentiellement par un certain niveau de redondance, pour garantir le fonctionnement des modules de base. La fiabilité d'un système sous l'aspect prévention a aussi ses limites, au delà desquelles le coût de la fiabilité devient prohibitif.

La sécurité d'un système n'est pas relative uniquement à l'accès à certaines ressources ou données mais aussi à la sécurité électrique.

Il est frustrant pour un utilisateur de constater avec dépit que deux entités ne peuvent s'interfacer par défaut de standardisation. Le démon de la recherche de l'originalité fait que le concepteur utilise des solutions spécifiques pour obtenir un mouton à cinq pattes, et il devient difficile de se référer à de quelconques standards, de communication, de format de représentation et de stockage des données, d'alimentation électrique... La non utilisation d'une standardisation ne peut se concevoir que pour des raisons d'absence de matériel ou de procédure adaptée sur le marché.

D'autres éléments interviennent pour caractériser en bien ou en mal un équipement, tels que la non pollution électromagnétique générée par l'équipement et, en sens inverse , l'insensibilité du terminal à des perturbations extérieures (environnement et réseaux), et la confidentialité des informations présentes en mémoire et spécifiques d' une application et d'un public déterminé. Dans le cadre de la messagerie, il devra être impossible par l'intermédiaire d'un autre terminal ou d'un équipement de niveau supérieur (gérant) de lire des fichiers de ce service sur notre terminal.

Tous ces aspects techniques sont interdépendants et participent concurremment à la définition des qualités et des défauts d'un appareillage.

I.4: Spécificités des solutions

La partie matérielle du terminal n'offre pas de particularités marquantes quand à sa conception, si ce n'est l'utilisation de deux bus de communication pour relier les composants, dont le bus I2C. Cette structure en double bus fait apparaître deux ensembles de composants de fonctionnalités réparties, d'une part le noyau matériel minimum composé du microprocesseur, des mémoires, de l'horloge système et de l'interface série donnant accès au réseau collectif domotique (composants répartis sur plusieurs cartes), et d'autre part les composants I2C tels que récepteur et transcodeur de télécommande infrarouge, calendrier/horloge, compteur d'événements. Cette structure correspond à la philosophie du terminal exposée précédemment. L'interface de visualisation utilisée (le récepteur de télévision) permet de ne pas créer un nouveau terminal spécifique et les commandes de programmation de service du terminal passent par l'intermédiaire de la télécommande et de pictogrammes ou menus déroulants affichés à l'écran (objectif d'ergonomie).

A l'heure actuelle, les applications informatiques ou de communication faisant intervenir des sources d'information nombreuses et disséminées, organisées en réseau (quelles que soient la structure et l'architecture de celui-ci), se conçoivent dans une structure à "intelligence" (ou à traitements) répartie.

L'aspect "intelligence" de ce terminal repose sur le logiciel de gestion du terminal, les services qu'il propose et le mode de fonctionnement de celui-ci. Un certain nombre de contraintes de type temporel ayant trait aux événements extérieurs (message en provenance du réseau et commande de l'utilisateur par la télécommande) nécessitent une gestion de ceux-ci par interruption. Le logiciel de gestion du terminal ne peut être conçu à partir de l'idée de quelques services utilisateurs qui ne sont actuellement qu'à l'état d'ébauches. Pour garantir à ce matériel, une modularité logicielle et matérielle lui permettant d'évoluer et pour assurer l'utilisation optimum du matériel embarqué, j'ai choisi de constituer une base logicielle fixe, qui sur tout micro-ordinateur, constitue le système d'exploitation. L'ensemble logiciel (noyau+application) est implanté en mémoire ROM du terminal et est destné à assurer sa gestion et fournir différents services, est un logiciel d'exploitation multitâche.

II ASPECTS MATERIELS DU TERMINAL

II.1: Cahier des charges

La plupart des qualités techniques que l'on souhaite voir attribuer à l'ensemble se retrouvent tant au niveau logiciel qu'au niveau matériel.

Au démarrage du projet, avant la phase de réalisation matérielle de l'équipement il semble qu'aucun cahier des charges (en tant que tel) n'ait été défini. Cependant un certain nombre d'idées ont prédominé pour l'élaboration de la structure matérielle du terminal; la simplicité de conception, la modularité, l'ergonomie, et un faible coût de réalisation.

La simplicité se situe en partie au niveau du choix des composants. Utiliser des composants banalisés, microprocesseur et périphériques 8 bits, de technologie TTL et CMOS, des protocoles d'échanges classiques (liaison RS232C) ou des technologies bien maîtrisées au préalable, par le concepteur.

Modulaire en étant constitué de cartes électroniques indépendantes placées sur un fond de panier, l'équipement se doit d'être évolutif en permettant d'ajouter tous types de cartes (extension mémoire, nouvelles interfaces) répondant à un nombre minimum de conditions; utilisation du fond de panier prédéfini, respect des normes d'alimentation, etc.

Le coût de réalisation modique est directement lié à l'utilisation de composants banalisés.

II.2: Structure matérielle

Le prototype matériel est constitué de cinq cartes électoniques spécialisées, placées dans un boîtier modulable, au format simple Europe ("rack 19 pouces")(figure n°:II.1).

-une carte alimentation.

-une carte unité centrale:

batie autour du microprocesseur 68008, elle porte l'ensemble de la mémoire ROM et RAM de l'équipement, ainsi que la logique de décodage associée et les tampons d'Entrées/Sorties.

-une carte de visualisation graphique:

portant un processeur vidéo (celui des premières versions du minitel) ainsi que sa RAM privée pour la génération des images, écrans, symboles et icônes graphiques.

-une carte interfaces.

C'est la carte de communication du terminal car elle porte l'interface RS232C dédiée aux échanges d'informations avec le réseau domotique collectif

Médiabus et le boîtier interface du bus I2C ainsi que les composants qui y sont connectés.

-une carte horloge système.

Dernière carte développée, elle constitue après le processeur le composant le plus utilisé pour effectuer le partage du temps nécessaire au système de gestion multitâche.

Dans la version présentée, les trois cartes de base sont placées sur des emplacements spécifiques, pour réaliser les connexions à la prise péritélévision et au connecteur du port série (Canon 25) situés sur la face arrière du boîtier.

La liaison entre les différentes cartes est effectuée par un ensemble de lignes (fond de panier) portant les signaux de base du système (figure n°:II.2) et réparties en deux bus, un bus série (bus I2C) et un bus parallèle. Le bus parallèle de fond de panier ne reproduit pas le bus VME, jugé disproportionné par rapport à l'application. On dispose d'une version réduite du bus processeur notamment au niveau des lignes d'adresses.

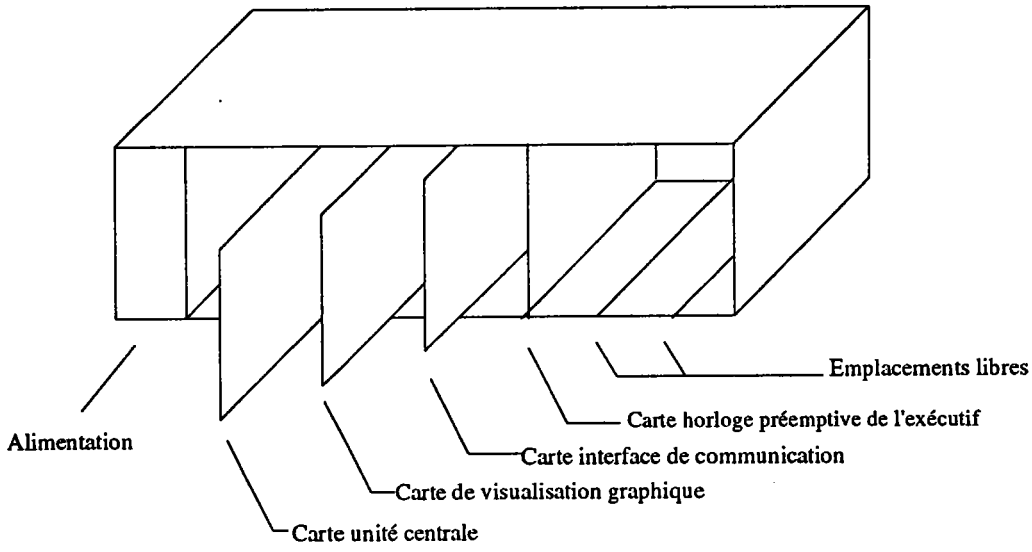


Fig n°II.1: Structure matérielle de l'unité centrale du terminal.

Le bus parallèle est utilisé pour connecter les composants de base du système; à savoir le microprocesseur et sa mémoire, le processeur graphique, l'interface série et le contrôleur de bus I2C faisant l'interface entre les deux bus.

Un bus série constitue le deuxième bus du système, utilisé pour relier les composants intégrés dans un deuxième groupe de périphériques. Ce bus est utilisé pour la réduction des liaisons entre composants qu'il apporte, et la simplification de la structure des cartes périphériques afin de réaliser une intégration plus importante.

L'interface série RS232C

Cette interface commune à tous les systèmes micro-informatiques est utilisée pour réaliser des échanges entre deux équipements selon un mode de communication série, et un protocole précis et normalisé (norme V24), autorisant des distances de transmission d'une centaine de mètres et des vitesses variables.

Le choix de cette interface, pour réaliser la liaison terminal-réseau Médiabus est dicté par la spécificité qu'elle possède de s'interfacer avec différents équipements et notamment avec un modem; modem connecté au réseau domotique collectif.

Le protocole de communication RS232C dont une trame est représentée par la figure n°:II.4 , consiste à envoyer des trains de données formatées (d'une dizaine de bits pour chaque donnée) à une vitesse et selon un format de présentation fixé des données au préalable, soit en mode unidirectionnel, soit en mode bidirectionnel alterné ou même bidirectionnel simultané.

Le composant utilisé, l'ACIA 6850 (Asynchronous Communication Interface) réalise le formatage des données et le contrôle de la transmission, en mode série asynchrone. Les éléments d'échange d'informations étant:

-le rapport de division de la fréquence de référence du signal d'horloge externe (division par 1, 16, 64). Pour disposer d'une plage plus large de fréquence, et d'une fréquence maximale plus élevé on peut avoir recours à un ensemble compteur, bascule, multiplexeur.

-la taille des données transmises est de 7 ou 8 bits.

-le contrôle de la transmission peut être réalisé par un bit de parité (contrôle avec parité paire ou impaire, contrôle sans parité).

-le nombre de bits d'arrêt: 1 ou 2 bits.

-l'autorisation de signaux d'interruption en réception

Ces éléments participant à la programmation de quatre registres internes sont accessibles par seulement deux adresses physiques:

-le registre d'émission accessible en écriture seule.

-le registre de réception autorisant seulement la lecture des données en provenance de l'extérieur (du modem) et positionné à la même adresse physique que le registre d'émission.

-le registre d'état de transmission portant des indicateurs de défauts de transmission (erreur de format, surcharge en réception, erreur de parité...) ou d'état des deux registres de transmission (registres pleins ou vides).

-le registre de contrôle programmé par les paramètres de transmission.

Autres particularités:

Le registre de contrôle n'étant accessible qu'en écriture il est nécessaire de garder trace de la programmation effectuée au niveau du registre sous forme d'une variable globale.

L'ACIA 6850 ne dispose pas d'une entrée de remise à zéro de ces registres (Reset), et seule une réinitialisation logicielle est possible (Reset Maître).

Les bus de communication

Pour relier les composants d'un système constitué d'un nombre limité de cartes électroniques rapprochées on utilise un bus de liaison de type parallèle. L'information est présente de façon complète et instantanée, et ne nécessite pas l'utilisation d'un protocole spécifique (mise à part la procédure de "hand-shake" matérielle). La fréquence de transmission entre composants est élevée: 8 Mhz.

Les paramètres de choix d'un bus de communication

Pour des éléments déportés placés à une certaine distance (une dizaine de mètres) et en environnement perturbé on utilise de préférence des bus de liaison de type série avec protocole de communication adapté. D'autres éléments justifient l'emploi de bus série tel que la simplicité de connexion et de réalisation des circuits imprimés.

Au niveau de la connectique le bus série apporte la réduction du nombre de liaison entre cartes et entre composants, ce qui facilite la conception des circuits (complexité moins importante, meilleure fiabilité, limitation des rayonnements électromagnétiques), et leur réalisation (réduction du nombre de pistes microscopiques et de trous métallisés, sur des circuits imprimés en double faces et huits couches). Pour des applications complexes la capacité d'intégration est très importante. Tous ces éléments d'ordre pratique ont naturellement une répercussion sur les coûts de développement, réalisation, maintenance.

Vitesse: 300/9600 bitd par seconde pour H=2.4KHz
Parité (contrôle): néant
Nombtre de bit de stop: 1stop
Taiile des données: 8bits

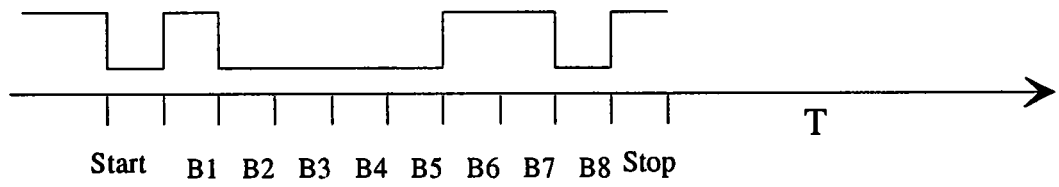


Fig n°II.4: Transmission série selon le protocole RS232C

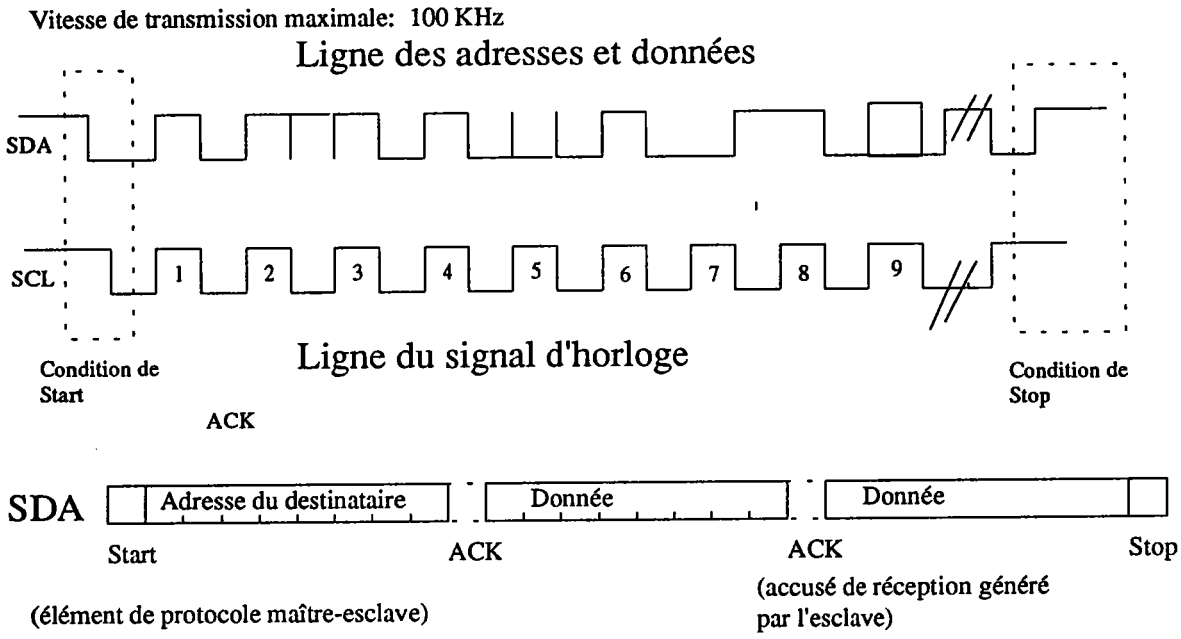


Fig n°II.5: Transmission série sur deux fils selon le protocole I2C

Parmi tous les bus ou systèmes de liaison série il reste à choisir celui qui correspond à l'utilisation désirée. Ce choix va se faire essentiellement suivant cinq paramètres qui sont; la vitesse de transmission, le protocole utilisé (formatage des données, organisation des échanges ou des accès, gestion des conflits), l'existence de composants de fonction maître ou/et esclave de fonctionnalité définie, les distances à franchir, l'infrastructure à mettre en place.

->La vitesse de transmission

-la vitesse de transmission est programmable et est imposée par le composant maître.

->Le protocole de communication:Le protocole adopté offre certaines possibilités techniques:

-le transfert de données entre deux composants est soit monodirectionnel, soit bidirectionnel en simultané ou alternativement.

-chaque composant a un double fonctionnement; émetteur et récepteur.

-un dispositif d'acquittement permet d'assurer la présence des intervenants et la validité des données transmises.

-le protocole de communication doit être suffisamment ouvert pour permettre l'ajout de futures applications.

-chaque circuit est référencé par une adresse définie matériellement ou de façon logicielle.

->Des composants de fonctions maître ou/et esclave

-un circuit dispose d'un mode d'intervention sur le bus en mode maître ou esclave. Certains composants bénéficient de ce double mode selon les circonstances.

-le composant maître du bus est celui qui est capable d'initialiser une communication et de générer le signal d'horloge synchrone.

-différents composants peuvent prendre le contrôle du bus en tant que maître, et un dispositif doit exister pour éviter les conflits de tentatives d'utilisation simultanées par plusieurs maîtres.

->La distance à franchir

-elle est liée et de façon inverse à la vitesse de transmission des données. Un bon paramètre de caractérisation du bus est ce produit: vitesse*distance. La limite de portée d'une liaison est conditionnée par le niveau de distorsions (par intégration) du signal et par des retards de temps de propagation induisant des collisions en échanges bidirectionnels.

->L'infrastructure à mettre en place.

-accepter la connexion de composants de différentes technologies (TTL, FET, BIPOLAIRE, NMOS, CMOS, ECL, I2L).

-le mode de transmission est de type série synchrone ou de type série asynchrone.

Mode de transmission synchrone/asynchrone

En mode synchrone, l'émission (et donc la réception) d'un ensemble de données est cadencée par une horloge et s'effectue sans pauses ni temps morts. Les liaisons dans ce mode sont bien plus rapides, mais plus contraignantes car les mots doivent se suivre les uns après les autres sans espace. La synchronisation entre l'émetteur et le récepteur s'effectue par le signal d'horloge utilisé en émission et en réception pour compter le nombre de bits transmis, et par un code de synchronisation sur un ou plusieurs octets placés en tête de trame (ou message).

Lorsque les données sont transmises de façon aléatoire dans le temps sans l'aide d'une horloge, on dispose d'une transmission en mode asynchrone. Cette transmission est rendue possible, par le formatage de chaque octet de données, qui comprend un bit de début, l'octet, un bit de contrôle d'erreur, et enfin un ou deux bits de fin . Chaque donnée se trouve ainsi "encapsulée" individuellement sous forme d'un paquet d'une dizaine ou d'une douzaine de bits. La resynchronisation en réception entre l'émetteur et le récepteur est effectuée sur chaque octet transmis par le bit de start.

Apperçu des différents bus ou liaison de communication

Les notions de bus et de liaison tout en recouvrant la même fonctionnalité (transmission bidirectionnelle entre deux équipements) se distinguent essentiellement par la portée (et les caractéristiques corollaires) et le formatage des données émises ainsi que par le mode de transmission utilisé (synchrone/asynchrone, parallèle/série). Presque tous les

fabricants d'équipements informatiques généralistes ou spécialisés ont développé leur bus ou leur liaison propre et parmi ceux-ci certains sont devenus des standards utilisés pour leurs spécificités.

-la liaison RS232C (V24); est une liaison asynchrone sur deux fils (+masse) utilisée comme interface (avec le réseau Médiabus via un modem spécialisé) pour relier des éléments distants selon des vitesses de transmission allant de 50 à 19200 bits/s, pour transmettre différents types de codes dont le code ASCII, et sur une distance maximale d'une vingtaine de mètres. C'est un standart de communication informatique.

-la liaison RS423 (V11); est conçue comme amélioration de la norme RS232C, pour la distance parcourue (600 mètres) et pour la vitesse de transmission (30 Kbits/s au maximum) sur deux fils.

-le bus IEEE 488; est un bus de communication hybride (série et parallèle sur 5bits) utilisé en instrumentation pour relier différents appareils de mesure entre eux dans le but d'organiser une gestion automatisée de ces différents équipements, et l'échange de données entre eux .

-le bus I2C accepte des composants de technologies différentes CMOS, I2L,NMOS, fonctionnant en multimaître, sur deux fils pour un débit maximum de 90 Kbits/s. Ce bus élaboré dans les années 80 a été développé pour les applications grand public telles que téléviseur, autoradio, équipement automobile, lecteur de disques compacts vidéo, CD ROM, téléphone.

-le bus Longworks est un bus série apparu récemment sur le marché et destiné à des applications, domotiques, immotiques, industrielles.

-les bus ISA (Industry Standart Architecture), EISA (Extended Industry Standart Architecture) sont des bus parallèles conçus dans le cadre de l'architecture PC, XT, et AT (8 Mhz-50 Mhz) des micro-ordinateurs IBM PC et compatibles. Le bus MCA est lui spécialisé à l'architecture IBM PS, mais n'offre aucune compatibilité ascendante avec les standards précédents

-les bus G64 (16 bits), G96 (32 bits), et VME (32 bits)sont des bus parallèles d'équipements industriels structurés pour des systèmes utilisant les microprocesseurs de famille 68xxx Motorola.

-les Multibus II et III sont aussi des bus parallèles d'équipements industriels constitués autour de microprocesseur Intel 16 bits.

-le D^2 bus est un bus de communication distant à vocation domestique.

La solution retenue; le bus I2C

Le bus retenu a été celui du constructeur Philips RTC. car il représente le meilleur compromis entre vitesse/distance, protocole, compte tenu de l'application que l'on souhaite en

faire, c'est à dire implanter un deuxième bus utilisé pour relier des composants avec une connectique réduite, et constituant un couche matérielle évolutive et d'importance secondaire. Ainsi le terminal dispose de deux bus reliant chacun deux ensembles de composants d'importances inégales (structure qui se retrouve au niveau de la hiérarchie des interruptions). Le bus I2C regroupe tous les éléments participant aux fonctions d'interfaçage avec l'extérieur et d'applications internes du terminal, (alarme, télécommande...) et le bus série les composants de base du système matériel et de l'interface de communication avec le réseau domotique collectif.

Bus et réseaux

Faire la distinction entre bus et réseaux devient une opération délicate. Les revues techniques spécialisées décrivent des bus domotiques, des bus de terrain qui correspondent plus par certains aspects à des réseaux.

La notion de bus était initialement associée à l'idée de transmission parallèle des données, impliquant une portée limitée, et utilisée à l'intérieur d'un équipement microprogrammé. Les protocoles d'échanges entre composants sont du type "hand-shake" (poignée de main), intégrés dans les composants.

La notion de réseau apparaît plus destinée à des équipements distants (de quelques mètres, à quelques milliers de kilomètres), réalisant leurs échanges à l'aide de protocoles de communication logiciels. Ces programmes de gestion des échanges constituent une couche logicielle importante et distincte de l'équipement.

La frontiére séparant le bus du réseau est de plus en plus floue, du fait de l'intégration sous forme microprogrammée de protocoles de communication complexes (exemple; le bus Longworks de la société Echelon et soutenu par Motorola et Toshiba). Les bus évolués actuels empruntent de plus en plus souvent les caractéristiques d'un réseau, et inversement. Un bus pourrait peut être défini comme étant une artère de communication ouverte non cyclique.

Le seul point commun est que les bus et réseaux vérifient la même structure de normalisation ISO.

L'interface graphique

L'écran de TV utilisé conjointement pour diffuser les programmes télévisuels classiques et les services "domotiques" est géré par un processeur vidéo pour les effets graphiques qu'il peut proposer tels qu'incrustation vidéo, choix des couleurs de caractères et de fond dans une palette de couleurs, choix de types de caractères, résolution...

En mode 40 caractères par lignes, l'utilisateur pourra programmer un affichage en double hauteur, double largeur, en clignotement, en inversion vidéo, en soulignement, en insertion, avec un ensemble de jeux de caractères alphanumériques prédéfinis, et une possibilité pour le développeur de créer son propre jeu de 128 caractères semi-graphiques.

L'autre résolution (80 caractères par ligne) n'offre que quelques unes de ces possiblilités (clignotement, inversion, souligné, sélection des couleurs).

Sur le plan technique, ce contrôleur graphique par ces 14 registres programmables, dispose d'un générateur de caractères semi-graphique, offre la compatibilité au standard TV 50-60 Hz en mode entrelacé ou non entrelacé, permet une synchronisation aisée avec une source vidéo externe par un comparateur de phase. Côté microprocesseur le bus données/adresses est multiplexé, et compatible avec les microprocesseurs ou microcontrôleurs 8 bits des familles Motorola et Intel. Les images et caractères semi-graphiques sont fabriqués et stockés en mémoire privée avant leur affichage. Selon la résolution et la taille de cette mémoire on peut, selon les résolutions, stocker de quatre à huit pages d'écran (mémoire de 16 Ko). Cette mémoire privée est au choix de type statique SRAM ou dynamique DRAM.

La structure mémoire

Le microprocesseur offre la possibilité de disposer de quatre espaces mémoire différenciés de 1 Mo chacun soit en mode protégé, soit en mode normal. Le prototype est configuré matériellement pour n'utiliser:

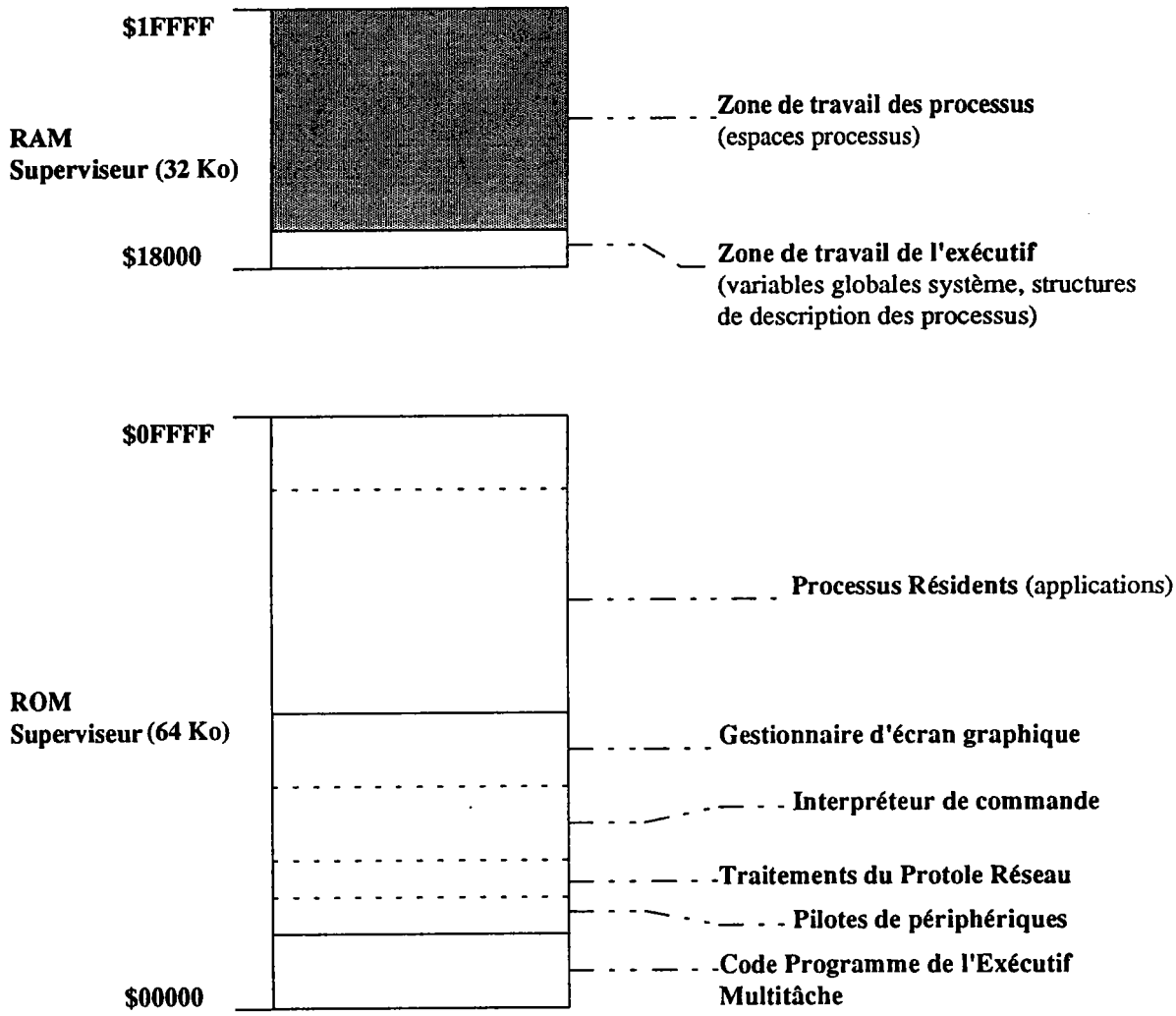
-qu'un espace ROM superviseur de 64 Ko

-qu'un espace RAM superviseur de 32 Ko

Cette structure (fig n°II.6) ne convient pas, à l'implantation d'un système complexe tel que celui qui est envisagé, par les risques de chevauchement des différents types d'informations qu'elle comporte. D'autres raisons plaident pour une restructuration de l'espace mémoire global, en trois ou quatre zones.

Cet équipement matériel, une fois réalisé, ne prendra vie que lorsqu'un logiciel exploitant pleinement ses possibilités y sera implanté. C'est la structure logicielle permettant d'arriver à ce but que nous allons définir dans les chapitres suivants.

Figure II.6 : Structure et occupation actuelle de la mémoire du terminal.



III LE LOGICIEL DE GESTION DE L'EQUIPEMENT

III.1 Premier développement

Pour réaliser les objectifs assignés compte tenu des contraintes matérielles le logiciel de gestion du terminal à réaliser devait être modulaire, évolutif et présenter certaines innovations.

Un premier logiciel basé sur l'exécution de services "domotiques/immotiques" fut développé en Juillet 1990, à partir des fonctions de gestion et de contrôle des différentes interfaces en place à ce jour, et organisées pour offrir différents services dont la messagerie, les alarmes, le télévote. Un micro-ordinateur Olivetti jouait le rôle de serveur à partir d'un logiciel de génération de trames d'informations en simulant un réseau. La nouveauté de l'équipement résidait dans l'utilisation de l'écran TV pour le contrôle et la visualisation graphique des résultats par l'emploi d'icônes, menus déroulants, et d'écrans de texte. La domotique étant encore en phase de balbutiement, aucun service intéressant particulièrement l'utilisateur ne s'imposait. Bâtir le logiciel autour d'une offre de services non définie précisément, et conçus à l'origine pour un réseau faisait du terminal un outil fermé, limité à quelques applications et ne correspondant à rien en cas de définition ultérieure de services, de supports matériels et de protocoles de communication autres.

les fonctions de programmation de base des composants, écrites pour ce prototype sont reprises et éventuellement modifiées dans la version actuelle du logicielle d'exploitation.

Une structure logicielle ouverte et spécifique était à définir; structure portant sur l'organisation et la gestion du logiciel. Organiser le logiciel consiste à définir les fonctions de base de gestion de chacune des interfaces et à les regrouper en bibliothèques, puis à un niveau plus élevé à les lier pour réaliser des opérations puissantes et de haute complexité et généralement développées à l'aide de langage de programmation évolué.

A ce stade il est devient nécessaire de définir une politique d'exécution des fonctions agglomérées en applications de services. Faut-il poursuivre le développement dans la voie de l'exécution des services de façon séquentielle? Peut-on s'inspirer de mécanismes de gestion des ressources utilisés sur certains micro-ordinateur? Avant de choisir il est nécessaire de définir les différents types de logiciels et quelques exemples illustrant ceux-ci.

III.2 Systèmes d'exploitation, exécutifs, multitâche multi-utilisateur, temps réel

Chaque micro ou mini-ordinateur dispose sur le plan logiciel d'un système d'exploitation dont le rôle est de gérer au mieux les ressources matérielles. Ce logiciel est d'autant plus nécessaire que les logiciels d'application pour être portables doivent rester indépendants des spécificités de chaque matériel. On a à l'esprit l'exemple des micro-ordinateurs compatibles IBM PC . Chaque constructeur a développé ses propres machines, toutes conçues selon une architecture de Von Neumann, mais suffisamment spécifiques pour se différencier et présenter des indices de performance et de compatibilité différents (Amstrad 1640 compatible IBM PC AT :taux de compatibilité logicielle de 75 %, taux de compatibilité matérielle de 55%). Les éléments même du système d'exploitation MS DOS entre plusieurs fabricants présentent des différences notamment au niveau du BIOS. Les différences ne portent pas seulement sur les composants employés mais aussi sur la gestion, le stockage et la représentation des données. Par exemple le système peut être chargé à partir du disque (disquette ou disque dur) à la base ou à l'extrémité supérieure de la RAM centrale.
Le système d'exploitation, outil de gestion du matériel, effectue des traitements sur requêtes, soit à partir d'une ou d'un ensemble de commandes utilisateur (mode interactif) qui seront traitées immédiatement ou ultérieurement (traitement en mode différé) suivant les critères de planification définis par le concepteur du progiciel (traitement par lots). Les requêtes peuvent aussi provenir de capteurs ou d'autres dispositifs matériels contrôlés par l'ordinateur et qui impliquent une réponse "en temps réel", à très court terme.

Les systèmes d'exploitation se classent en trois grandes catégories selon le principe d'exécution des programmes utilisateurs:

Monotâche, mono-utilisateur.

Le système est constitué autour des trois éléments qui sont le gestionnaire de fichier en arbre, le gestionnaire de périphériques construit à partir de toutes les fonctions des interfaces de base et de l'interpréteur de commande agrémenté de fonctions de gestion des fichiers spécifiques (fichier de traitement par lots). MS DOS en est l'exemple le plus connu.
Un seul programme applicatif utilisateur monopolise l'ensemble des ressources matérielles du micro-ordinateur.

Multitâche

Plus complexes, ils proposent deux systèmes de gestion importants: le système de gestion en temps partagé des programmes utilisateurs, et un système de gestion de la mémoire (principale et secondaire), ressource commune à toutes les applications. Des mécanismes "annexes" visent à prévenir les éventuels conflits d'accès aux ressources matérielles, et les

conflits d'exécution. Un interpréteur de commande est lui aussi disponible et propose des fonctions de manipulation liées au noyau multitâche. Ces systèmes sont comme OS-2 implantés sur disque et utilisés sur micro-ordinateur, pour le développement de logiciels, pour des applications bureautiques, pour des simulations de tous ordres (électronique, mécanique...) de taille raisonnable. Ce sont essentiellement des systèmes interactifs.

Multi-utilisateur

Faire évoluer un système multitâche en un système multi-utilisateur représente un investissement moins important que de partir d'un système monotâche pour le faire migrer en un système multitâche. Il reste cependant à augmenter le nombre et le niveau des protections d'accès aux ressources entre plusieurs utilisateurs. Le type même de ce genre de systèmes interactifs est UNIX, qui fonctionne selon les versions sur micro ou mini-ordinateurs, pour tous types d'applications d'orientation scientifique.

Selon l'utilisation et la taille que l'on veut donner à l'ensemble informatique ou microprogrammé le système doit posséder certaines spécificités aptes à lui permettre de réagir correctement dans un ensemble défini de cas. Un système multi-utilisateur de gestion de comptes bancaire n'a rien de commun ni avec un système d'exploitation multitâche d'une station de conception graphique, ni avec un exécutif multitâche intégré dans un lecteur de disque laser vidéo CDI. Chacune de ces applications induit par son mode fonctionnement et les contraintes imposées un type de gestionnaire de l'équipement en cause.

Le temps réel

La notion de temps réel est difficile à définir précisément car elle est relative aux aspects temporels d'interaction de l'équipement avec son environnement, ce qui implique que le système réagisse de façon adaptée et en un temps compatible avec les évènement qu'il doit percevoir et traiter [réf:]. Les temps de réaction à une sollicitation et de traitement sont de l'ordre de la dizaine à la centaine de microsecondes.

Présentation de quelques systèmes d'exploitation ou d'exécutifs

Différents systèmes d'exploitation ont été étudiés pour couvrir l'ensemble des systèmes d'exploitation pouvant servir de points de repère aux choix tactiques et stratégiques des mécanismes à mettre en oeuvre dans le logiciel d'exploitation du terminal.

A côté des systèmes d'exploitation les plus connus (UNIX, GCOS, VMS, OS9, DOS, OS2, VRTX), il existe une palette très étendue de systèmes répondant à des applications spécifiques. Ils utilisent des mécanismes de gestion adaptés en fonction de l'environnement

matériel sur lequel ils sont implantés (mini ou micro-ordinateur) et de la vocation du système global [ref:]. A partir de ce dernier aspect il est possible de les répartir en trois groupes;

-1)Pour les applications de type industriel:

Les contraintes temporelles de traitement et de gestion des travaux (délai et débit), ainsi que les contraintes d'implantation matérielle du système de régulation vont dicter le choix du système. Un système de régulation sur une chaîne d'épépinage de maïs est basée sur un système VME-UNIX. Lorsque l'environnement est particulièrement difficile (poussières incompatibles avec un disque dur) et que le système doit gérer un ensemble d'équipement complexes (bandes de transport, chariots mobile, four) sur une chaîne de production de tuiles, on utilise alors un système d'exploitation tel que OS9 implanté en ROM.

Ces systèmes sont dédiés à la gestion des ressources utilisées par des applications définies et conçues sur mesure (configuration paramètrée). Ils peuvent néanmoins supporter des ensembles de développement logiciel ainsi que des interfaces utilisateurs (écran, clavier) plus ou moins ergonomiques (Xwindows pour OS9).

Certaines caractéristiques structurelles du système sont attachées aux propriétés du système. Ces logiciels faisant l'objet de licences d'exploitation il est difficile de connaître exactement leur structure et leur mode de fonctionnement interne. On peut néanmoins en avoir une idée par l'étude de leur caractéristiques externes et en connaissant les grands principes de la stratégie de gestion des tâches. Ainsi Real Time Craft privilégie une gestion des processus par priorité (autant de files d'attente que de niveaux de priorité), avec ou sans préemption, et attente sur évènements (voir l'annexe n°6). OS9 reprend cette philosophie de priorité affectée à chaque processus, mais avec une file d'attente et une évolution du niveau de priorité en fonction de l'âge du processus.

-2)Pour les applications de gestion:

La gestion des comptes bancaires ou un système de réservation centralisé (réservation des places d'avion) nécessite des systèmes d'exploitation gérant d'énormes masses de données et d'opérations. Ils sont installés sur des unités de gestion et de stockage de grandes dimensions (disques et bandes magnétiques) et reliés à un grand nombre de terminaux ou de systèmes intermédiaires distants par liaisons spécialisées (Transpac). Le type d'applications installé est celui des SGDB (système de gestion de bases de données). L'architecture de tels réseaux informatiques est étoilée. Leur fonctionnent repose sur la gestion d'évènements en provenance de terminaux et sur les traitements séquentiels des opérations non urgentes. De ce fait leur mécanisme de commutation des tâches privilégie d'une part l'ordre de survenance des évènements extérieurs et d'autre part une gestion avec file d'attente unique en FIFO (premier arrivé, premier traité).

-3)Pour des équipements scientifiques:

Ils sont conçus en fonction de la taille de l'équipement et du nombre d'intervenants simultanés (Mini et micro-ordinateur), mais ils restent aptes à effectuer tout type de travaux scientifiques tels que: traitement d'images, développement de logiciels natifs ou croisés, recherche de nouvelles formes de programmation (langage naturel, langage orienté objet) ou d'applications (reconnaissance vocale ou textuelle) et enfin conception et simulation. Les systèmes d'exploitation utilisés UNIX, VMS sont généralistes, multi-utilisateur ayant de grandes capacités de calcul. Leur fonctionnement est basé sur des mécanismes de gestion par priorité et préemption. Par exemple, Unix gère ses processus selon leurs niveaux de priorité qui évoluent au fil du temps (voir l'annexe n°6); leur niveau de priorité diminue dans le temps (mécanisme inverse de celui d'OS9). Une seule file d'attente fonctionnant selon le principe du tourniquet est utilisée. Rappelons que le système Unix est multi-utilisateur, multitâche, mais ne permet pas de répondre en temps réels aux évènements. Il peut acquérir cette propriété par adjonction d'un module temps réel spécifique Chorus.

III.3 Positionnement du logiciel d'exploitation par rapport à quelques systèmes

Le logiciel de gestion du terminal ne prétend pas rivaliser sur le plan des performances temps réel, de la compacité du code généré, avec les meilleurs systèmes d'exploitation (tel que OS9 utilisé en référence). Il se conçoit plutôt comme un exécutif multitâche ayant une capacité à répondre rapidement à des sollicitations externes définies (processus externes), et aussi à effectuer des traitements de priorité temporelle moindre mais de complexité supérieure (processus internes).

Des solutions particulières sont définies pour répondre à des contraintes matérielles sévères. Ainsi la gestion de la mémoire centrale limitée est optimisée du fait de l'impossibilité d'utiliser des techniques de mémoire virtuelle (pas d'unité de stockage telle que disquette ou disque dur). Ce terminal doit proposer un niveau de sécurité de fonctionnement élevé accompagné de mécanismes de réinitialisation gradués et sélectifs des dysfonctionnements en fonction du défaut détecté.

Son autre particularité repose sur les différentes sources d'informations auxquelles il est amené à se connecter; sur un réseau collectif immotique (Médiabus) desservant un immeuble, un lotissement, un quartier, et sur un réseau domotique domestique interne à l'appartement (par exemple Batibus) . Le terminal de communication étant un noeud de communication entre ces deux espaces et permettant à l'utilisateur d'y avoir accès à partir d'un même boîtier.

Le terminal a pour vocation de donner accès à des services et fonctionnalités diversifiés proposés aussi bien par cet équipement que par les serveurs de réseau immotique,

et les équipements domestiques. Il constitue ainsi un systèmes de gestion centralisé domestique et un point de communication ouvert sur l'extérieur. Ce terminal ne doit pas être un équipement esclave. Pourquoi ce terminal ne gèrerait-il pas seulement l'aspect domestique (chauffage, alarmes...)? Pour cela il faut en faire un équipement à "intelligence répartie".

Doté d'une interface graphique, et de programmes utilisateurs résidents et importés et d'un exécutif multitâche ce logiciel est un hybride pour lequel le terme de logiciel d'exploitation est le plus approprié.

III.4 Structuration en couche, et implantation du logiciel en mémoire

Structure en couches

En phase de conception, pour des raisons de modularité, de clarté du développement et de distinction des fonctionnalités (ensemble de fonctions regroupées en agence), le logiciel du terminal a été structuré en trois couches logicielles (fig n°III.1);

-le moniteur de l'exécutif auquel est affecté la gestion en temps partagé des programmes applicatifs, et la gestion de la mémoire centrale de l'équipement. Bien que, de façon générale, l'ensemble des fonctions de gestion de la mémoire ne soit pas considéré comme partie intégrante du noyau, il a fallu l'implanter dans la couche la plus basse du logiciel car il se révèle pour ce projet être aussi important que celui assurant la multiprogrammation des applications.

Les fonctions de l'exécutif portent d'une part sur l'ordonnancement, la synchronisation, la communication et l'environnement d'exécution des processus complétées par les routines de traitement d'interruption, et d'autre part sur les fonctions d'allocation et de commande de désallocation d'espaces mémoire. A ces fonctions s'ajoutent les fonctions élémentaires et évoluées de pilotage des périphériques essentiels.
La mise en place de ces différents éléments et le schéma général de fonctionnement de l'exécutif est confié à un ensemble d'instructions constituant "la colonne vertébrale" du logiciel et se termine par une scrutation en boucle infinie des files des travaux à effectuer.

-l'interpréteur de commande et le gestionnaire d'écran, éléments visibles de l'utilisateur, lui permettent de commander l'exécution des programmes applicatifs domotique ou immotique et de réaliser d'autres actions (configuration du terminal). Les fonctions de haut niveau de cet ensemble concerne l'agence de gestion du processeur vidéo et interviennent sous le contrôle du noyau gérant des ressources matérielles.

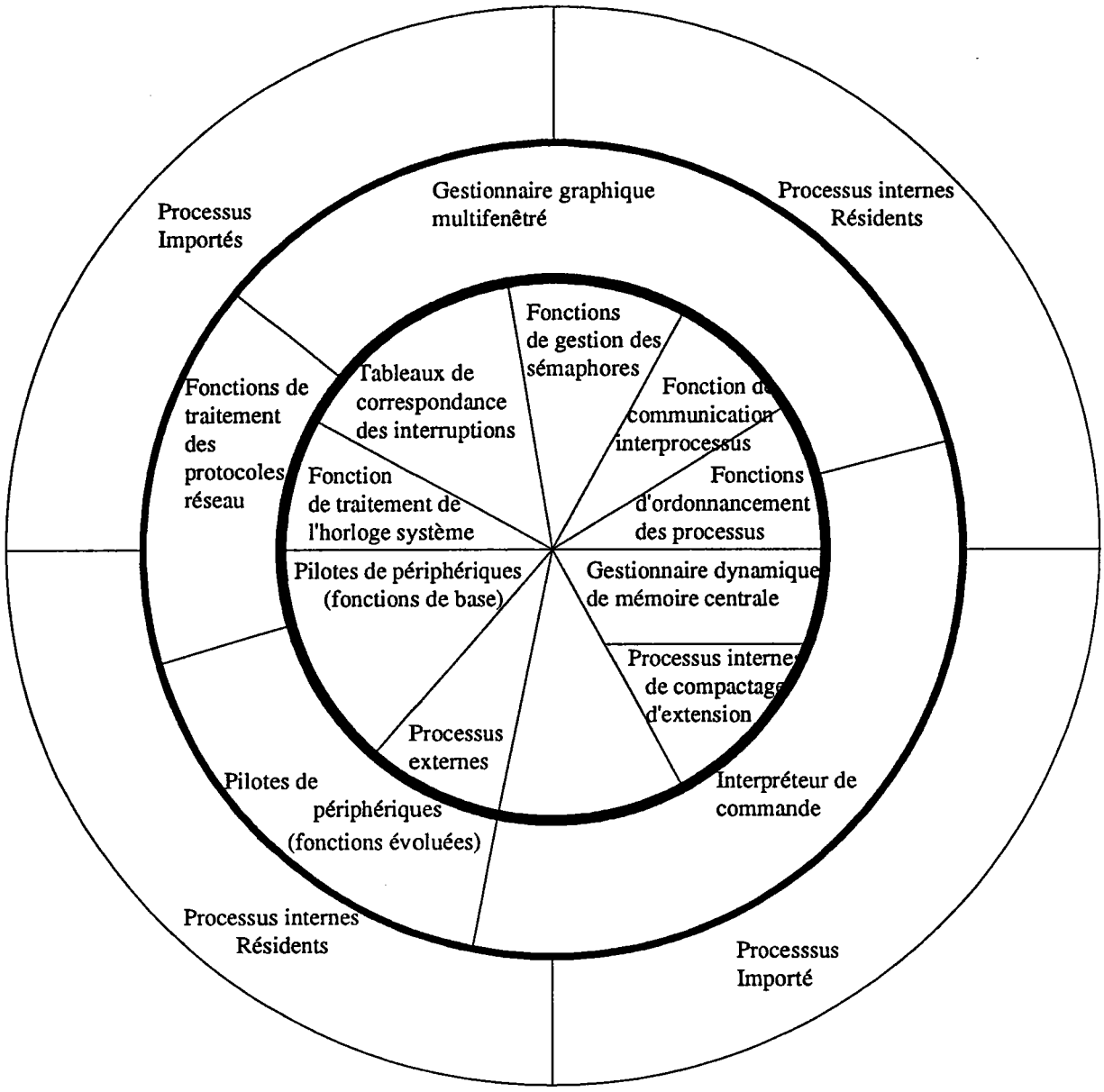


Fig n°III.1: Structure en couches du logiciel d'exploitation du terminal

-L'ensemble des programmes applicatifs résidents propose des fonctionnalités disponibles au niveau du terminal lui-même, et quelques applications impliquant un réseau immotique. D'autres applications utilisant les fonctions des couches inférieures, pourront être ajoutées ultérieurement (mise à jour des ROM) ou être téléchargées.

Implantation

Dans le but d'assurer une maintenance et une mise à jours aisée ces trois ensembles hiérarchisés peuvent être répartis en deux composants de mémoire permanente (EPROM, EEPROM). La couche exécutive multitâche, étant intégrée complètement dans un premier boîtier (ROM1; $00000000-$00007FFF) ne nécessitera pas un remplacement fréquent.

Quelques fonctions de l'interpréteur de commande s'y trouveront aussi. Les programmes applicatifs résidents et les fonctions graphiques de haut niveau seront, elles chargées dans un deuxième composant (ROM2; $00008000-$0000FFFF) et susceptibles d'être changées fréquemment.

III.5 Fonctionnement linéaire du logiciel

A partir de la mise en marche du terminal, le logiciel va connaître trois grandes phases d'exécution (fig n°III.2), correspondant chacune à un certain nombre d'opérations ordonnées.

-1) La phase d'initialisation des composants périphériques de base de l'équipement; à savoir le port série, les composants connectés au bus I2C et l'interface entre ce bus et le bus parallèle, le processeur vidéo, et enfin l'horloge système. A la fin de cette phase toutes ces interfaces ainsi que la mémoire sont testées. Le test de la mémoire consiste essentiellement à déterminer l'espace mémoire physiquement disponible. En cas d'insuffisance, le système prévient l'utilisateur du défaut, et interrompt les opérations d'initialisation. Lorsqu'un composant présente un dysfonctionnement la phase d'initialisation de ce composant est reprise à zéro.

-2)La phase d'installation du système n'intervient que si la phase précédente concernant les ressources matérielles a été correctement effectuée. Cette phase consiste premièrement à charger (à copier) les variables globales initialisées présentes en ROM dans la RAM superviseur, puis sont créés ou sont réservés les espaces pour les différentes tables et files d'attente utilisées par le système, le tableau descripteur de contexte des processus internes (TDCP) résidents, les deux filles d'attente d'exécution, l'emplacement du tableau descripteur d'allocation mémoire (TDAM). Ensuite sont créés deux tampons d'émission et de réception de l'interface série, un tampon de réception des codes de télécommande et différents tableaux utilisés pour des alarmes et pour le gestionnaire d'écran (Tableau de Correspondance Icône Fenêtre Processus: TCIPF).

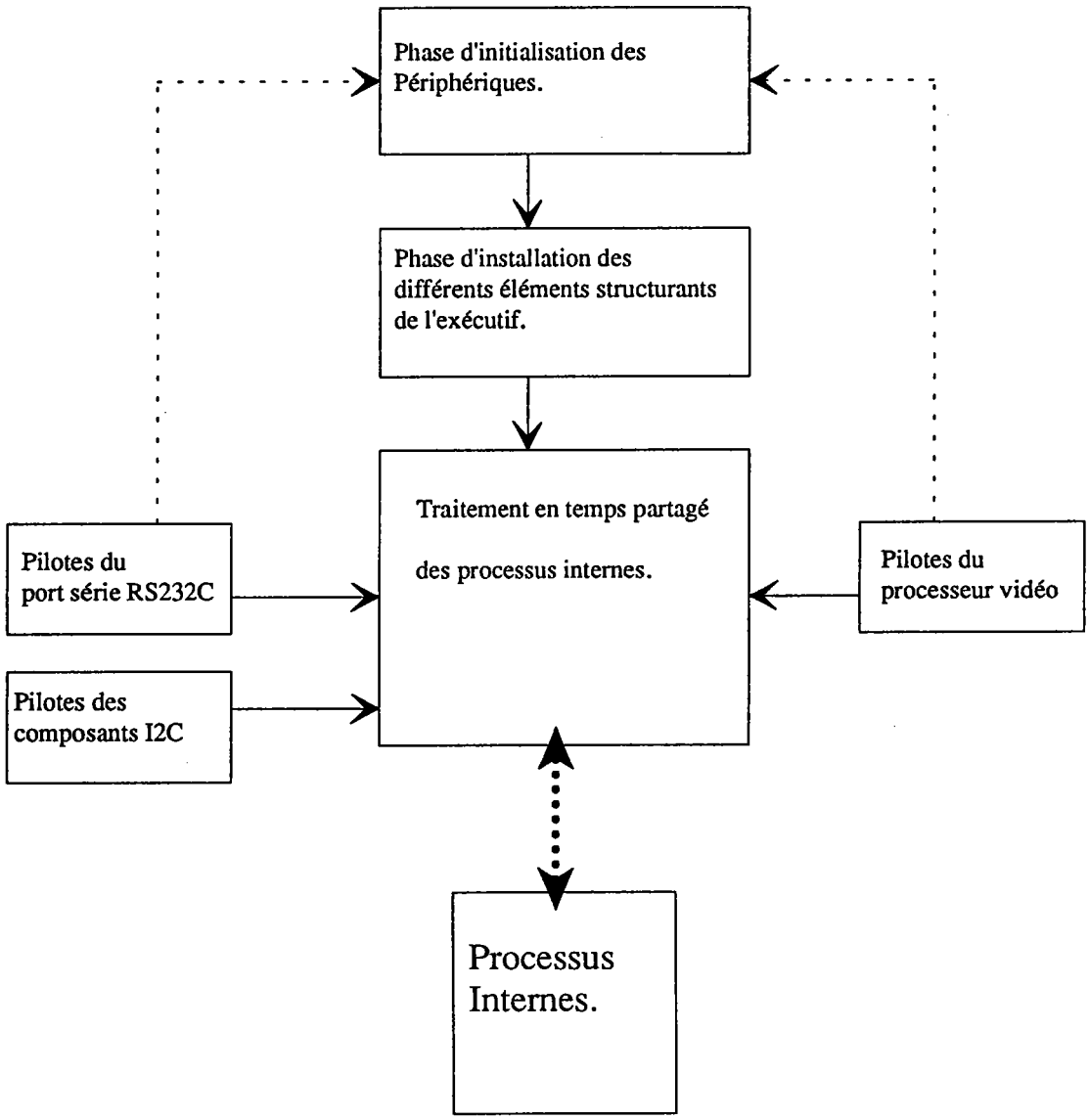


Fig n°III.2: Mise en place du logiciel d'exploitation.

Cette phase se termine par le test d'une variable marquant le nombre d'étapes d'initialisation effectuées. A chaque étape cette variable est incrémentée, et est utilisée par le système pour lancer une réinitialisation graduée matérielle ou logicielle du système. Rappelons qu'à la suite de la détection et la localisation d'un défaut, le système doit essayer de le corriger sans faire appel à une réinitialisation complète de l'équipement (matérielle et logicielle). A la fin de cette phase tous les instruments nécessaires sont mis en place pour l'exécution des différentes applications.

-3)<u>La phase de traitement en temps partagé des applications.</u> Les deux étapes décrites précédemment se sont déroulées de façon linéaire selon un ordre déterminé et en dehors de

toutes contraintes de temps. Cette phase est la phase permanente de choix des processus internes et de leur exécution de façon pseudo-parallèle. Différents mécanismes faisant intervenir des interruptions, exceptions, déroutements entrent en jeu concurremment pour assurer la gestion temporelle d'une part, et la gestion des ressources matérielles d'autre part, et de ce fait cassent tout cheminement linéaire à moyen terme.

La gestion pseudo parallèle des tâches internes utilise une fonction d'ordonnancement, alliée à une horloge de synchronisation système et à sa routine de traitement d'interruptions. La fonction d'ordonnancement lit tour à tour chacune des files d'attente et exécute par tranche chacune des tâches qui y sont présentes, en fonction d'informations caractérisant les processus et disponibles dans les deux descripteurs système. L'horloge intervient (par le signal d'interruption horloge) pour marquer la fin d'une période de temps allouée au processus pour son exécution. La routine d'interruption effectue la sauvegarde du contexte et de l'environnement du processus et modifie le cheminement linéaire des opérations (poursuite d'exécution du processus) par un retour au début de la fonction d'ordonnancement et le lancement d'une autre application.

Ce cheminement déjà perturbé l'est encore plus selon les conditions de survenance de l'interruption de l'horloge système qui entraîne une réponse différée du système au signal d'horloge, et par les multiples sources d'interruption externes liées aux périphériques (réseau collectif domotique, télécommande...) qui requièrent des traitements et réponses adaptés en de courts laps de temps. Ces traitements étant effectués on revient à la scrutation des files d'attente et à l'ordonnancement des tâches, opérations qui constituent la tâche de fond du système. Cette tâche de fond permet au processeur d'être en fonctionnement permanent, seule façon de pouvoir prendre en compte les interruptions matérielles incidentes.

En bref...

Cette description sommaire du système permet déjà d'une part d'entrevoir les solutions apportées pour répondre aux conditions particulières de fonctionnement de l'équipement et enfin de cerner l'état d'esprit qui a présidé à sa conception.

IV L'EXECUTIF MULTITACHE DU TERMINAL

IV.1 Rôle de l'exécutif multitâche

Le rôle de l'exécutif est de gérer l'exécution de plusieurs tâches de façon pseudo-parallèle avec les conflits d'accès engendrés par ce mode de gestion pour l'utilisation de certaines ressources matérielles et logicielles par plusieurs processus.

Une exécution pseudo-parallèle des processus consiste à exécuter séquentiellement, partiellement et de façon cyclique des processus actifs de façon suffisamment rapide pour donner l'impression de la simultanéité des traitements.

L'exécution des programmes et processus est orchestrée par un programme spécialisé ("scheduler", "job-controler") effectuant le choix et le contrôle d'exécution compte tenu de contraintes:

-ne pas permettre à un processus de monopoliser le processeur.
-garantir à chaque tâche un traitement adapté.
-garantir une exécution correcte et prévisible des processus.
-respecter des temps de réponse définis par rapport aux événements déterministes.
-gérer les conflits d'exécution.

L'utilisation de termes anglais francisés, dans la plupart des ouvrages est source de confusions et d'incompréhensions. Ainsi "scheduler" se traduit par le terme de programmateur de travaux selon un ordre défini, et prend le nom d'ordonnanceur [réf DOR]. Le terme correct correspondant serait celui d'ordonnateur. A ce néologisme s'ajoute une différece d'appréciation sur la signification opérationnelle du scheduler. En général il réalise toutes les opérations d'allocations du matériel, de choix et de contrôle des processus [ref]. Pour certains auteurs, l'ordonnanceur est complétée par une autre unité de traitement; le "dispatcher" ou répartiteur, qui effectue l'élection des processus et les changements de contextes [ref DOR] (fig n°IV.1).

Dans la suite de cet exposé, la notion d'ordonnanceur ou de fonction d'ordonnancement sera utilisée pour qualifier l'ensemble logiciel de gestion des taches (scrutation des files d'attentes, mise à jour des variables des contexte de processus, lancement d'exécution, chargement des files d'attente, pénalisation des processus longs, retour d'exécution d'une tâche, gestion des sémaphores, gestion de l'horloge système).

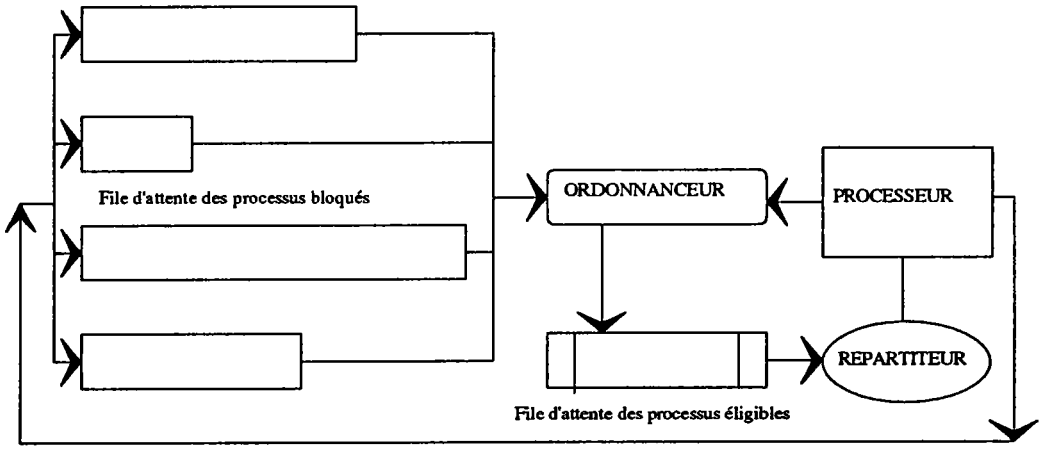


Fig n° IV.1: L'ensemble répartiteur-ordonnanceur

VI.2 Applications, tâches et processus

Programme d'application

Une application est un ensemble d'opérations, de traitements cohérents (processus, tâches), mettant en oeuvre différentes ressources matérielles, dans le but de proposer un service spécifique à un utilisateur défini.

La notion de temps réel s'applique aussi bien pour les systèmes d'exploitation que pour les applications. Pour assurer la satisfaction des contraintes temporelles il est préférable que l'ensemble logiciel obéisse à la notion de temps réel.

Une application temps réel présente un certain nombre de caractéristiques [ref:DOR]

-chaque application est divisée en une ou plusieurs tâches.

-l'exécution d'une tâche s'arrête soit parce qu'elle est terminée, soit parce qu'elle est interrompue par une autre tâche.

-à chaque tâche est affectée une priorité, permettant d'arbitrer l'exécution séquentielle des tâches.

-en cas de conflit d'exécution un deuxième mécanisme (premier arrivé ; premier sorti, ou processus le plus court) permet de le résoudre.

Notion de processus ou tâche

Une tâche est une suite d'instructions, constituant une entité logique et concernant un nombre limité de ressources matérielles ou logicielles. Une application met en oeuvre différents processus.

Selon les auteurs [réf:GRI] les notions de tâches et processus sont différentes et s'imbriquent dans un sens ou dans un autre. Par exemple une tâche est constituée de différents processus. Ce schéma de description implique un triple niveau de structuration des entités logiques programmées [processus-tâche-application] (fig n°IV.2).

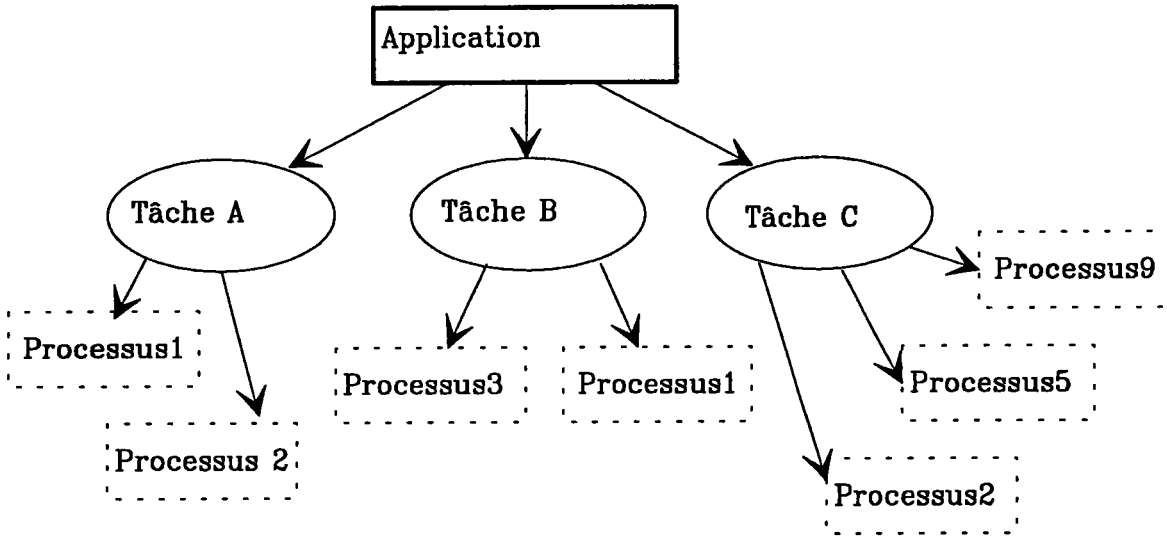


Figure n°IV.2 :Structure hiérarchique de constitution d'une application en trois niveaux

Cependant on remarque que la notion de processus recouvre exactement celle de tâche avec une connotation d'application dans les domaines industriels (fig n°IV.3). Cette notion de processus est plutôt utilisée pour des systèmes tels que OS-9. Ainsi on parle de processus de contrôle et de régulation d'une chaîne de fabrication en film mince, d'un processus de recueils échantillonnés de mesures.

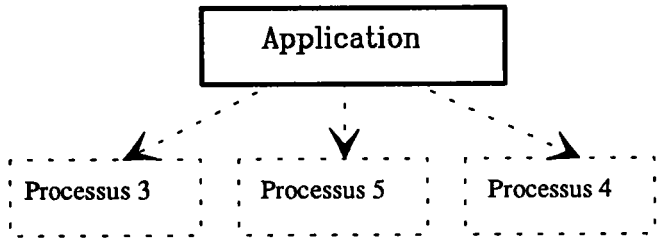


Figure n°IV.3 :Structure hiérarchique de constitution d'une application en deux niveaux

Dans l'ensemble du document nous considérerons les deux termes tâche et processus comme synonymes.

Une entité indépendante

Un processus est en principe un programme d'application de portée limitée écrit en un langage quelconque et indépendant du système d'exploitation . Mais cette indépendance est toute relative puisque dépendant du support matériel (microprocesseur) et du système d'exploitation qui en gère les ressources (fig n°IV.4).

Cette indépendance réside au niveau du programme dans le fait qu'il n'y a pas de liens explicites entre l'exécutif et le processus tel qu'un appel de sous-programme, ou d'appel de fonction. Un processus, pour être exécuté doit être reconnu au préalable par l'exécutif et référencé en tant que tel, mais son existence reconnue n'implique pas automatiquement son exécution. Le processus peut être ou non exécuté, immédiatement ou en différé, une ou plusieurs fois.

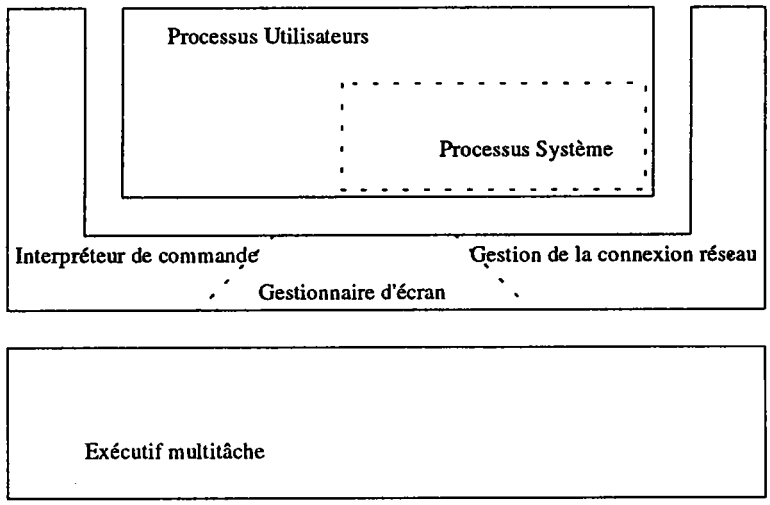


Fig n°IV. 4: Schéma général d'imbrication
des différentes couches du logiciel d'exploitation.

La dépendance entre processus et système d'exploitation réside dans les systèmes de gestion des périphériques (pilotes de périphériques) et les bibliothèques de fonctions généralistes proposées par l'exécutif au processus. Ces fonctions intégrées sont incluses pour faciliter le travail du développeur et pour lier l'application à un système. Un processus nécessitant une gestion graphique intégrant des fonctions de multifenêtrage ne peut fonctionner correctement avec un exécutif ne proposant qu'une gestion d'écran en mode texte. Il y a donc dépendance entre le processus et la couche inférieure constituant l'exécutif dont certaines fonctions sont utilisées par le processus. C'est pour cette raison que les logiciels d'application devant être exécutés sur un système donné sont de préférence développés dans ce même environnement (Windows, Pctools... fonctionnant sous MS-DOS).

Les échanges de données entre processus

La nature de l'exécution d'un processus est dictée par l'éventuelle structure d'échange d'informations entre deux processus. En exécution séquentielle, les processus s'exécutent les uns à la suite des autres et se transmettent d'éventuelles informations selon un ordre de préséance. Ainsi un processus ne peut s'exécuter qu'à la condition que le précédent soit terminé et ait déposé les données en mémoire dans une structure de transfert définie. Une exécution des processus en parallèle nécessite un mécanisme d'échange d'informations bidirectionnel avec partage des données en écriture, lecture. Dans ce système un processus peut s'arrêter et attendre les informations fournies par un autre processus ayant débuté à n'importe quel moment. Ce mécanisme est plus souple mais nécessite une structure d'échange très formalisée. La description de mécanismes (modèle producteur-consommateur) sera faite en détail dans un chapitre suivant (cf: V.).

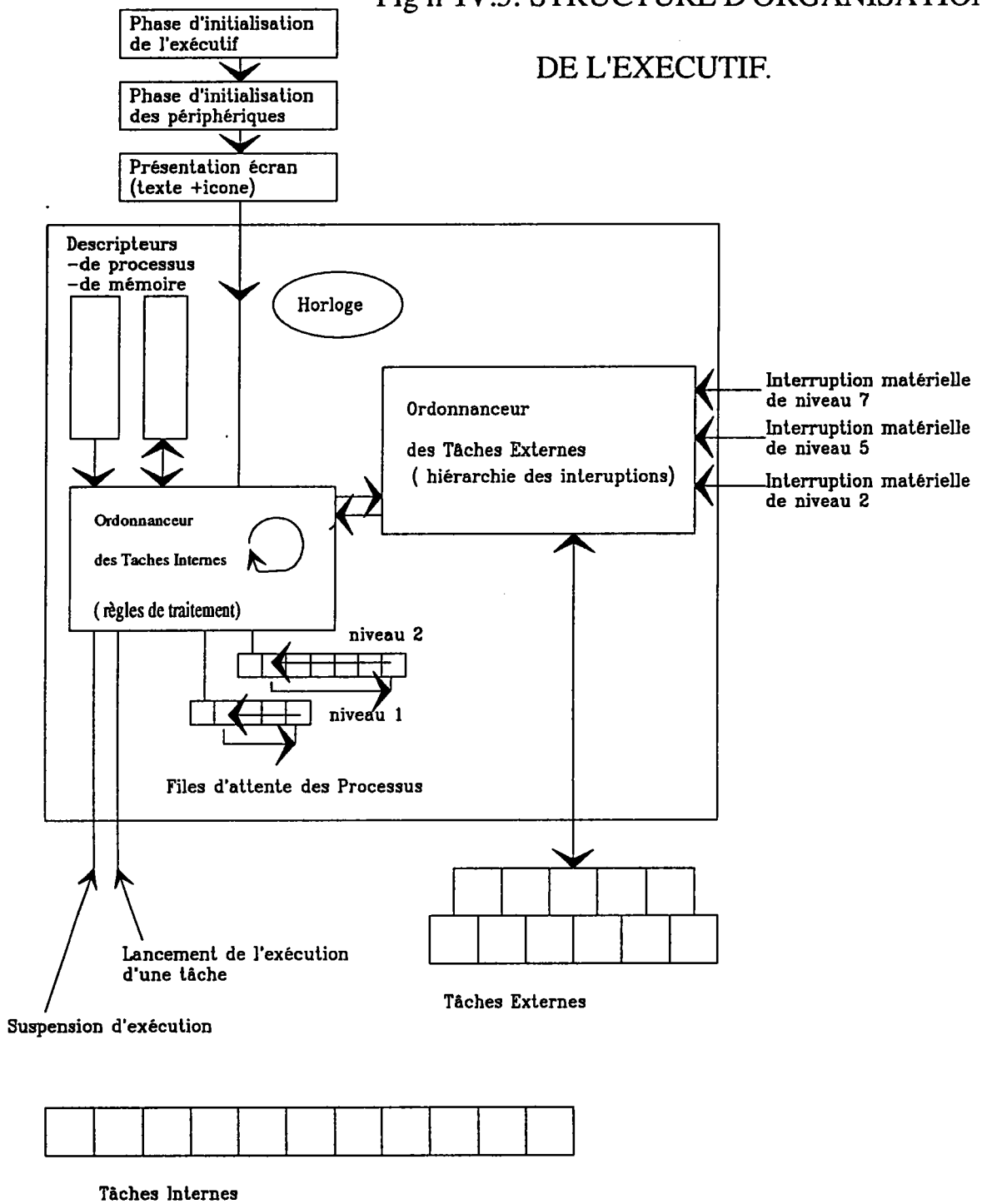
Différents types de processus

Ce terminal, tout en disposant d'un système d'exploitation et de logiciels d'application, ne pourra constituer une plate forme de développement permettant d'écrire des applications (compilateur, interpréteur de langage). Les applications sont soit intégrées en ROM, soit téléchargées, l'allocation mémoire étant alors différente dans son mécanisme et dans la taille mémoire allouée. Le processus sera qualifié de processus "résident" (en ROM) ou de processus " importé" (téléchargé).

Une deuxième différenciation des processus porte sur l'application; un processus "système" est un programme d'application à l'usage du système lui-même. C'est le cas des processus de gestion de la mémoire: processus de compactage, processus d'extension de la mémoire. Ces processus opèrent sur la mémoire système protégée et ont recours à des fonctions de niveau superviseur. Les autres processus sont des processus dits "utilisateurs" ayant de façon générale à interagir avec l'utilisateur. Ces processus utilisateurs constituent la partie applicative du terminal.

Tous les processus ne sont pas destinés à recueillir et à stocker des données. Ainsi pour une raison d'économie d'espace mémoire on fait la distinction entre les processus nécessitant une zone mémoire de stockage des données (processus "enregistrant") et les processus n'effectuant pas ce stockage (processus "non-enregistrant").

Fig n°IV.5: STRUCTURE D'ORGANISATION DE L'EXECUTIF.



La quatrième qualification est plus une description formelle du processus. On parlera dans la suite de processus internes et de processus externes.

Un processus interne est un processus ayant une référence dans les différentes tables du système et représenté par un numéro de processus; il peut être résident ou importé, nécessiter une zone de stockage des données ou pas, être un processus système ou utilisateur.

Son exécution est décidée par l'ordonnanceur des tâches internes à la requête de l'utilisateur ou même d'un autre processus (fig n°IV.5).

Un processus externe est lié à l'exécutif et au matériel, car son exécution est liée directement au système de gestion des interruptions matérielles hiérarchisées (exécution automatique lancée par une interruption)(fig n°IV.5). Ces processus sont des processus courts, n'impliquant pas de longs traitements, et chargés de la réception et du traitement des données en provenance des différentes interfaces. Ils ont la capacité de lancer l'exécution des processus internes (autotest...). Cette distinction entre processus internes et externes permet de traiter différemment les processus "temps réel" prioritaires en exécution (processus externes) et les processus exécutés en temps partagé de moindre priorité temporelle (processus internes) (Fig n°IV.6). Ce mécanisme est différent de celui de système d'exploitation tel que celui-d'OS-9 pour lequel tous les processus doivent êtres lancés par l'ordonnanceur, y compris ceux de traitement des interruptions (Fig n°IV.7).

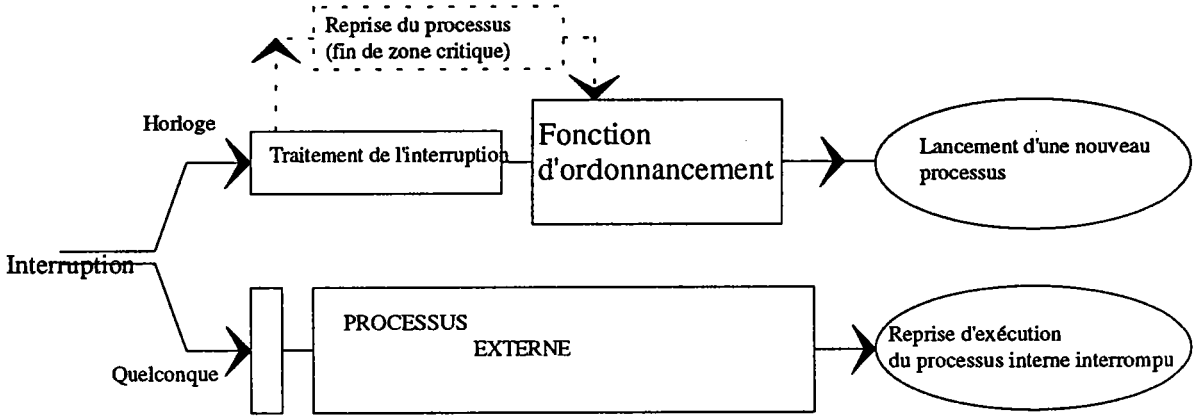


Fig n°IV.6 :Structure d'intervention de la fonction d'ordonnancement réalisée par l'exécutif de l'équipement.

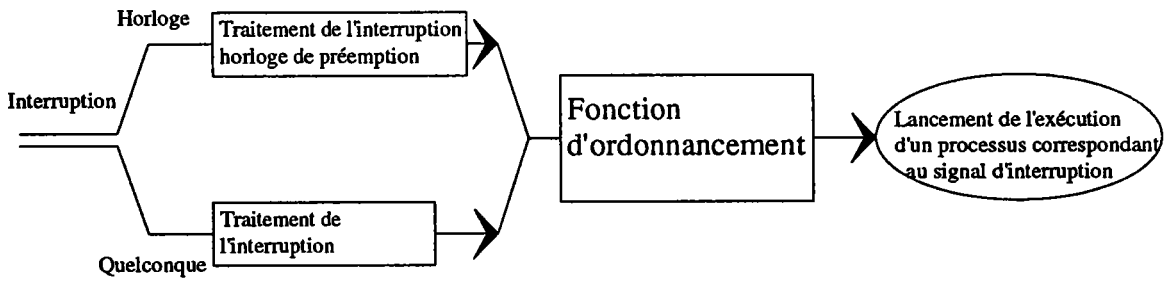


Fig n°IV.7 :Structure de la fonction d'ordonnancement couramment utilisée par les systèmes d'exploitation.

Un processus est donc décrit par quatre qualificateurs ayant une influence sur l'allocation mémoire qui lui est faite. Ainsi pour un processus interne donné on obtient une allocation mémoire différenciée:

-processus résident enregistrant(système ou utilisateur): données+pile
-processus résident non-enregistrant(système ou utilisateur): pile
-processus importé enregistrant(système ou utilisateur): code+données+pile
-processus importé non-enregistrant(système ou utilisateur): code+pile

Cet équipement est un terminal destiné à mettre à disposition de l'utilisateur différents services. Ainsi j'ai choisi de concevoir l'exécutif à partir de la notion de processus, pour ne pas limiter la portée de cet outil, et pour maintenir des possibilités d'ouverture à tous types d'applications et de services. Cette différenciation faite au niveau des processus (nature, type) est dictée par le choix des mécanismes de gestion de la mémoire centrale et ne trouve pas d'équivalence dans les systèmes d'exploitation ou exécutifs généralistes.

IV.3: Représentation d'un processus dans le système

Un élément descripteur de contexte d'exécution d'un processus

Tout processus pour être exécuté et se voir allouer une aire mémoire d'exécution et de stockage, doit être référencé dans un tableau descripteur de contexte de processus par son élément descripteur.

L'ensemble des éléments descripteurs est habituellement représenté par une liste et utilise le double chaînage de ces éléments (fig n° IV.8) pour faciliter les opérations de manipulation d'un processus donné (recherche, mise à jour). Ainsi un élément descripteur est relié par un premier pointeur au descripteur précédent, et par un deuxième pointeur à l'élément suivant.
Cette représentation induit d'une part une liste des processus identifiés, et d'autre par une ou plusieurs files d'attente utilisé par le "scheduler".

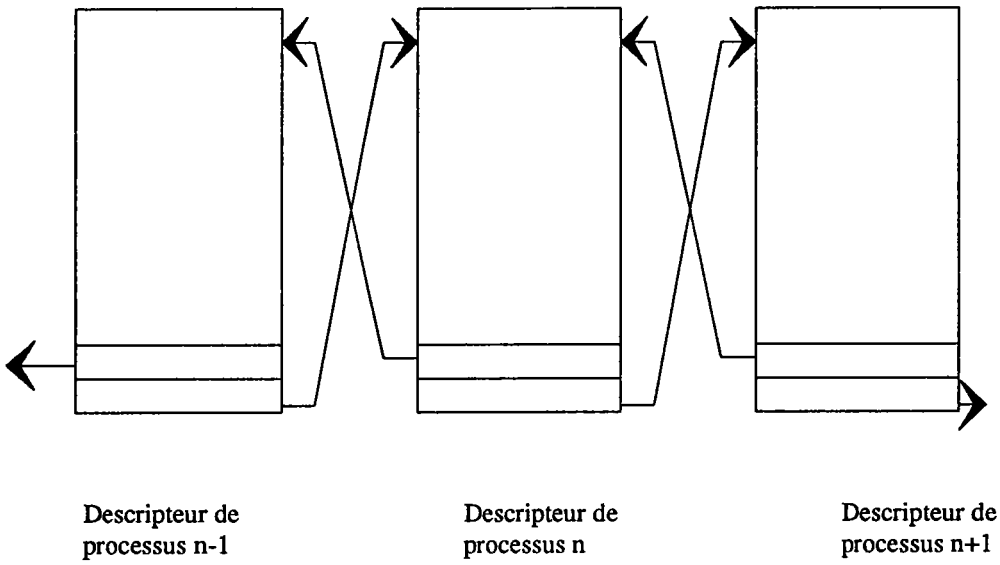


Fig n°IV.8 :Structure chaînée des descripteurs de processus

Un élément descripteur renferme généralement un certain nombre d'informations telles que le nom, la priorité, l'état, le type et le contexte du processus. Le contexte de processus renferme les valeurs courantes du compteur ordinal, l'état du processus, le contenu des registres généraux du processeur, les droits d'accès aux ressources logicielles et matérielles, les pointeurs et index d'accès aux zones de travail virtuelles en mémoire, c'est à dire un ensemble de variables dynamiques absolument nécessaires à l'exécution du processus.

Le descripteur de contexte de processus ne doit êre confondu avec le descripteur de contexte du système contenant lui des informations vitales de l'exécutif (valeurs de pile système, registres internes utilisés pour pointer vers les différents tableaux,...).

Si le principe du descripteur de processus est retenu pour l'exécutif, sa forme, son contenu, son fonctionnement sont différents car l'aspect de la gestion mémoire n'y est pas intégré. D'une part j'ai créé deux éléments descripteurs contenant l'un les variables statiques et l'autre les variables dynamiques du processus, toutes valeurs utilisables par les deux mécanismes de gestion du noyau, le gestionnaire en temps partagé et le gestionnaire de mémoire. D'autre part le tableau descripteur de contexte de processus est permanent par les références des processus résidents qu'il renferme. De plus il est extensible par l'apport de processus importés.

Cet élément qui contient les informations générales et statiques du processus est organisé en quatre mots de 32 bits (fig n°IV.9):

1er mot long: informations générales du processus (figure n°IV.10)

-numéro de processus: fixe

-le lieu d'implantation du code programme du processus: ROM (processus résident), RAM (processus importé).

-la nature et le type de processus

-le nombre d'aires mémoire allouées (enregistrements) et liées au processus.

-un indicateur de préemption du processus (cf n°).

2éme mot long: adresse de base de la zone code exécutable du programme

3éme mot long: adresse de base de la zone des données non initialisées pour les processus enregistrants.

4éme mot long: adresse de base de la zone mémoire affectée à la pile locale du processus.

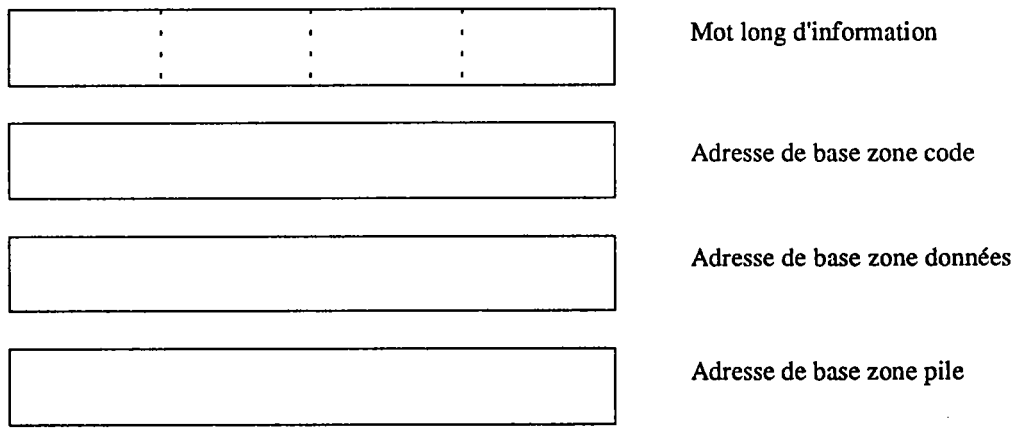


Fig n°IV.9: Elément descripteur de contexte processus (variables statiques)

Informations descriptives du processus (paramètres statiques)

L'ensemble des informations sur le processus et ses caractéristiques statiques sont incluses dans le 1er mot long de son élément descripteur. Sa structure est représentée par la

figure n°IV.10. Ces informations sont utilisées dans le cadre de la gestion des processus et en partie dans le cadre de la gestion mémoire.

Le numéro de processus codé sur 8 bits (256 processus sont référençables) constitue le nom du processus et est utilisé pour la recherche d'un processus et d'un enregistrement , pour la restauration et la sauvegarde des contextes des processus, pour le chargement en file d'attente.

Le niveau de priorité est un indicateur de la priorité de traitement du processus et donc de la file d'attente à charger. De façon normale la priorité est fixée lors de l'élaboration de l'élément descripteur de contexte du processus.

L'implantation en mémoire du code programme du processus nous informe de la nature du processus résident ou importé. Cette information est encore utilisée au niveau de la gestion mémoire.

Le nombre d'enregistrements liés à un processus correspond exactement au nombre de zones mémoire allouées pour l'exécution du processus et le stockage de données. Une zone mémoire d'exécution est constituée soit d'un espace réservé à la pile (processus non enregistrant), soit d'un double espace mémoire alloué aux données non initialisées, et à la pile locale du processus (processus enregistrant).

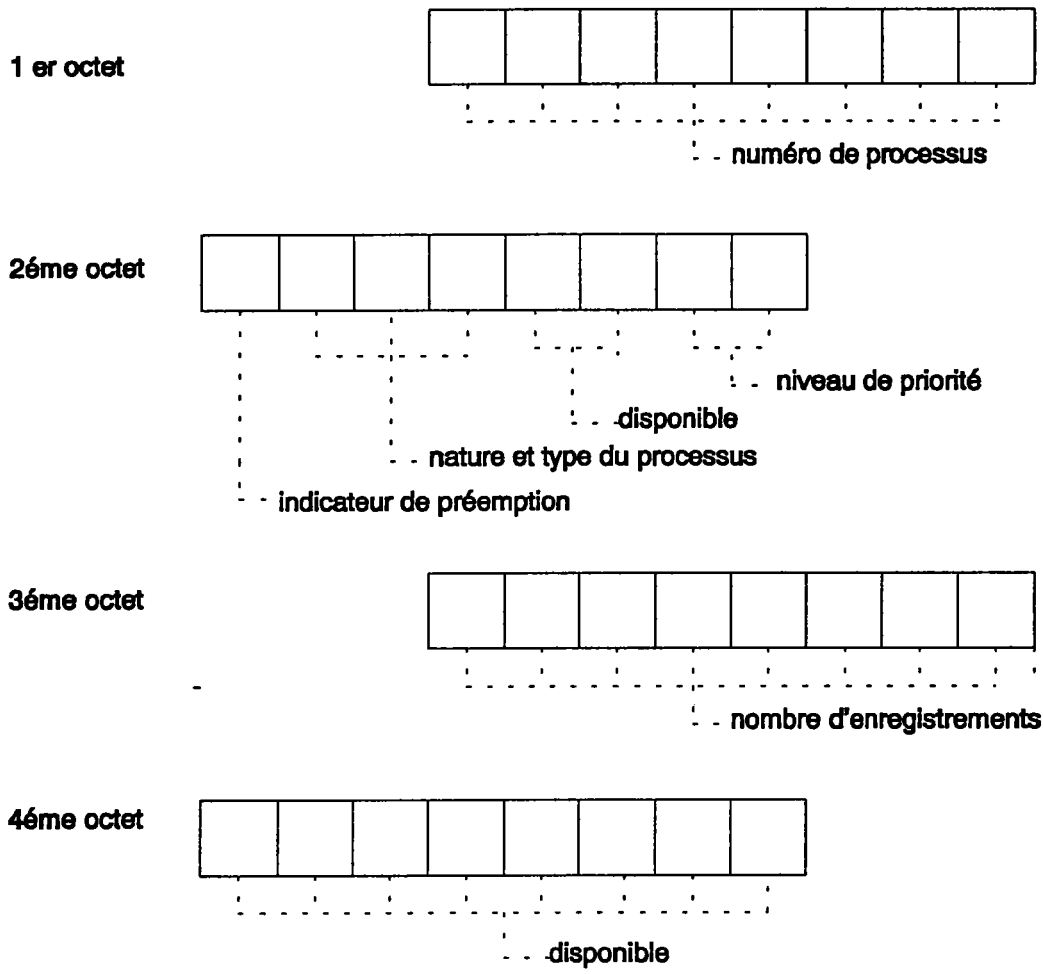


Fig n°IV.10: Détail du premier mot long (mot d'information) d'un élément descripteur de contexte processus.

Validation du mécanisme de préemption/non préemption des processus (cf n°).

Tous les processus, en principe peuvent être interrompus au cours de leur exécution par l'horloge système pour l'exécution séquentielle d'un autre processus. Il peut être nécessaire pour quelques processus d'interdire l'application de ce mécanisme. C'est le cas pour les processus système de compactage et extension de la mémoire.

Les processus résidents sont référencés en ROM par leur élément descripteur de contexte de processus.
Les processus importés seront chargés en RAM Superviseur à partir du réseau. Les premières informations fournies par le serveur de programmes constitueront l'élément descripteur de l'application téléchargée.

Les différents états d'un processus

Un processus se trouve obligatoirement défini par son état indiquant le niveau d'exécution atteint et permettant ainsi aux automatismes du système de savoir que faire de ce processus. La figure n°:IV.11 met en évidence le schéma d'évolution d'état du processus en fonction des événements suceptibles d'intervenir.

D'autres états du processus pourraient êtres définis marquant ainsi la puissance de l'exécutif ou système d'exploitation, par la multiplication des événements intervenant au cours de la vie d'un processus (inexistant, hors service, éligible, élu, en attente de ressource, en attente de délai, en attente d'événement).

Etant un paramètre dynamique, l'état du processus ne se trouve pas consigné dans l'élément descripteur de contexte processus, mais dans le ou les éléments descripteurs d'allocation mémoire de ce processus.

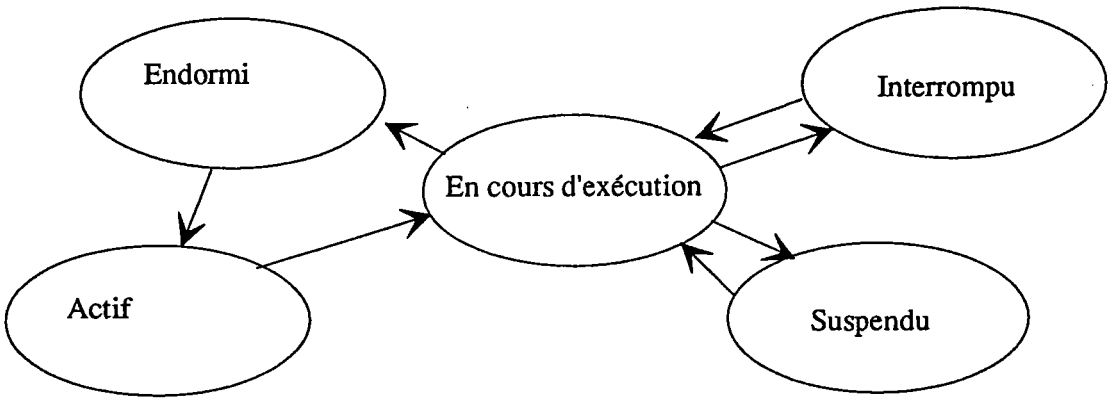


Fig n°IV.11: Les différents états d'un processus.

VI.4 Synchronisation des processus

Partage des périphériques et sections critiques

Ce mécanisme particulier est implanté pour des systèmes à fonctionnement parallèle ou pseudo-parallèle, ou même multiutilisateur. Ce mécanisme de synchronisation intervient pour gérer l'accès par plusieurs processus à un même périphérique; accès qui peut se révéler catastrophique s'il n'est pas mis en place (blocage du système).

Considérons le cas concret de deux processus recourant à une procédure d'affichage de leurs données respectives. Le premier processus exécute une routine d'affichage d'un caractère à l'écran et pour cela intervient au niveau de plusieurs registres du processeur semi-graphique. Ces registres programmés gèrent les couleurs de fond de caractère et de caractère, le type et la position à l'écran du caractère... A la suite d'un signal d' interruption de l'horloge système ce processus est interrompu dans son exécution, voit son contexte de processus sauvegardé et enfin une nouvelle tâche est lancée en exécution. Cette nouvelle tâche utilise les mêmes fonctions d'affichage à l'écran. La tâche précédente avait réalisé le positionnement du curseur à l'écran, mais pas complètement l'opération d'affichage du caractère. Deux résultats peuvent être obtenus;

- soit le nouveau processus effectue sans problème une nouvelle séquence complète d'affichage de caractères et dans ce cas, lors de la reprise d'exécution du premier processus les paramètres de programmation initiaux sont modifiés et il y a simplement interférence entre les données et les affichages présentent des défauts de positionnement et de couleurs. L'exécution des processus n'en est pas perturbée du point de vue commutation des tâches.

- soit lors du retour d'exécution au premier processus, l'enchaînement de programmation des registres est perturbé dans la cohérence des informations programmées, et on aboutit à un blocage du composant qui ne réagit plus à aucune sollicitation. Dans ce cas (le plus grave) il ne reste alors plus qu'à réinitialiser matériellement tous les registres internes du processeur grahique. Tous les travaux d'affichage précédents sont perdus.

Pour éviter ces dysfonctionnements il faut donc rendre indivisible certains traitements sensibles, protéger certaines sections de code exécutables (sections critiques) tout en restant sensible à la survenance d'événements. Le but étant de synchroniser l'accès, l'utilisation d'une ou plusieurs interfaces par rapport à différents processus requérant les services de cette interface. Par rapport à ces interfaces le traitement des processus ne peut plus être parallèle mais séquentiel et complet.

Les solutions logicielles au problème de partage des ressources matérielles

Trois solutions existent pour résoudre les problèmes de synchronisation d'accès de plusieurs processus à une même ressource:

-les masques d'interruption: on interdit la survenance d'événements de priorités inférieures à celle programmée. C'est le cas du signal d'horloge système qui régule l'exécution des processus. Un processus disposant d'un petit nombre de grandes sections critiques, peut n'être jamais interrompu par l'horloge système. L'aspect multitâche disparaît alors.

-les sémaphores: ce sont des variables de type particulier qui marquent l'entrée ou la sortie de zone critique d'un processus. Elles s'accompagnent de fonctions de modification de la valeur de ces variables. Il existe deux types de sémaphores, les sémaphores simples ne permettant l'accès à une zone critique qu'à un seul processus, et les sémaphores compteurs donnant accès à une ressource, à un nombre limité de processus. La notion de sémaphore ne peut être envisagée que si le microprocesseur dispose dans son jeu d'instructions d'une fonction indivisible de test et de positionnement de sémaphores.

-les événements: ces objets sont liés au langage de programmation utilisé, et leurs modes d'action sont similaires à ceux des sémaphores. Un signal événement diffère d'un sémaphore par son aspect dynamique de fonctionnement qui le rapproche plus du signal d'interruption. Il se trouve être sur le plan de l'efficacité d'une portée plus limitée que les sémaphores (indivisibilité) et les masques d'interruption. Par contre ils sont particulièrement utilisés dans l'exécution de processus requérant des données à survenance aléatoire, et dans les échanges d'informations entre différents processus. Une variable événement est accompagnée de deux fonctions de positionnement du signal indiquant que la donnée est présente ou absente.

Avant de décrire le fonctionnement précis de synchronisation des accès aux ressources, il nous faut apporter quelques compléments d'information.

Une section critique est un ensemble d'instructions interruptibles mais présentant un certain niveau de sécurité (par les sémaphores) pour assurer l'exécution complète de cette zone critique. A chaque zone critique est affectée au moins un sémaphore.

Une zone critique est encadrée par deux fonctions. Une fonction d'incrémentation du sémaphore (positionnement à 1 pour un sémaphore simple) à l'entrée de la zone critique, et une fonction de décrémentation (positionnement à zéro pour un sémaphore simple) en sortie d'une zone critique (fig n°IV.12).

La fonction d'entrée en zone critique est conçue à partir d'une instruction spécifique du processeur (instruction TAS du 680xx: Test And Set) effectuant en un seul cycle machine (instruction indivisible) la lecture du sémaphore et le test d'état de ce sémaphore. L'indivisibilité de cette instruction fait que la survenance d'une interruption ou exception ne peut être prise en compte que soit avant l'exécution de l'instruction lorsque le processeur n'est pas encore entré en zone critique, soit après exécution de l'instruction à l'intérieur de la zone critique.

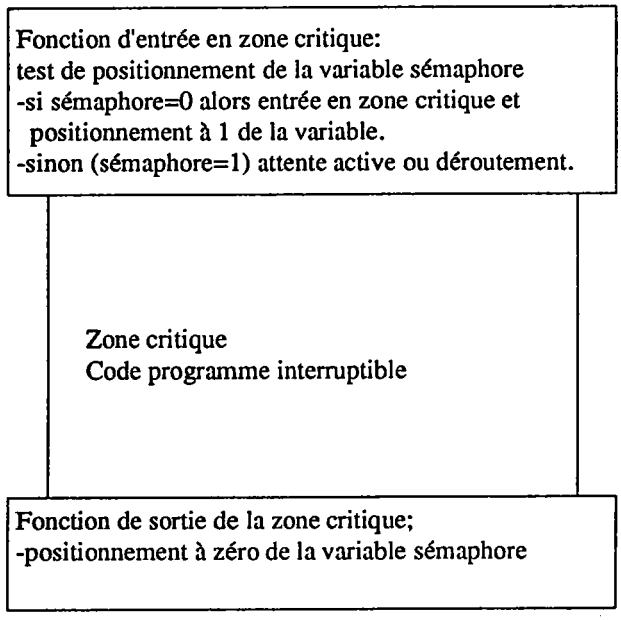


Fig n°IV.12: Environnement d'une section critique.

Toutes les interfaces ne nécessitent pas d'être protégées en accès de cette façon et on peut les classer en deux catégories selon le périphérique:

-les périphériques partageables. La mémoire peut être considérée partageable dans sa globalité puisqu'elle renferme les informations de différents processus. Elle peut être aussi considérée comme non partageable au niveau de l'espace mémoire alloué de façon unique à un processus. Les échanges de données entre processus par tampon seront régis par sémaphore compteur.

-les périphériques non-partageables: nécessitant l'utilisation d'un mécanisme de synchronisation d'accès tels que le système de visualisation graphique, une imprimante à liaison parallèle, un réseau de communication via le port série RS232C, la mémoire centrale.

Ainsi quatre sémaphores simples ont été définis pour les interfaces: sémaphores pour le processeur vidéo (SEM_VDP), pour le port série (SEM_RS232), pour l'interface I2C (SEM_I2C), pour la mémoire centrale (RAM)(SEM_MEM).

A chaque périphérique non partageable est associé un sémaphore et différentes fonctions d'incrémentation ou de décrémentation du sémaphore.

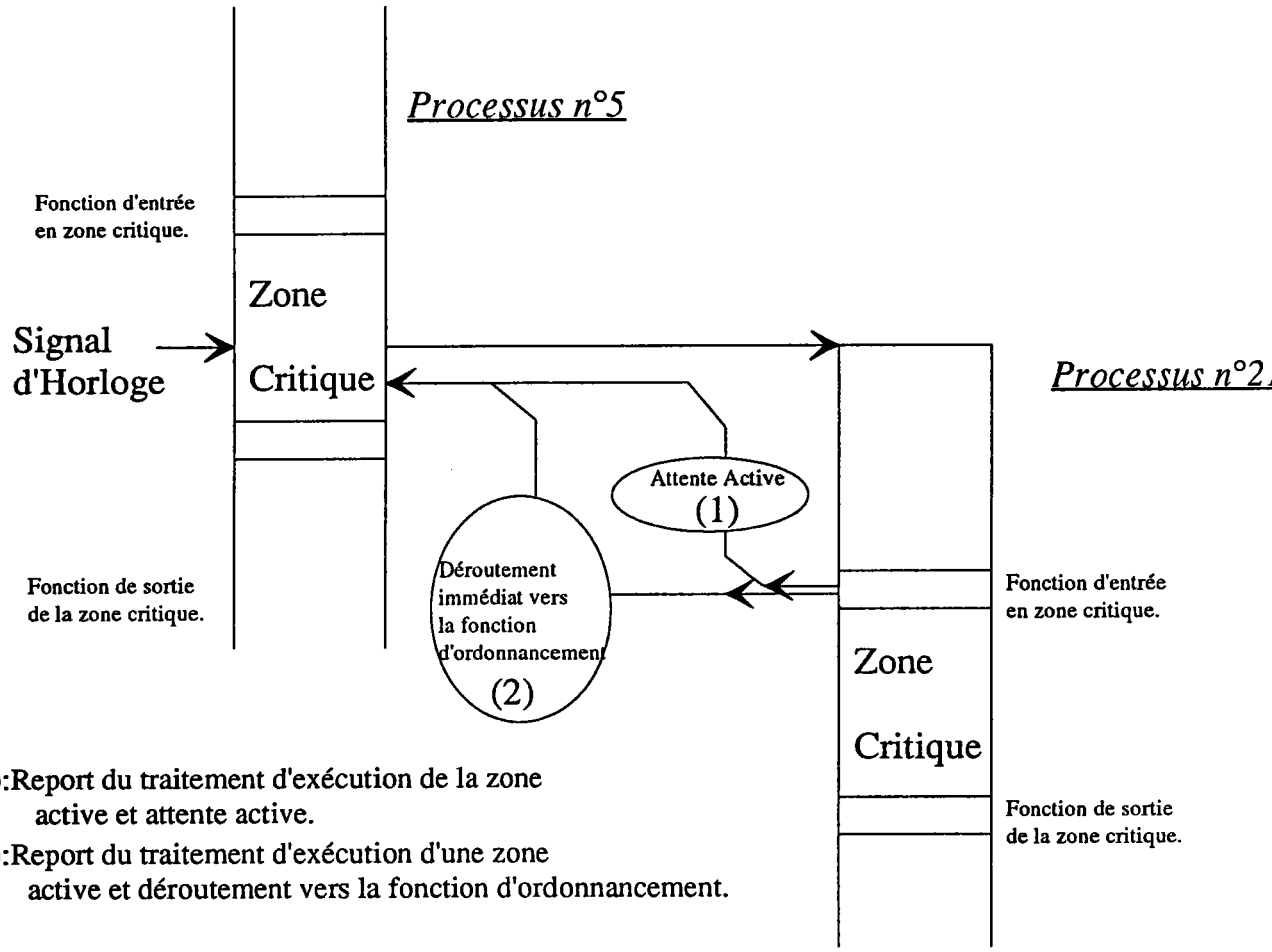
Sections critiques et horloge système

Une section critique est un groupe d'instructions participant à la réalisation d'une opération sur une interface donnée. Lorsqu'un signal d'interruption de l'horloge système

survient, l'exécution du code de section critique est interrompu et le contexte du processus est sauvegardé. Un choix stratégique est à effectuer.

L'exécution du processus interrompu est reportée ultérieurement et le sémaphore correspondant reste positionné (sémaphore simple). Un autre processus est lancé, et s'il y a utilisation du même périphérique ce deuxième processus est arrêté à l'entrée de la section critique. Soit le processus rentre dans un boucle d'attente jusqu'au moment où un signal d'horloge intervient providentiellement pour lancer un autre processus qui pourrait être celui occupant la section critique, soit le processus abandonne immédiatement son exécution et se retrouve chargé en fin de file d'attente. Ces deux solutions (figures n°IV.13) ont comme gros défaut de gaspiller du temps CPU, soit dans la boucle d'attente (processus en attente active), soit par utilisation répétée de la procédure d'ordonnancement avec des processus utilisant ce même périphérique. De toute façon l'exécution des autres tâches est subordonnée à l'exécution complète de la zone critique par le processus exécutant le code de cette zone critique.

Le report du traitement complet de la zone critique n'est visiblement pas une bonne solution. Après survenance de l'interruption d'horloge système, et sauvegarde du contexte du processus, l'état du sémaphore est testé. L'interruption survient-elle à l'intérieur d'une zone critique? Si non, un nouveau processus est lancé par le processus d'ordonnancement. Si oui, on décide de terminer immédiatement l'exécution de la section critique du processus en cours et seulement cette section critique (fig n°IV.14).



(1):Report du traitement d'exécution de la zone active et attente active.

(2):Report du traitement d'exécution d'une zone active et déroutement vers la fonction d'ordonnancement.

Fig n°IV.13: Possibilités de traitements d'une section critique par report du traitement de celle-ci..

La fonction de décrémentation du sémaphore (positionnement à zéro du sémaphore simple) intervient alors en fin de zone critique pour revenir au niveau superviseur et aux opérations de sauvegarde du contexte de processus et d'ordonnancement. Cette fonction dispose de l'information (variable globale) précisant qu'un signal d'horloge est intervenu au milieu du traitement d'une section critique.

Cette solution implique de terminer l'exécution d'une section de code critique avant l'exécution de tout autre processus susceptible ou non d'utiliser cette ressource matérielle et aussi d'exécuter un processus au-delà de son unité de temps. Quel que soit le découpage effectué du code exécutable en sections critiques, cette stratégie s'applique correctement (fig n°IV.15 & IV.16).

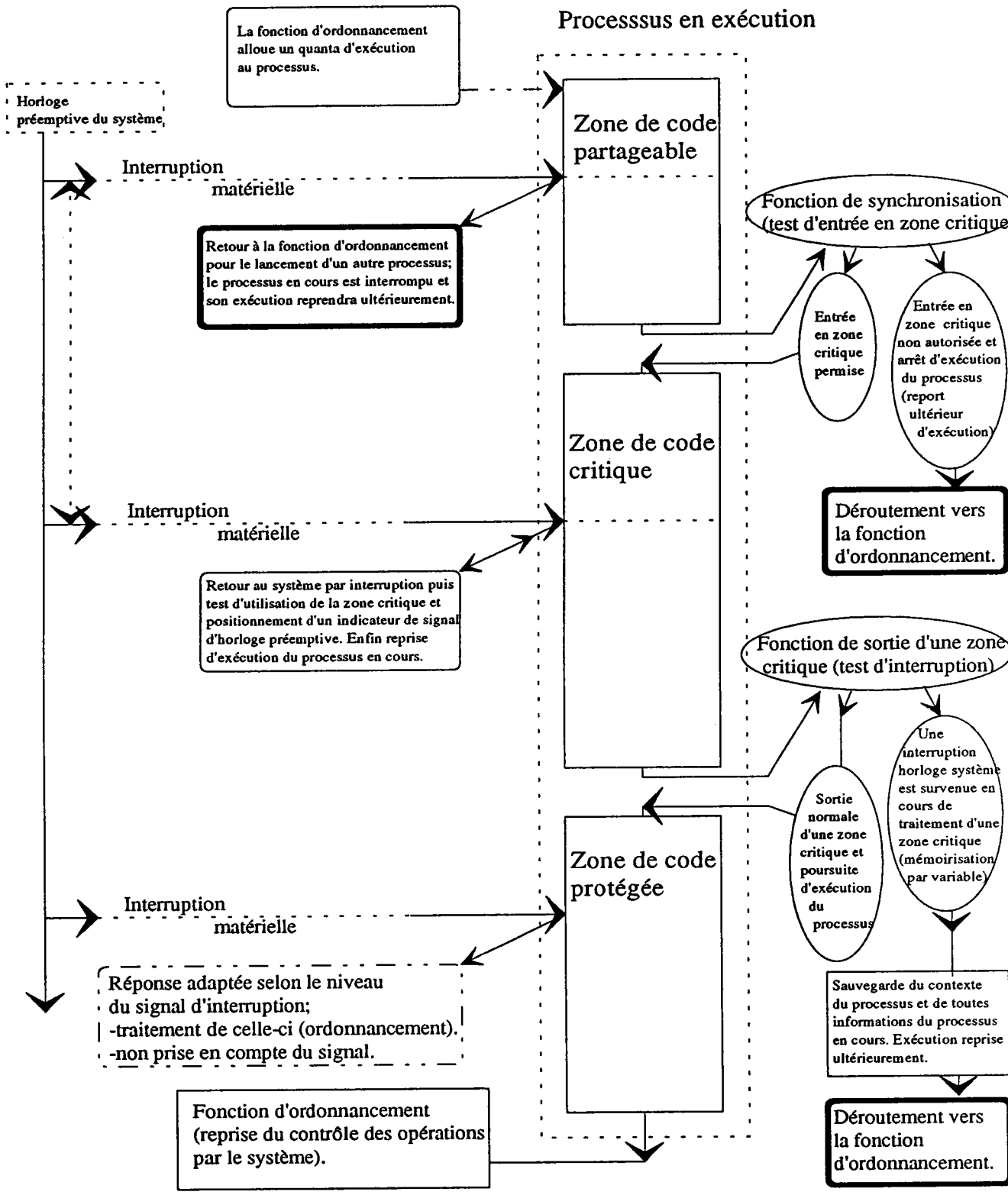


Fig n°IV.14: Synchronisation de l'accès aux ressources matérielles; sémaphores et sections critiques.

Définition des zones critiques

L'utilisation de sémaphores ne présente pas de problèmes particuliers, mais cependant deux éléments importants sont à considérer; la définition et le découpage des zones critiques.

A première vue pour limiter au maximum les problèmes d'interférences entre processus utilisant concurremment les mêmes interfaces, toutes les fonctions intervenant dans une même section de programme d'un périphérique devraient constituer une même zone critique. Ainsi pour effectuer l'affichage d'un nombre composé de trois chiffres, la zone critique comprendrait les fonctions de programmation du jeu de caractères, de couleurs de fond et de caractère, de positionnement et enfin une boucle d'affichage des trois caractères. La section de code exécutable est très longue, ce qui est un handicap pour l'aspect multitâche. Plusieurs signaux d'horloge système peuvent intervenir sans pouvoir arrêter son exécution, et de plus plusieurs sections critiques d'un même processus peuvent s'enchaîner. Enfin les fonctions d'incrémentation et de decrémentation du sémaphore sont présentes dans le programme d'application lui-même, ce qui est une cause d'oublis d'utilisation des sémaphores pour des développements futurs.

L'implantation des fonctions de gestion des sémaphores est effectuée à l'intérieur même des fonctions élémentaires (fonctions pilotes de bas niveau) ce qui rend transparent l'utilisation des sémaphores au programmeur d'applications, et offre une plus grande possibilité d'intervention de l'horloge système sur les tâches. Pour certaines fonctions complémentaires et juxtaposées au sein du programme, une seule section critique pourrait être constituée. Seuls des essais pratiques peuvent définir le positionnement optimum des sémaphores.

Fig n°IV.15: Schéma d'interactions entre les trois couches constituant le logiciel d'exploitation lors de la survenance d'une interruption générée par l'horloge système (1ére partie).

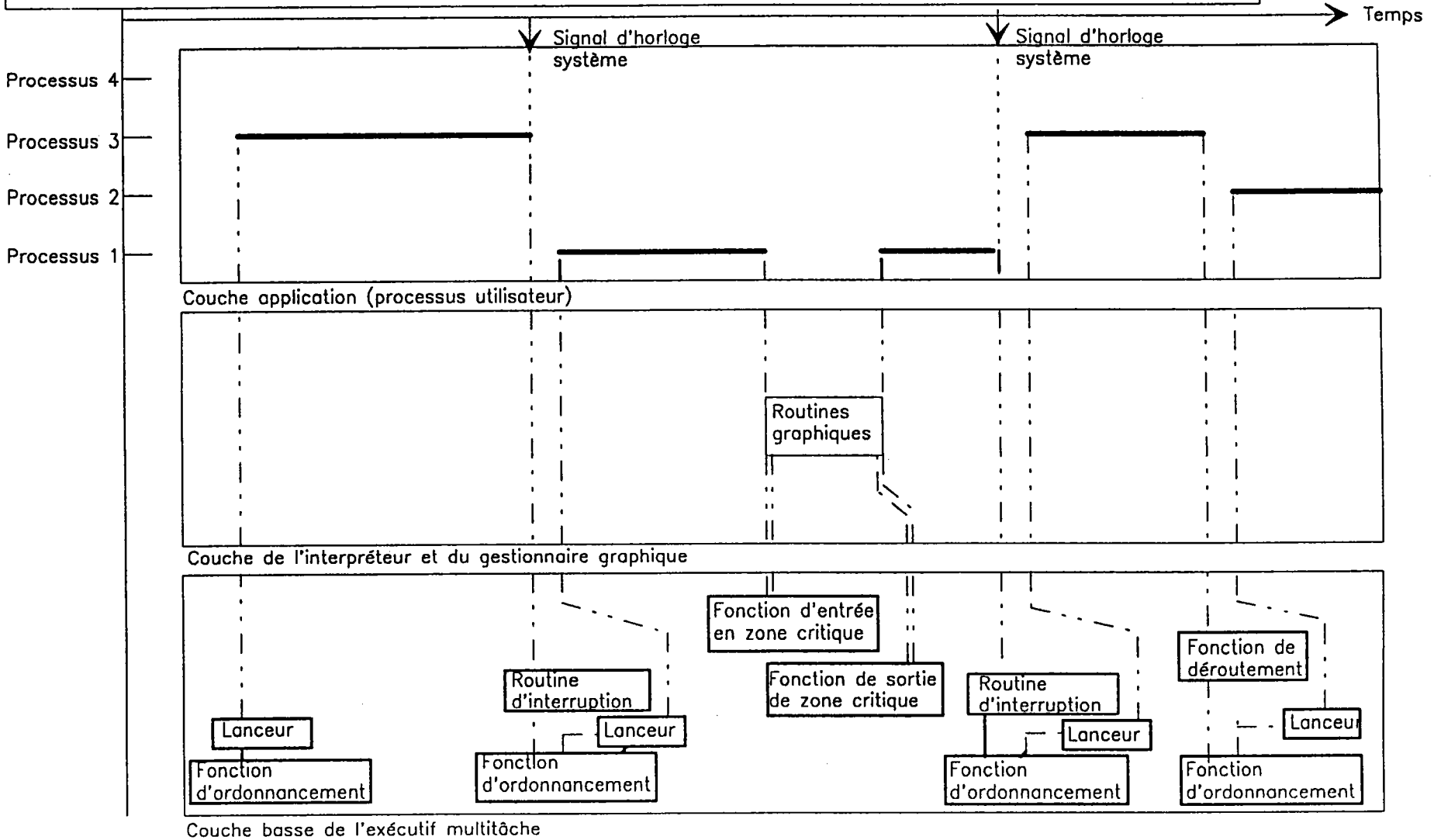
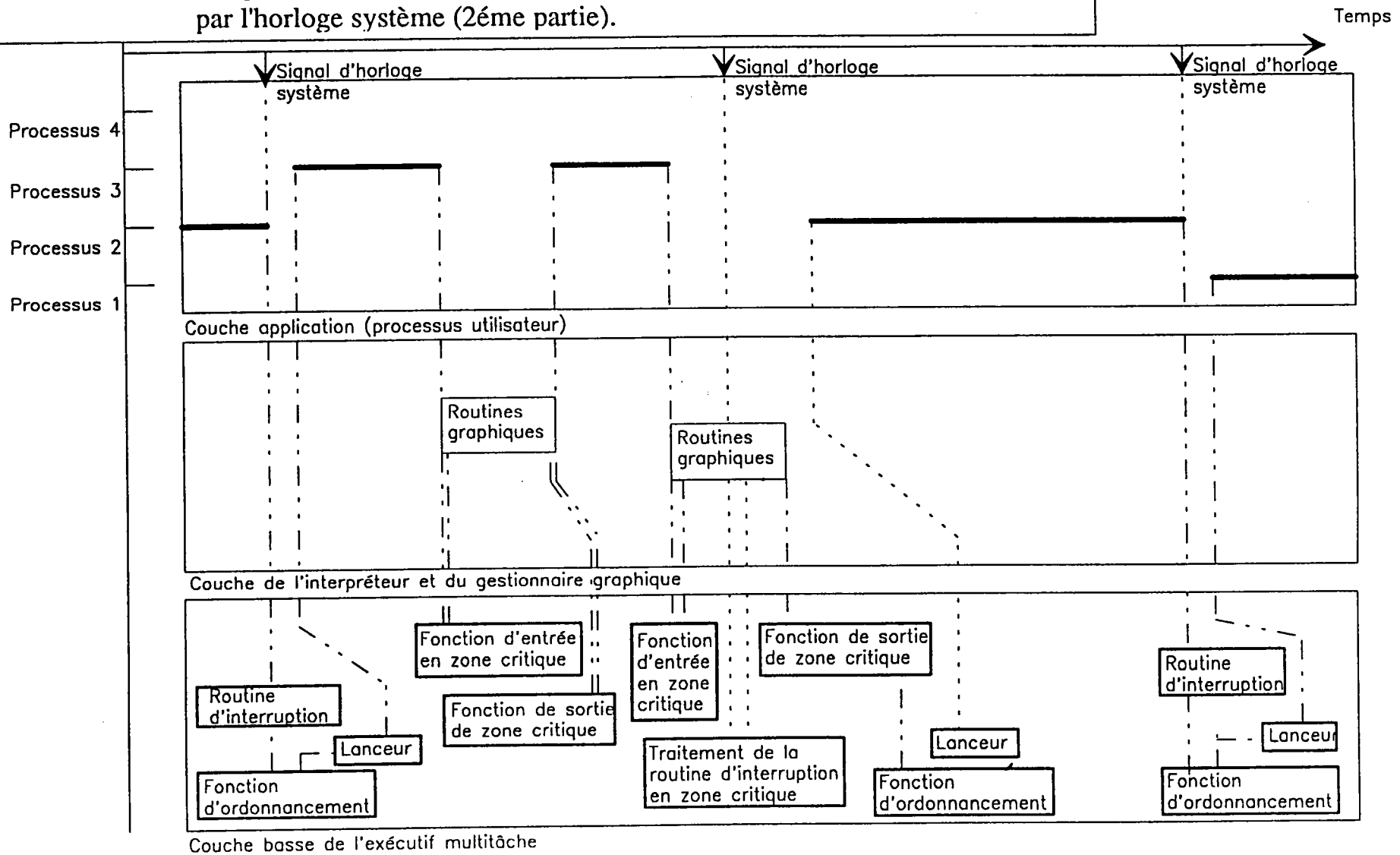


Fig n°IV.16: Schéma d'interaction entre les trois couches constituant le logiciel d'exploitation lors de la survenance d'une interruption générée par l'horloge système (2éme partie).



IV.5: Communication inter-processus

Le modèle Producteur Consommateur

Ce modèle intervient lui aussi au niveau de la gestion du parallélisme d'exécution des processus, dans le cas spécifique d'une collaboration entre processus par échange temporisé de données . L'un des processus place les données en tampon, alors que l'autre lit celles-ci ultérieurement. Un problème de passage des données survient lorsque le tampon étant plein le processus producteur ne peut se poursuivre, et pour éviter de laisser ce processus en attente d'une interruption d'horloge, celui-ci suspend lui-même son exécution et rend la main au système. Le processus consommateur va résoudre partiellement le problème en lisant et en utilisant le contenu du tampon, jusqu'à ce qui n'y ait plus de données à traiter. Tour-à-tour les processus producteur et consommateur rendent la main au système lorsqu'ils sont dans l'incapacité de poursuivre leurs traitements respectifs et interdépendants.

Le partage de données entre processus implique dans ce cas de considérer la mémoire tampon comme une ressource partageable entre ces seuls processus, et nécessite l'instauration de sémaphores compteurs.

Au niveau de l'allocation mémoire il n'a pas été prévu de zone mémoire spécifique pour les tampons d'échange de données, du fait de l'exigüité de la mémoire centrale et des mécanismes supplémentaires à mettre en oeuvre. La zone tampon définie est incluse dans l'espace mémoire des données non initialisées du processus producteur (fig n°IV.17).

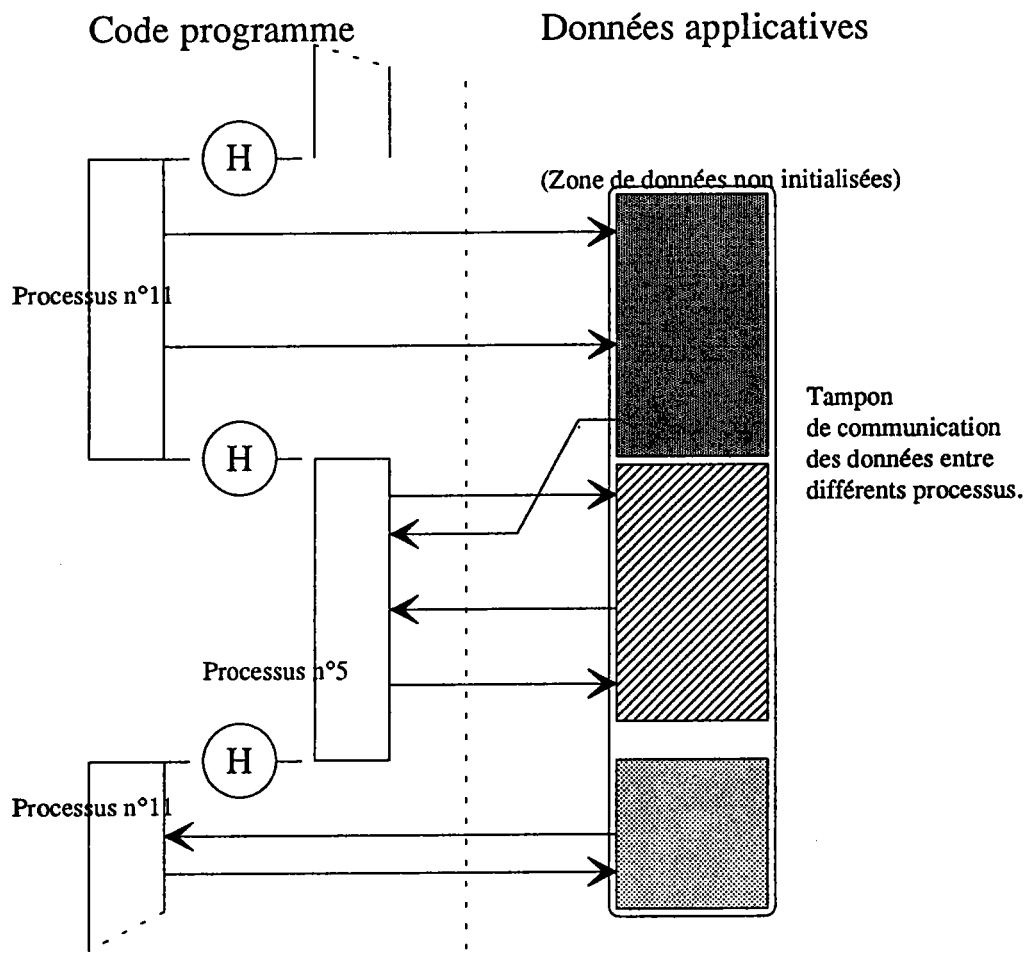


Fig n°IV.17: Partage de données inscrites en mémoire entre plusieurs processus.

IV.6 La gestion du temps et des processus

La gestion du temps et des problèmes qui en découlent est spécifique des systèmes multitâches ou/et multiutilisateurs. Les travaux sont effectués selon un ordre de priorité temporelle croissante en mode différé ou en mode conversationnel (traitement immédiat), puis en temps réel (contraintes de temps fortes).

La multiprogrammation (multitâche, multiutilisateur) a essentiellement pour but d'optimiser l'utilisation du microprocesseur. Elle implique un partage des ressources en matériel et du temps et donc un contrôle strict de la part du système sur ceux-ci.

Structure commune à tout système ou exécutif

Le structure d'un système d'exploitation ou d'un moniteur est fonction des possibilités qu'on souhaite lui donner. Il n'y a rien de commun entre la structure d'un exécutif monotâche

et celle d'un système multitâche/multiutilisateur. Par contre les structures multitâches et multiutilisateur sont voisines. Une structure multiprogrammée nécessite d'être constituée:

-de fonctions de manipulation des processus en file d'attente ou en liste.
-de fonctions de gestion dynamique des indicateurs associés aux processus (état, priorité).
-de fonctions d'ordonnancement des processus.
-des routines de traitement des interruptions matérielles et logicielles.
-de fonctions de lancement et de récupération d'un processus en fonction de ces caractéristiques.
-de fonctions de création et destruction de processus (des éléments descripteurs de contexte processus).
-de fonctions de sauvegarde et restauration de contexte et d'environnement des processus.
-de fonctions de création, destruction d'éléments de tableau d'allocation mémoire.
-de fonctions d'allocation et d'initialisation de la mémoire.
-de fonctions de contrôle d'accès à la mémoire.
-de fonctions de manipulation des espaces mémoire (extension, réduction) et de leur représentation.
-de fonctions pilotes de périphériques de base.
-de fonctions spécialisées liées au microprocesseur utilisé, pour la gestion des interruptions et des sémaphores.
-de différents tableaux de sauvegarde des contextes et environnements des processus.
-de tables de gestion mémoire mémorisant la cartographie mémoire.
-de files d'attente ou de listes d'exécution.

L'ensemble de ces fonctions et tables constitue le noyau de base d'un exécutif (constituant une couche logicielle), sur lequel se greffent des ensembles de fonctions structurées en bibliothèques de gestion spécialisées des interfaces (pilotes de périphériques). Les autres fonctions orientées sur l'extérieur et notamment l'utilisateur (interpréteur, interface graphique) ne font pas partie du moniteur.

Traitement des tâches en mode séquentiel

Ce mode de fonctionnement est celui des systèmes monotâches et monoutilisateurs pour lequel les différents travaux sont exécutés complètement les uns à la suite des autres. Un seul processus mobilise l'intégralité des ressources matérielles disponibles. Par exemple l'intégralité de la mémoire centrale est allouée à l'unique application. Tous les périphériques sont disponibles de façon permanente quel que soit leur mode de fonctionnement (sous interruption, par scrutation). Aucune contrainte de temps n'intervient dans ce mode de fonctionnement.

Traitement des tâches en temps partagé ("time sharing")

Ce mode de gestion part de l'idée d'utiliser la différence de vitesse entre le CPU et les périphériques qui sont généralement très lents pour réaliser d'autres opération dans l'intervalle.

Le principe de base est d'allouer périodiquement, dans le cadre d'un système multiprogrammé, à chaque tâche devant s'exécuter une tranche de temps CPU (time slice) fixe ou variable. La périodicité étant fonction du nombre de tâches à traiter "simultanément".

Algorithmes d'ordonnancement

La fonction centrale ("dispatcher", "scheduler","job controler") du système d'exploitation ou d'un exécutif, choisit l'ordre des processus à exécuter et attribue le contrôle de l'unité centrale à tour de rôle aux différentes tâches en compétition selon une politique prédéfinie par le constructeur et le contexte d'utilisation du système (environnement industriel, en réseau).

Pour définir un algorithme d'ordonnancement différents paramètres peuvent être considérés:

-les priorités des processus, fixes ou évolutive dynamiquement.

-l'ordre de présentation du processus pour exécution.

-le temps d'exécution d'un processus, fixe ou variable.

-des files d'attente affectées de niveaux de priorité différents.

-ordonnancement selon le plus haut niveau de priorité:

On range les tâches à exécuter par ordre décroissant de priorité dans la liste des tâches éligibles ou en file d'attente (fig n°IV.18). Puis on prend la première tache éligible, qui se voit allouer toutes les ressources matérielles. L'arrêt d'exécution d'une tâche survient lorsque soit la tâche est entièrement terminée, soit la tâche s'interrompt d'elle-même, soit parce que le système lui en donne l'ordre en utilisant une horloge système à expiration de sa tranche de temps.

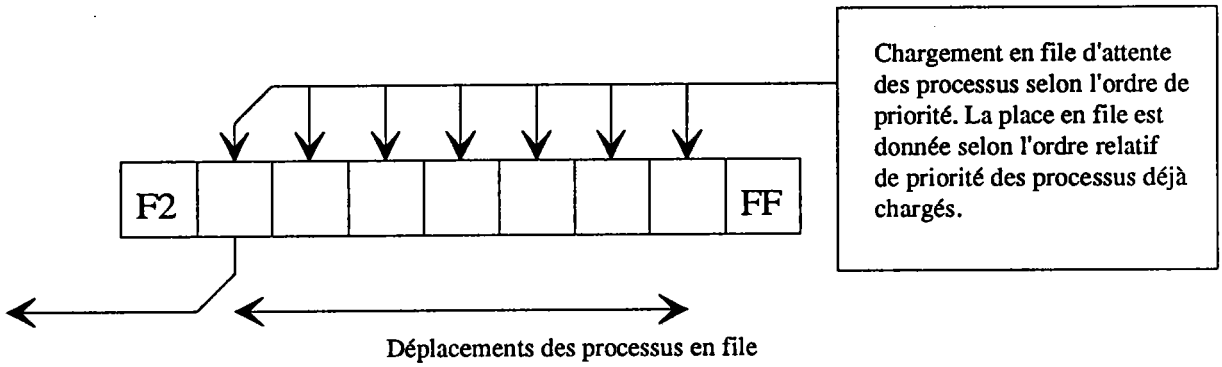


Fig n°IV.18 :Algorithme d'ordonnancement suivant le principe de la plus haute priorité.

Une tâche de priorité supérieure à celle de la tâche en cours d'exécution, entrant dans la liste des tâches éligibles, va interrompre la tâche en cours qui retourne dans la liste des tâches éligibles et l'ensemble des ressources sont réallouées au processus de priorité la plus forte. Ce principe de fonctionnement est celui de la préemption.

-ordonnancement selon l'ordre d'arrivée (principe du tourniquet):

La première tâche présente dans la liste des tâches éligibles ou dans une file d'attente se voit allouée la totalité des ressources durant une unité de temps d'exécution. Cette tâche est ensuite rechargée en liste éligible ou en fin de file d'attente. La tâche la première arrivée est la première exécutée (fig n°IV.19).

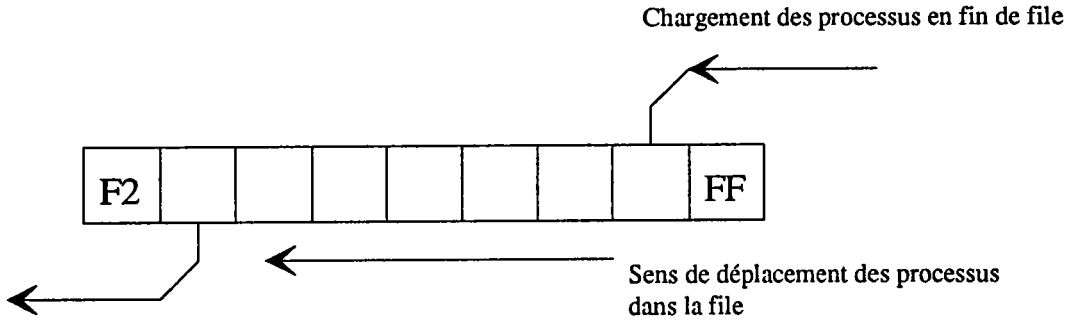


Fig n°IV.19: Algorithme d'ordonnancement suivant le principe du premier arrivé, premier servi.

Cette méthode du tourniquet ("round-robin") est souvent utilisée dans les systèmes en temps partagé avec traitement en différé, mais si le nombre de processus à exécuter est important la périodicité d'exécution d'un processus augmente et pour des systèmes interactifs cette méthode doit être utilisée en combinaison avec une autre.

L'une des variantes peut être l'association du tourniquet avec des processus affectés d'un niveau de priorité, le temps d'exécution dépendant de la priorité du processus. Chaque processus se voit garantir une exécution périodique.

-ordonnancement selon des files rétroactives:

On utilise ici différentes files d'attente (de priorités décroissantes) fonctionnant selon quelques règles simples:

-1) une tâche nouvellement créée (nouvelle exécution ou première exécution) est mise dans la première file.

-2) à chaque exécution la tâche est chargée dans une file de niveau inférieur.

-3) chaque file fonctionne selon le principe du tourniquet.

-4) ce n'est que lorsque tous les processus d'une file sont exécutés que l'ordonnanceur scrute la file de niveau immédiatement inférieur pour en exécuter les processus.

Cette méthode conduit à traiter préférentiellement les processus les plus courts sans que les temps d'exécution de système ne soient connus au préalable par le système, mais ce type de gestion est lourd à mettre en oeuvre et gourmand en mémoire pour les files d'attente.

Stratégie d'ordonnancement

Les mécanismes décrits ne peuvent à eux seuls effectuer la tâche d'ordonnancement, et le choix d'une stratégie est nécessaire car il permet une mise en application pratique du mécanisme choisi.

-priorité aux tâches les plus courtes.

Le temps nécessaire à chaque tâche pour être exécutée est estimé. Le niveau de priorité affecté à cette tâche est inversement proportionnel au temps CPU d'exécution. Les tâches les plus courtes sont traitées en premier. Utilisée conjointement avec un mécanisme privilégiant la plus haute priorité et avec un renouvellement important des tâches courtes, cette méthode pénalise les processus consommant du temps CPU. Il est impossible de prédire le moment de leur exécution et la période d'exécution est longue. De plus il est impératif pour le système de connaître le temps moyen d'exécution de chaque processus.

-priorités variables.

Chaque tâche commence son exécution avec un niveau de priorité défini. Une échelle de priorités des tâches peut déjà exister. Le niveau de priorité d'une tâche va alors fluctuer en fonction de certains paramètres tels que:

-le nombre de tranches de temps nécessaire pour son exécution (évolution linéaire). Plus une tâche utilise de tranches de temps et plus son niveau de priorité diminue.

-la survenance d'événements internes (échange d'informations interprocessus) ou externes (suite à l'arrivée d'une donnée par une interface).

-priorité aux Entrées/Sorties.

On choisit dans ce cas de favoriser le traitement des tâches liées aux Entrées/Sortie et les temps de réponse par rapport aux périphériques sont très courts, et sont garantis même en cas de charge importante de traitements.

IV.7:Structure de l'exécutif multitâche du terminal

Nous venons de voir les différentes structures possibles pour constituer un système d'exploitation ou exécutif multiprogrammé et notamment les différentes politiques et stratégies utilisées par des systèmes connus.
Ce sont les règles de fonctionnement appliquées par la fonction d'ordonnancement qui fixent le comportement général du système. Les mécanismes précédemment définis de synchronisation et de communication des processus par rapport aux périphériques sont indépendants de la fonction d'ordonnancement. Ils interviennent cependant par adaptation du noyau au monde extérieur et particuliérement aux périphériques non partageables.

Le gestionnaire des processus internes

Plusieurs types ou classes de processus ont été définis permettant au système d'adapter son comportement et sa gestion aux particularités de ceux-ci. Un processus interne est un programme résident et indépendant ayant une référence en mémoire centrale de l'équipement sous la forme d'un élément descripteur de contexte de ce processus (pour le gestionnaire de processus) et un ou plusieurs éléments descripteurs d'allocation mémoire. Le nombre de processus internes est en principe infini. Cependant il se trouve limité d'une part par la taille de la mémoire centrale sur le plan matériel, et d'autre part par la représentation en descripteur de contexte de processus (codage du numéro de processus sur huit bits) qui n'autorise que 256 processus.

Contenu du descripteur de contexte de processus

Chaque processus est représenté par un seul élément descripteur de contexte de processus. En mémoire centrale pourront êtres décrits 256 éléments descripteurs de contexte

représentés par quatre mots longs, soit 4096 octets. Le nombre d'enregistrements liés à un processus ne multiplie pas le nombre d'éléments descripteurs de contexte.

Les informations nécessaires à l'exécution d'un processus se distribuent en deux catégories: les informations constantes quel que soit le nombre d'exécutions du processus d'une part. (ces informations telles que le nombre d'enregistrements, le type de processus, son implantation, son niveau de priorité sont stockées en descripteur de contexte). Les informations à caractère évolutif sont placées en élément descripteur d'allocation mémoire (l'état du processus, le nombre de quanta de temps consommé par le processus...).

Le mode de représentation choisi pour des systèmes tels UNIX (notion de processus père/fils) est repris partiellement. En effet chaque processus est représenté par un élément descripteur de processus, et chaque processus exécuté dispose d'un élément descripteur d'allocation mémoire.

Les files d'attente

En fonction de l'indice de priorité stocké en élément descripteur de contexte, le numéro de processus est placé dans une file d'attente d'exécution correspondant à ce niveau. Chaque file d'attente est structurée avec un indicateur de début (F2F2 pour la file de priorité la plus élevée) et un indicateur de fin de file (FFFF quelle que soit la file). Dans une file sont stockés le numéro de processus (un octet)à exécuter et le numéro d'espace mémoire nécessaire à l'exécution du processus (fig n°:IV.20).

Fonctionnement d'une file

Chaque file d'attente a un fonctionnement dynamique et sa taille évolue en fonction du nombre de processus à exécuter. Un processus à exécuter est chargé en fin de file. Le processus exécuté est le premier processus de la file (placé juste après l'indicateur de début de file). Un processus interne perdant le bénéfice du processeur (par interruption d'horloge système) est rechargé en fin de file, puis le contenu de celle-ci est translaté.

Système d'ordonnancement mixte

Le principe adopté pour l'ordonnancement des tâches selon une gestion multitâche consiste à utiliser un système mixte par priorité et par tourniquet, (figure n°IV.20). Aucune méthode de gestion multiprogrammée n'est pleinement satisfaisante.

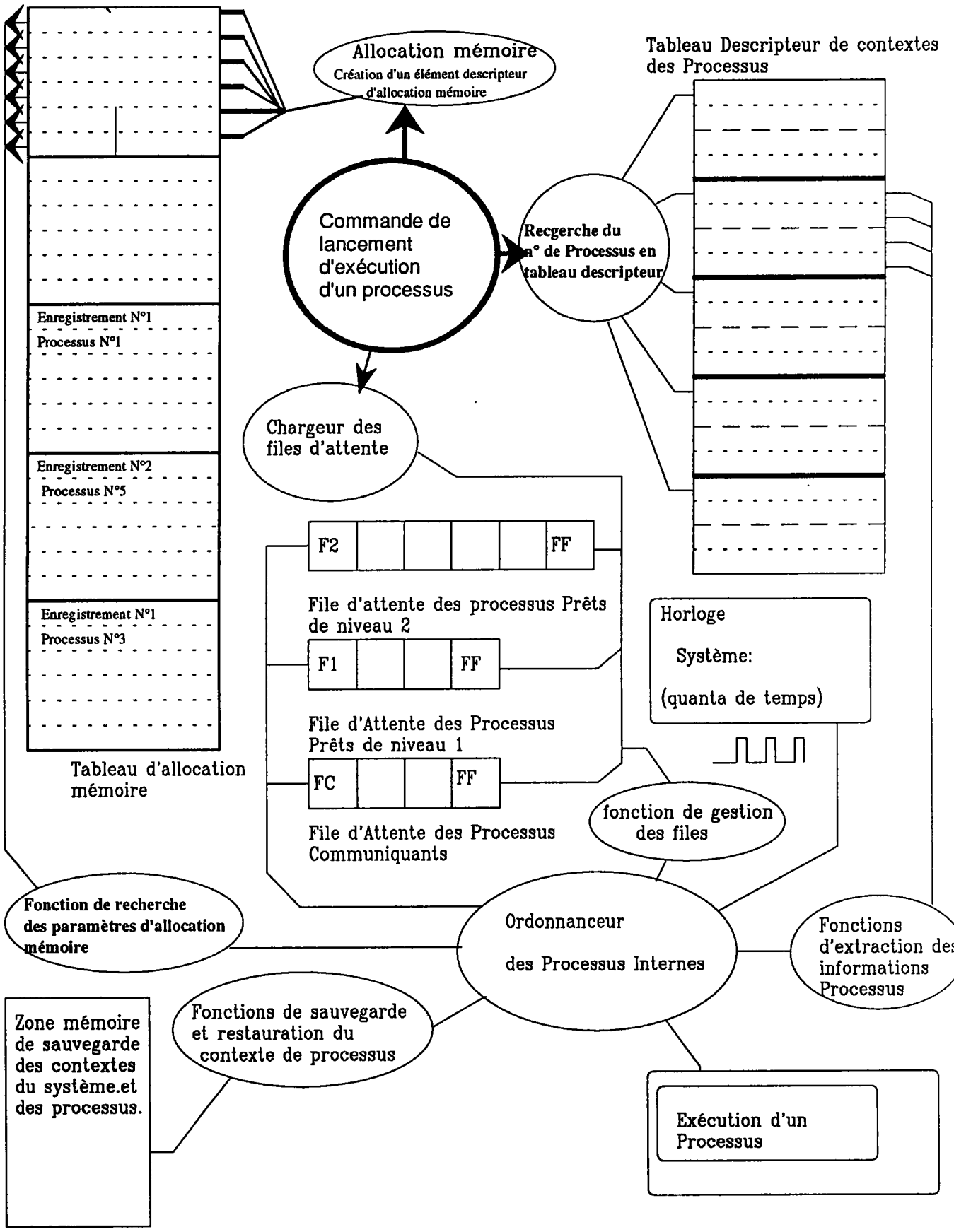


Figure n°IV.20 : Les échanges entre les différents éléments du noyau.

La méthode d'ordonnancement par tourniquet est simple de conception et de mise en oeuvre. Plus le nombre de processus à exécuter est important et plus le temps séparant deux exécutions partielles d'un même processus est long. Chaque tranche de temps d'exécution (fixe) est de l'ordre de 30 ms.

Le système fonctionne à partir de deux files d'attente affectées chacune d'un niveau de priorité, et la méthode du tourniquet s'applique à chacune d'elle. La fonction d'ordonnancement qui constitue la tâche de fond du système, scrute la file de niveau le plus élevé et lance l'exécution des processus qui s'y trouvent. Une tâche non exécutée entièrement est rechargée en fin de file. La file d'attente de priorité inférieure est scrutée et ses travaux exécutés seulement lorsque celle des tâches de niveau 2 est vide. Sont exécutés en premier les travaux prioritaires (niveau 2) qui correspondent à des tâches utilisateur. Les autres travaux sont des travaux différables et périodiques tel que le processus de test automatique du matériel.

Pénalisation des processus gourmands en temps d'exécution

En principe tous les travaux de même priorité sont exécutés à leur tour durant des tranches de temps égales mais pour favoriser le traitement des processus utilisant peu de temps d'exécution, un système de pénalisation périodique des processus longs est instauré. Le descripteur d'allocation mémoire contient l'information de quantité de temps d'exécution utilisé par le processus. Lorsqu'un processus a consommé un certain nombre de quanta de temps, c'est à dire lorsque son indicateur est nul, le processus se présentant à l'exécution se voit rechargé directement en fin de file sans exécution , puis se voit allouer deux nouvelles unités de temps d'exécution.

Le mécanisme de préemption

Pour réaliser la multiprogrammation, il est nécessaire de mettre en place un mécanisme de lancement ou relancement puis d'arrêt d'exécution d'un processus. La fonction d'ordonnancement comporte une procédure de lancement (ou relancement), mais il a fallu au préalable arrêter l'exécution d'un processus précédent.

Non-préemption

Le processus dispose de la faculté de stopper lui-même son exécution et de céder l'utilisation du processeur au système et notamment à l'ordonnanceur, ce qui est réalisable à la condition que cela soit prévu lors de la conception des processus pour que chacun ne s'exécute pas plus d'un certain temps. La découpe de service réalisée en processus s'opère en fonction du temps et en unités fonctionnelles. Aucune horloge régulatrice n'est nécessaire, mais ceci impose d'assujettir la structure et l'écriture des processus à la durée de temps choisie. On

impose au programmeur d'applications de tenir compte dans son développement des contraintes de temps. Les processus s'exécutent alors entièrement.
La non-préemption des processus est utilisée pour les échanges de données en processus selon le modèle producteur-consommateur.

Préemption

L'autre solution est celle qui a été retenue, et elle nécessite l'utilisation d'un composant utilisé en horloge. Le signal périodique fourni sert à marquer les intervalles de temps d'exécution et vient interrompre l'exécution d'une tâche. C'est un mécanisme de préemption qui laisse la gestion du temps à la charge du système. Par le signal d'horloge le système prend d'autorité le contrôle des opérations, sauvegarde les informations d'environnement du processus interrompu et poursuit les opérations de choix d'exécution du processus suivant (fig n°:IV.21). Ce mécanisme offre deux avantages. Il permet éventuellement de moduler le temps d'exécution d'un processus en fonction de ses caractéristiques et aucune contrainte de conception en fonction du temps n'est imposée au développeur d'applications. C'est le mécanisme généralement utilisé pour rythmer l'exécution des différents travaux.

La modulation du temps d'exécution d'une application constitue une autre politique d'exécution, permettant d'introduire d'autres facteurs de régulation. Le nombre de processus attendant leur exécution est l'un d'entre eux. Cette modulation du temps ne peut être réalisée que dans certaines limites bien définies. Il existe un temps d'exécution minimal, au-dessous duquel le processeur passe relativement plus de temps à réaliser des opérations de commutation de contexte et d'environnement que d'exécution des applications (limite inférieure de 15 ms pour un rapport de temps système/temps application). Dans l'autre sens un processus ne doit pas mobiliser à son avantage tout le système durant une grande période de temps, car on perd vis à vis de l'utilisateur la notion de multitâche et de quasi simultanéité (limite supérieure de l'ordre de 60ms avec un rapport de 1/10).

L'ordonnancement des processus externes

Les processus externes

Les processus externes (en nombre réduit) sont des processus spécialisés d'exécution rapide, ne faisant pas généralement appel à la ressource graphique, et ne sont pas gérés par le mécanisme d'ordonnancement mis en place. Ils ne sont pas référencés en descripteur, et sont appelés directement soit par exception, soit par interruption. Leur intervention est précise et spécifique, Ils sont conçus pour exécuter rapidement certaines opérations liées à des événements nécessitant un traitement immédiat. Un processus externe a la faculté de décider de l'exécution d'un processus interne.

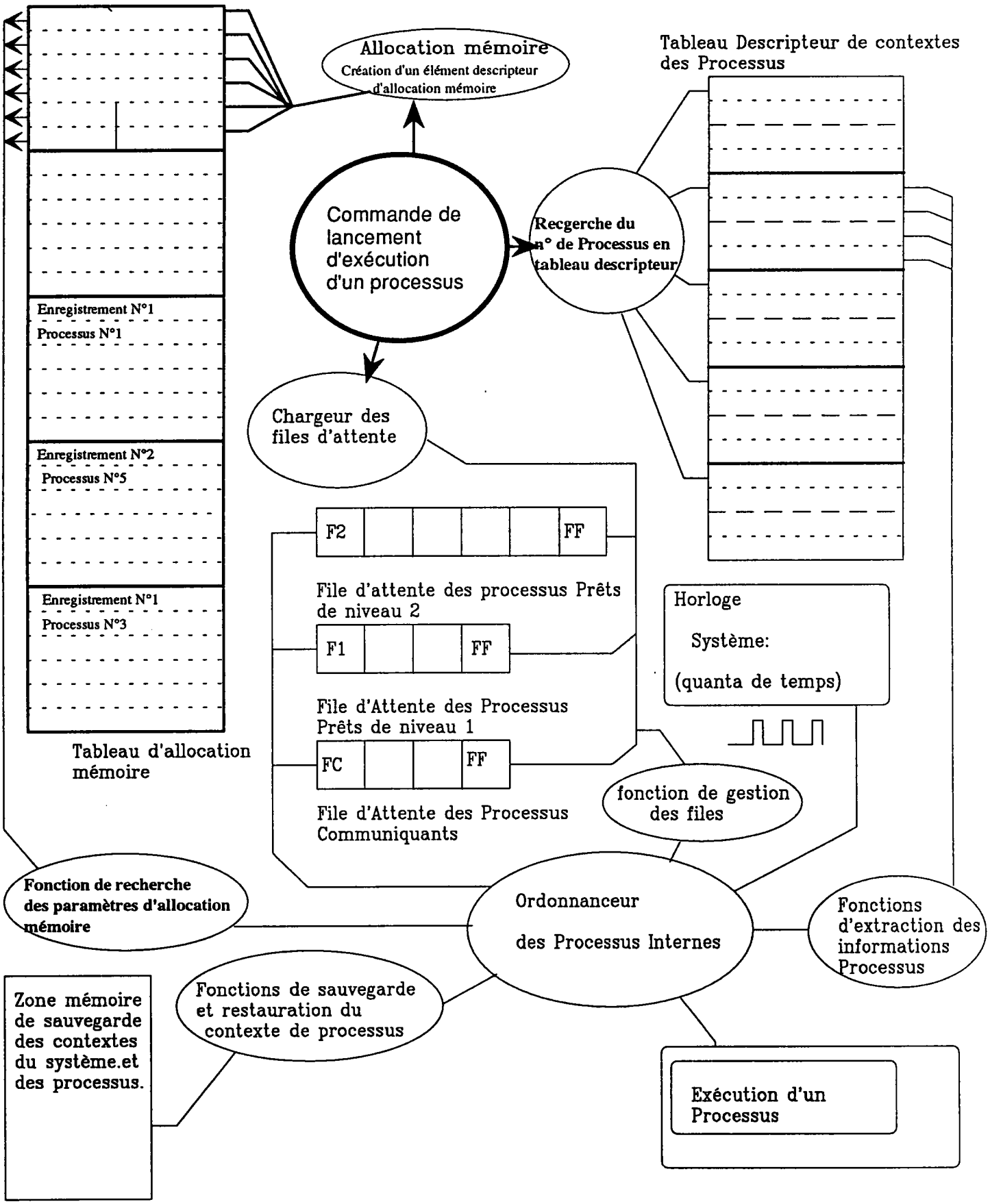
La distinction que j'établis entre une routine de traitement d'interruption et cette notion de processus interne s'explique par la philosophie du système multitâche. Comme un processus interne, il constitue un programme indépendant, effectuant un traitement particulier, d'intervention adaptée et évolutive. L'exemple type en est le processus de réception des données en provenance du réseau, qui reçoit un caractère au niveau d'un traitement minimal, et opère les traitements portant sur la trame et le stockage définitif en mémoire dans le cadre complet du traitement du processus.

L'ordonnanceur de ces processus externes n'est pas logiciel, mais matériel puisque constitué par le mécanisme matériel de gestion hiérarchique des interruptions (fig n°IV.5). La priorité logique du processus correspond à son niveau d'interruption matériellement installé. Le nombre de processus externe est forcément limité au nombre d'entrées d'interruption, mais il peut être étendu soit par l'emploi de démultiplexeurs et d'une logique matérielle de détermination de la source d'interruption, soit par l'utilisation d'un élément logiciel supplémentaire permettant de déterminer la source d'interruption parmi plusieurs sources connectées à la même entrée physique.

L'avantage d'un tel procédé est de fournir un délai de traitement rapide, limité au détournement et au retour d'interruption, puis au traitement. Ce mécanisme est celui faisant la différence avec nombre d'autres systèmes.

En revanche l'inconvénient réside dans la liaison instituée entre le niveau d'interruption et le matériel. En cas de modification de priorité d'une tâche il faudra modifier la table d'interruption et changer physiquement la connexion de la source d'interruption.

Figure n°IV.20 : Les échanges entre les différents éléments du noyau.



IV.8: Les opérations réalisées par le gestionnaire de processus internes

Lancement d'exécution (fig n°IV.21.a)

L'opération de lancement d'exécution d'un processus quelle que soit sa nature ou son type est intégrée dans le processus d'ordonnancement. Elle consiste à charger les registres internes spécialisés aux accès des différentes zones allouées à un processus (programme, données, pile), puis au moment opportun à rompre l'enchaînement logique des opérations par manipulation de la pile. En effet lors de l'appel d'une fonction le processeur mémorise uniquement en pile l'adresse de retour à la fonction ou au programme principal et pour détourner l'exécution du programme, il suffit de modifier cette adresse et de la remplacer par l'adresse correspondant à la partie du programme où l'on désire réorienter l'exécution (fig n°IV.21). L'ordre des opérations est spécifique car il faut au préalable réaliser l'accès à la pile ou se trouve l'adresse de la prochaine exécution du détournement, avant d'effectuer le déroutement. L'importance de l'ordre des opérations et de leur nature s'illustre lors de la reprise d'exécution d'un processus. En effet le processus dispose d'une pile locale, dont les derniers éléments (les plus récents) décrivent l'environnement du processus au moment de son arrêt provisoire par l'horloge système. On y trouve, dans un ordre défini et invariable, les valeurs des registes internes du processeur, la valeur du pointeur ordinal, le contenu du registre d'état, les valeurs de pointage en mode utilisateur et en mode superviseur de la pile système et de la pile locale. Dans la fonction de relancement on rétablit la connexion avec les zones piles et données, on remet en place les valeurs contenues dans les registres de données et enfin on effectue le retour de sous-programme qui réalise le détournement. Le processus peut alors s'exécuter, par rapport aux différentes zones qui lui ont été attribuées.

Intervention du système aprés exécution totale ou partielle d'un processus (fig n°IV.21.b)

Il est nécessaire de prévoir le comportement du système:
-à la suite d'un signal d'horloge

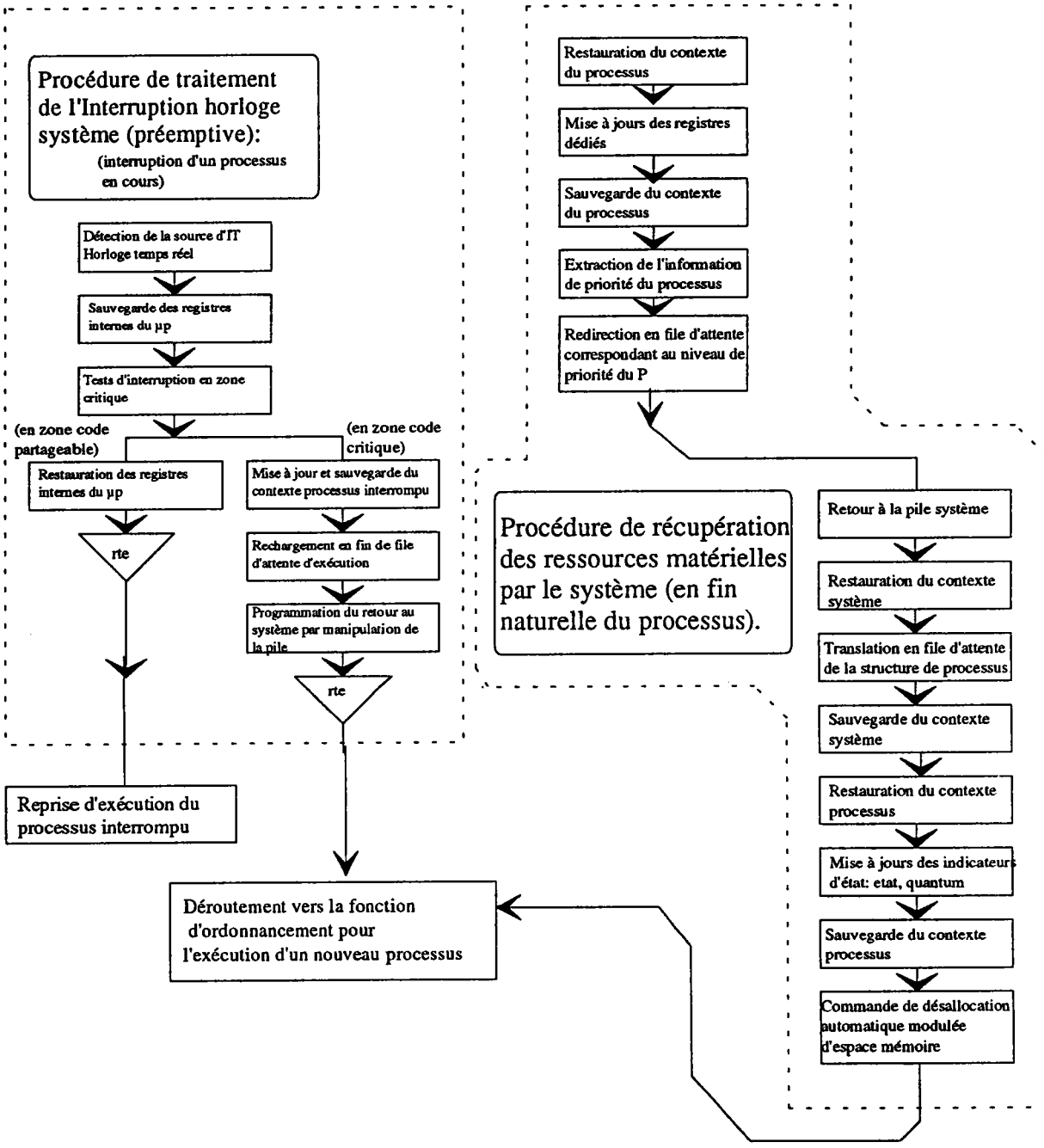
Cette opération est réalisée à la suite de la survenance du signal d'horloge appliqué sur une entrée d'interruption. Lors de l'exécution d'un sous-programme d'interruption après avoir testé la non activation d'un sémaphore, sont effectuées les opérations de sauvegarde et de mise à jour du contexte du processus (processus interrompu en tableau descripteur), de sauvegarde

de l'environnement c'est à dire du contenu des registres internes (dont les registres d'accès à la pile locale,et la pile secondaire). Un deuxième ensemble d'opérations concerne le déroutement vers le système. A la suite de l'exécution de la routine d'exception matérielle, la suite logique des opérations suppose de revenir au processus. Pour le quitter il faut modifier l'adresse de retour au programme principal placé en pile système après avoir rétabli l'accès à celle-ci. Ainsi les opérations microprogrammées au niveau du processeur ne sont pas perturbées, elles opèrent seulement sur des valeurs modifiées. Le détournement (fig n°IV.21) dans ce cas conduit à revenir au début de la phase d'ordonnancement lors de la scrutation des files d'attente.

-en fin naturelle du processus

Tous les processus ne sont pas conçus pour s'exécuter en boucle infinie. Il faut donc prévoir une procédure particulière permettant au système de savoir que le programme a terminé son exécution, et que les opérations d'ordonnancement peuvent reprendre. Plusieurs solutions sont concevables et entraînent des développements plus ou moins importants. En premier lieu on peut intégrer ou faire appel à une boucle d'attente infinie, qui sera interrompue par l'horloge système. Ensuite, il est concevable que le processus signale sa fin par un indicateur. Et enfin la solution qui a été retenue consiste pour le développeur à intégrer en dernière ligne de son programme une fonction spéciale de déroutement vers le système pour réaliser d'autres opérations. Cette solution a été retenue, car en l'absence de tout signal d'horloge, le fonctionnement est de type monotâche.

Fig n°:IV.21b: Fin ou interruption d'exécution d'un processus



Les opérations de préparation d'activation d'un processus

Les opérations présidant à la prise de commande de l'ordre de l'utilisateur d'avoir accès à un service concernent l'interpréteur. Cependant, une fois que cet ordre a été reçu, validé, décodé, et que le système est entré en mode superviseur, il faut lancer l'exécution du processus correspondant. Un processus pour passer de l'état endormi à l'état actif, c'est à dire être chargé dans une file d'attente, doit être au préalable reconnu par son type, sa nature, puis

se voir attribuer un espace mémoire de travail de base dont les caractéristiques sont consignées dans un élément descripteur d'allocation mémoire. Il reste encore à déclarer le processus actif avant d'effectuer son chargement dans la file d'attente correspondant au niveau de priorité de celui-ci (fig n°: IV.21). Le chargement en file d'attente d'un processus garantit à ce processus qu'il sera exécuté.

Plus précisément les opérations d'activation d'un processus vont porter sur le numéro et le nombre d'enregistrements référencés (une exécution commandée pour un processus correspond à un enregistrement), puis sur l'état du processus et la commande de désallocation mémoire prévisible automatique, et la taille de la mémoire. On observe que ces élements portent essentiellement sur les variables ayant trait à la mémoire, éléments constitués à partir de paramètres invariables intégrés dans l'élément descripteur de contexte du processus concerné.

A ce moment seule la ressource mémoire est attribuée officiellement au processus; les autres ressources matérielles ne lui sont pas attribuées et ceci pour deux raisons. S'il fallait attendre que toutes les ressources soient disponibles pour activer un processus et le charger en file, seulement un ou deux seraient présents en file (limitation), et bon nombre d'ordres d'exécution d'un processus ne seraient pas pris en compte.Et enfin un processus n'utilise pas tout le temps les mêmes ressources; il est inutile de réserver l'utilisation d'une ou plusieurs ressources à un seul processus durant tout le temps de son exécution.

La majorité des traitements liés aux événements survenant dans l'exécution d'un processus, intègrent les opérations de sauvegarde et de restauration du contexte et de l'environnement du processus.

Sauvegarde et restauration du contexte d'un processus

Les paramètres d'un processus évoluant dynamiquement sont regroupés dans son élément descripteur d'allocation mémoire, et y sont modifiés à chaque événement important. Il portent sur son état, sur le nombre de quanta de temps consommé et sur la valeur courante des différents pointeurs d'accès aux zones code, données, pile.

Lors de l'opération de sauvegarde, les contenus des registres spécialisés D0-D3 sont chargés dans l'élément descripteur d'allocation mémoire correspondant au numéro de processus et au numéro d'enregistrement. Au préalable une opération de tri effectuée à partir de ces deux paramètres permet de retrouver l'élément descripteur concerné pour y faire le transfert. Les adresses d'accès aux zones données et pile sont converties en déplacement par rapport à une adresse de référence présente dans le descripteur, et non pas par rapport à une adresse de référence courante.

L'opération de restauration comme l'opération précédente suppose d'effectuer un tri à partir de ces deux mêmes paramètres (n° de processus et n° d'enregistrement) puis de calculer à l'aide de l'adresse de référence et de différentes valeurs de déplacement les adresses réelles physiques de pointage des différentes zones données et pile. Un traitement particulier est réservé pour l'adresse d'accès à la zone code du processus (en sauvegarde ou restauration) selon que le processus est résident ou importé. Pour un processus résident l'adresse physique est stockée directement "en clair" dans l'élément descripteur, alors qu'elle y figure sous forme de déplacement pour un processus importé. Cette particularité est naturellement prise en compte dans les fonctions de lancement d'exécution, de restauration et de sauvegarde du contexte du processus.

Sauvegarde et restauration du contexte système

Ces opéations concernent les informations vitales au système pour l'accès aux différentes tables du systèmes; les pointeurs de base et d'accés courant aux files d'attente, l'adresse de base des tableaux descripteurs de contexte de processus et tableaux descripteurs d'allocation mémoire, et enfin adresse courante de la pile. Ces valeurs, stockées à l'extrémité haute de la RAM et chargées en registres d'adressse du µp, donnent un accés indirect à ces tables. L'utilisation dédiée aux variables systèmes des registres internes du microprocesseur n'est réalisée qu'en phase système (phase d'ordonnancement).

Les protections

Sur un microprocesseur qui exécute un million d'instructions par seconde,il est possible de faire cohabiter plusieurs dizaines voir centaines de tâches indépendantes. Ces tâches étant activées séquentiellement ou pseudo parallèlement, il convient de définir des protections placées à différents niveaux pour éviter des conflits d'exécution entre tâches, des pertes de données. Les outils disponibles sont d'ordres différents:

-la séparation des tâches en mémoire.

-les instructions privilégiées du processeur. Certaines opérations ne peuvent être effectuée qu'en mode superviseur, pour éviter des manipulation désastreuses de données à l'intérieur des processus.

-les différents modes de fonctionnement du processeur (mode superviseur, mode utilisateur). Ces modes de fonctionnement conditionne l'utilisation de différentes zones mémoire et dans une moindre mesure les instructions disponibles.

-les différents type de processus offrant une sécurité de fonctionnement pour les processus résidents.

Utilisation dédiée des registres internes du processeur

Quatre registres internes de données (D0, D1, D2, D3) du microprocesseur sont réservés à l'utilisation du système lorsque celui-ci se trouve en phase d'ordonnancement. Ils sont utilisés pour stocker provisoirement les données caractéristiques et dynamiques du processus dont l'exécution immédiate est programmée.

Les informations d'état du processus, ainsi que les valeurs des pointeurs d'accès aux différentes zones mémoire, nécessaires à l'exécution du processus sont chargées dans les registres internes à partir de l'élément descripteur d'allocation mémoire correspondant. Cette opération a lieu à différents moments, au début de la phase d'ordonnancement et de la phase de déroutement. L'opération inverse est réalisée juste avant et après l'exécution du processus.

La recherche des variables dynamiques d'exécution d'un processus dans le tableau descripteur est une opération longue. Plus le nombre de processus et le nombre d'enregistrements est important, plus le temps de recherche est long. L'utilisation des registres internes permet d'économiser du temps pour l'accès à ces variables en vue de leur manipulation (commutation d'état...) au cours des opérations d'ordonnancement et d'interruption d'exécution. Le transfert entre élément descripteur et registre est global.

IV.9: Code programme généré

Code programme réentrant

Plusieurs tâches doivent pouvoir se partager la même "copie" d'un code objet sans qu'il y ait d'interférence entre elles. Le programme ne doit pas pouvoir s'automodifier, et chaque processus doit posséder une zone mémoire distincte pour les variables affectées à chaque exécution du processus.

Code programme récursif

C'est un cas particulier de sous-programmes imbriqués, pour lequel ce sous-programme peut s'appeler lui-même. Ce sous-programme ne peut alors écrire en RAM, mais seulement en pile qui est utilisée pour la transmission des arguments.

Code relogeable ou translatable

Cette caractéristique est nécessaire pour les programmes occupant la mémoire centrale de l'équipement, permettant une application correcte du mécanisme de gestion dynamique de la mémoire. Ceci implique que le code programme utilise les modes d'adressage indirects (adressage indirect avec déplacement, simple ou avec index, postincrémenté ou prédécrémenté, ou relatif au PC). Le code exécutable fourni par le compilateur C n'est à l'origine que relogeable (car non entièrement optimisé).

Les notions de relogeabilité et de translatabilité, parce que voisines, portent à confusion. Un code relogeable est un code défini en adressage indirect, faisant référence néanmoins à des adresses définies par l'opération de création de liens ("linkage") qui fournit un code exécutable. Les adresses d'un code relogé en un autre espace mémoire doivent êtres recalculées par le gestionnaire de mémoire, qui intervient donc au niveau du code exécutable. Un code translatable est un code positionnable et exécutable en n'importe quel endroit de la mémoire (sans calculs préalables).

Les langages de haut niveau fournissent un code relogeable, et non translatable du fait d'un choix d'optimisation (taille ou translatabilité). Seul l'assembleur permet de réaliser des programmes entièrement translatables.

Portabilité du code

Le code généré par un outil de développement est qualifié de portable lorsque les programmes générés sont exécutables sur différentes machines, ce qui implique l'utilisation de structures et de fonctions standardisées (norme ANSI du langage C) de la part du développeur.

IV.10: Conclusions

En somme...

Ce double système d'ordonnancement des tâches est basé sur la notion de priorité: priorité des taches externes sur les tâches internes, priorité des tâches externes et des tâches internes entre elles. Le principe d'ordonnancement des processus internes est mixte et associe une gestion par priorité et par tourniquet des tâches en attente d'exécution. Les processus internes et leur assemblage en programmes d'application correspondent à des applications complètes, faisant intervenir toutes les ressources matérielles, et les processus externes effectuent des traitement spécialisés, immédiats, et limités mettant en oeuvre un petit nombre d'interfaces.

Fig n°:IV.21: Organigramme d'Ordonnancement

Procédure de traitement de l'Interruption horloge système (préemptive):
(interruption d'un processus en cours)

Détection de la source d'IT Horloge temps réel

Sauvegarde des registres internes du pp

Tests d'interruption en zone critique

(en zone code partageable)

Restauration des registres internes du pp

rte

Reprise d'exécution du processus interrompu

(en zone code critique)

Mise à jour et sauvegarde du contexte processus interrompu

Rechargement en fin de file d'attente d'exécution

Programmation du retour au système par manipulation de la pile

rte

Procédure d'installation du progiciel:

Test et définition de la taille de la mémoire embarquée.

Chargeur des variables globales du système en RAM.

Initialisation et configuration des périphériques: -I2C
-RS232C
-VDP
-PTM

Test d'installation préexistante du système (variables instal).

Installation des différentes tables files de l'exécutif multitâche.

Chargement des tables et tableaux du système

Positionnement de l'indicateur d'installation de l'exécutif

Test de fonctionnement des interfaces du terminal:
-I2C
-RS232C
-VDP
-PTM

Test de reconnaissance du dysfonctionnement logiciel

Restauration du TDCP

Restauration du TDAM

Restauration des files et tableaux annexes

Module de réinitialisation différenciée et automatique des périphériques et du système.

Réinitialisation individuelle d'un périphérique

Réinitialisation du contrôleur d'interface parallèle/I2C

Réinitialisation du port série RS232C (logiciel)

Rinitialisation des principaux registres du VDP

Procédure de récupération des ressources matérielles par le système (en fin naturelle du processus).

Restauration du contexte du processus

Mise à jours des registres dédiés

Sauvegarde du contexte du processus

Extraction de l'information de priorité du processus

Redirection en file d'attente correspondant au niveau de priorité du P

Retour à la pile système

Restauration du contexte système

Translation en file d'attente de la structure de processus

Sauvegarde du contexte système

Restauration du contexte processus

Mise à jours des indicateurs d'état: etat, quantum

Sauvegarde du contexte processus

Commande de désallocation automatique modulée d'espace mémoire

Positionnement du masque d'interruption (niveaux 2)

Restauration du contexte système

Remise à zéro de la variable d'installation du système

Test de reconnaissance de début de file de niv2

Test de reconnaissance de fin de file de niv 2

Test de reconnaissance de fin de file de niv 1

Test de reconnaissance de début de file de niv 1

Positionnement du masque d'interruption au niveau

Restauration du contexte processus (zones memoire)

Extraction de l'information d'état du processus

Extraction de l'information de quantaum de temps

Test sur le nbre de quantums utilisés par processus

Modification de l'état du Processus

Sauvegarde du contexte Processus

Extraction de l'information de résidence

Test de résidence du code programme du processus

Lancement d'un processus résident en exécution

Lancement d'un processus importé en exécution

EXECUTION DU PROCESSUS

Rechargement en fin de file d'attente de la structure

Translation des processus en file d'attente

Réallocation de deux unités de temps supplémentaires

Positionnement du masque d'interruption au niveau 5

Positionnement du masque d'interruption au niveau

Restauration du contexte processus (zones memoire)

Modification de l'état du Processus

Sauvegarde du contexte Processus

Extraction de l'information de résidence

Test de résidence du code programme du processus

Lancement d'un processus résident en exécution

Lancement d'un processus importé en exécution

EXECUTION DU PROCESSUS

Opérations d'ordonnancement des processus internes

V MODELES ET MECANISMES DE GESTION MEMOIRE

V.1: Supports mémoire et représentations de l'information

Pour un système à microprocesseur, la mémoire est le lieu de stockage de toutes les informations nécessaires au fonctionnement de celui-ci (code programme et données initialisées, pile), et de toutes celles produites par lui (données non initialisées organisées communément en fichier).

Les stratégies de mise en oeuvre pour organiser et gérer la mémoire, qui elle aussi constitue un périphérique (par rapport au micro-processeur), sont très variées suivant le support matériel (mémoire centrale, ou support de masse) et suivant son utilisation (informatique scientifique, multimédia).

Les différents supports mémoire

Sur le plan matériel, lors de la conception d'un équipement quatre caractéristiques sont prédominantes pour le choix du ou des supports mémoire:

-la vitesse d'accès (lecture, écriture)
-le coût
-la capacité de stockage
-l'infrastructure matérielle mémoire

Ces quatre aspects sont interdépendants. La capacité de stockage est d'autant plus réduite que la vitesse d'accès et donc le coût sont plus élevés.

La mémoire volatile se compose:

-des registres internes de l'unité centrale.
-de la mémoire centrale du système.

La RAM (Random Access Memory) ou mémoire vive, constituée de circuits intégrés, autorise des accès très rapides notamment pour la RAM statique (lecture et écriture). Elle se décline en deux technologies: la RAM dynamique (nécessitant un dispositif de rafraîchissement de son contenu) de grande capacité (1Mo en un seul boîtier) et la RAM statique (équipant le terminal) rapide, plus chère et simple de mise en oeuvre.

La mémoire permanente: (ou mémoire de masse)

Elle utilise comme support, soit des circuits électroniques (ROM), soient des bandes (cartouche streamer), disques magnétiques (disquette ou disque dur) ou disques optiques numériques (CD ROM).

La ROM (Read Only Memory) est une famille de composants électroniques programmables, accessibles en écriture seulement. Une fois programmé ce composant ne peut alors qu'être lu. Un système additionnel de protection contre la duplication peut être mis en place. Sa capacité est la plus faible parmi les supports de mémoire permanente mais son accès est rapide. Ce composant est le plus adapté pour contenir les programmes non évolutifs et non automodifiants tels que l'exécutif.

Les supports magnétiques autorisent le stockage de grandes quantités de données, à des coûts raisonnables et de façon durable à condition de respecter certaines précautions d'utilisation.

Le CD-ROM est le dernier grand moyen de stockage mis au point et de capacité de stockage énorme. Il n'est pour l'instant pas encore réinscriptible (lecture seule).

Le terminal ne dispose pas actuellement d'unité de stockage de masse pour les données, ce qui implique une gestion très optimisée de l'espace mémoire RAM disponible, et la présence en ROM de l'exécutif, de l'interface graphique, et des programmes d'application résidents.

La mémoire virtuelle

Lorsque la taille de la mémoire physique est inférieure à la taille de la mémoire adressable (espace d'adresse logique) et lorsqu'une mémoire secondaire à accès aléatoire est disponible (disquette, disque dur), le manque de mémoire pour utiliser de gros programmes d'application peut être pallié par l'utilisation du procédé de gestion de mémoire virtuelle. Son principe est de placer en mémoire centrale (limitée) les éléments de programmes ou de fonctions d'utilisation fréquentes et de stocker les autres fonctions en mémoire secondaire [réf:]. L'appel d'une fonction peu utilisée provoque le chargement automatique du code de celle-ci en mémoire centrale après déchargement préalable d'une fonction de la mémoire centrale sur le support magnétique. Cette méthode peut être mise en place par deux procédés:

-un procédé de recouvrement (overlays) qui transfère uniquement un nombre limité de fonctions.

-un procédé de chargement dynamique qui opère des transferts plus importants impliquant le programme principal sans ses sous-programmes. Le chargement des fonctions s'opérera par la suite si celles-ci sont appelées. Ce procédé de mémoire virtuelle présente l'inconvénient de ralentir l'exécution par de fréquents échanges entre mémoires. C'est pour cette raison que les exécutifs temps réel ne l'utilisent pas. Cette gestion est normalement

programmée et effectuée par le système de façon automatique et se trouve implantée sur des systèmes aussi bien monotâche, que multitâche et/ou multi-utilisateur.

Cette gestion mémoire virtuelle n'est pas applicable actuellement du fait de l'absence d'unité de stockage secondaire, mais en cas d'implantation d'une telle unité, ce mécanisme d'échange mémoire devrait être intégré et adapté dans le noyau de l'exécutif.

La mémoire étendue

Avec une taille mémoire adressable réduite (1 Mo pour le 68008) et en l'absence de support magnétique, il est possible par un mécanisme logiciel spécialisé d'étendre et de gérer un espace mémoire supplémentaire, dépassant la capacité d'adressage du bus d'adresses du processeur. Cet espace mémoire physique sera divisé en pages de taille fixe, gérées indépendamment les unes des autres par commutation de pages. Ce procédé est celui utilisé pour gérer la mémoire étendue du DOS (mémoire EMS des compatibles IBM, PC-AT).

Représentations physique et logique

Un programme apparaît sous forme modulaire et linéaire en représentation logique. Les modules sont des éléments effectuant certains types d'opérations à partir de fonctions, procédures ou routines développées en langage de programmation.

Lorsqu'un programme est chargé en mémoire centrale pour y être exécuté, il acquiert alors une représentation physique. Cette représentation physique dépend du support matériel.

Les fichiers

Les fichiers sont des représentants intermédiaires entre aspect logique et aspect physique: -les fichiers sont des ensembles de données organisées et disposées en mémoire à des adresses précises, gérées par le système d'exploitation: représentation physique.
-les fichiers constituent une représentation visible par l'utilisateur des données ou des programmes structurés: représentation logique.

Un fichier est un ensemble d'informations homogènes qui sont soit des instructions (code programme exécutable), soit des données définies et organisées (fichiers source programme, fichier binaire exécutable, fichier de données utilisateur...) issues de programmes d'application (fichier texte, images scannérisées).

La mémoire permanente a pour unique fonction de stocker des données et des programmes sous forme de fichiers. Pour être exécutés,lus, modifiés, les fichiers doivent êtres chargés en mémoire centrale. Deux types de manipulation sont possibles:

-les opérations portant sur le contenu du fichier lui-même:la lecture, l'écriture des données en fichier.
-les manipulation de fichiers:la création, la destruction, la copie de fichiers.

La représentation physique du fichier (placement et contexte) est différente selon le support utilisé (mémoire centrale ou mémoire secondaire). La représentation physique du fichier exécutable en mémoire centrale dépend du système d'exploitation (calculs d'adresses par rapport à un emplacement mémoire prédéfini).

La notion de fichier est donc de porté générale. Particulièrement adapté pour décrire le type et le nombre d'informations qu'il renferme, il ne peut donner aucune indication sur son implantation en mémoire et sur l'environnement participant à son utilisation.
Les données d'un fichier texte sont placées dans la zone des données non initialisées (data en langage C). Le code programme d'un fichier exécutable est chargé lui, dans la zone de code de l'espace mémoire alloué à un processus activé (zone TEXT d'un langage évolué). Une zone spéciale est réservée aux constantes du programme (section BSS).

Le fichier est en ce qui me concerne un élément de description trop limité pour exprimer les événements de gestion de la mémoire; et par suite , je lui préfère le terme "d'espace processus " qui sera celui utilisé pour décrire physiquement l'aire mémoire nécessaire à l'exécution du processus, quelle qu'en soit la nature. Cet espace processus est à géométrie variable selon que le processus est résident ou importé, enregistrant ou pas.

V.2: Organisation et gestion physique de la mémoire

La représentation physique porte aussi bien sur l'organisation matérielle que sur la gestion de la mémoire.

L'organisation du support

L'une des premières opérations du système d'exploitation est de définir un espace mémoire divisé en quatre zones nécessaires au stockage de chaque programme à exécuter et au stockage des données utilisées (données de configuration) ou produites par son exécution (zones données et pile). Ces zones de tailles fixes ou variables, sont représentées soit par des adresses de début et de fin de zone, soit par leur adresse de début et leur taille de zone
Une fois cette organisation générale mise en place (fig n°V.1), le fichier programme peut être chargé et exécuté en mémoire centrale.

Un programme en cours d'exécution utilise quatre zones mémoire:

-1) une zone contenant le code binaire exécutable du programme (fichier exécutable).

-2) une zone constituée par les données initialisées (constantes) du programme(fichier exécutable).

-3)une zone remplie au fur et à mesure de l'exécution du programme par des données non initialisées (fichier application).

-4)un espace suffisant pour la pile nécessaire à l'exécution du programme, mais ne faisant pas l'objet d'un fichier.

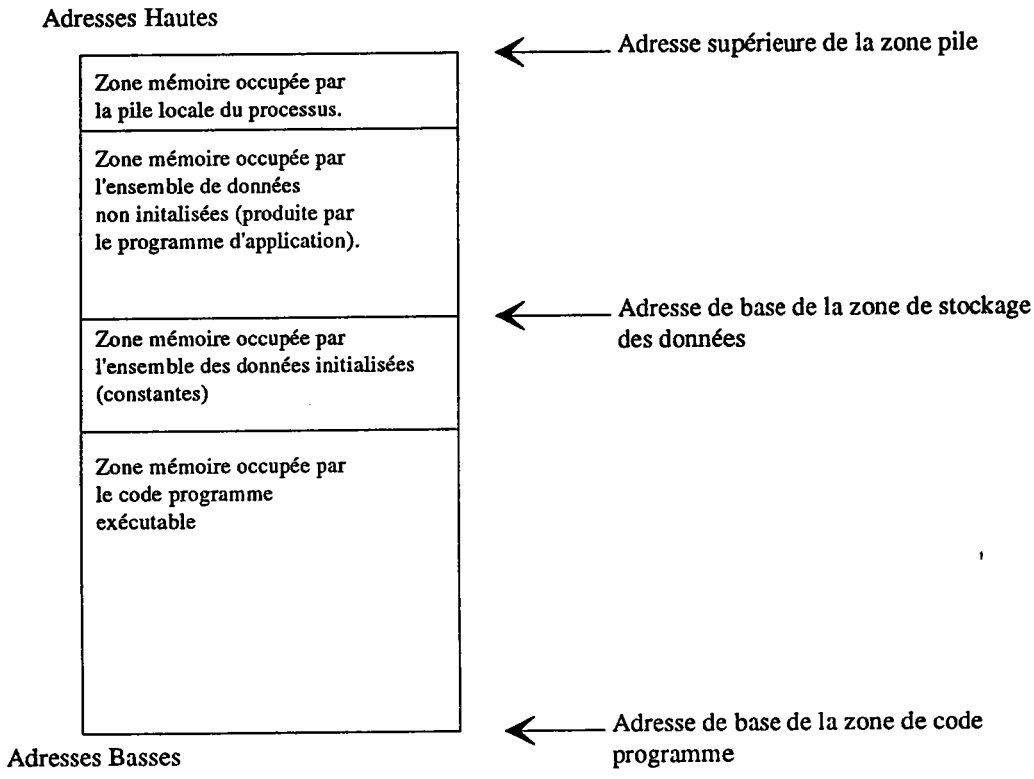


Fig n°V.1: Principe de l'organisation mémoire pour tout processus.

Après la structuration de ces zones pour constituer un espace processus, il est nécessaire de mémoriser sous forme de tableaux cette organisation, puis le système d'exploitation doit garantir l'intégrité de ces zones, en y interdisant les accès non autorisés. Le contenu de certaines zones utilisées comme tampon de communication interprocessus doit être partagé et contrôlé pour éviter les pertes inutiles par duplication.

La gestion mémoire

Dans une organisation donnée, le système d'exploitation attribue certaines zones à un ou plusieurs programmes activés (multitâche) ou à différents utilisateurs (multiutilisateur) par application d'une stratégie d'allocation.

Les opérations d'allocation consistent à attribuer de façon formelle un espace mémoire ou une zone mémoire à un processus par création d'une représentation (un tableau) contenant des adresses (ou des déplacements permettant de calculer ces adresses) de placement et d'accès aux différentes zones que le gestionnaire de mémoire définit au fur et à mesure.

Les opérations de gestion portent:

-sur les mots: lecture,écriture en mémoire sous contrôle de l'exécutif.

-sur les zones et espaces processus: allocation ou libération (totale ou partielle), extension ou diminution, subdivision d'une zone.

-sur l'ensemble de la mémoire subdivisée en espaces, allouée aux différents processus: tests mémoire...

Avec un mécanisme de gestion de mémoire virtuelle, le dernier rôle du système d'exploitation est de fournir à l'utilisateur de façon transparente la possibilité de disposer de tout l'espace mémoire nécessaire en s'affranchissant de toute limitation matérielle de la mémoire centrale par la méthode d'échange ("swapping") de données ou programmes avec la mémoire secondaire.

Deux modèles fondamentaux d'allocation sont disponibles pour gérer la mémoire centrale [réf: BEA]: -allocation contigüe.

-allocation non contigüe.

La plupart des systèmes d'exploitation partitionnent l'espace mémoire en zones elles-même subdivisées en blocs de taille fixe ou variable. Cette division en blocs est soit logicielle, soit matérielle et, dans ce cas elle dépend du microprocesseur.

Représentation physique et familles de processeur

Dans le cadre de la représentation physique, l'organisation et la gestion de la mémoire vont dépendre du type de microprocesseur et de sa structure microprogrammée.

Les microprocesseurs de la famille Intel

Les microprocesseurs de cette famille (8086, 80286, 80386) disposent de deux aires d'adressage mémoire: -pour les périphériques.

-pour la mémoire centrale paginée /segmentée.

Ce dernier espace mémoire adressable est segmenté, en blocs de taille variable mais limités (segment de taille maximale de 64 Ko pour le 8086 et 4 Go pour le 80386 en mode protégé). Un segment est défini par une adresse de base (adresse basse) du segment, sa taille et son droit d'accès (type et accession). Au niveau du microprocesseur 8086, trois jeux de registres de 16 bits sont implantés:

-quatre registres généraux: accumulateur (AX), base (BX), compteur (CX) et données (DX).

-quatre registres pointeurs et index de 16 bits (SP,BP et DI,SI).
-quatre registres de segment qui pointent sur les différents segments accessibles; segment de code (CS), segment de pile (SP), segment de données (DS), segment d'extension (ES).

Le nombre d'adresses pouvant être ainsi générées à partir des 16 bits de chacun de ces registres est de 65536 octets (64 Ko), ce qui définit la taille maximale d'un segment. Ces segments peuvent être disjoints, se recouvrir totalement ou partiellement. Leur nombre n'est pas limité (mais leur taille est réduite en contre partie) et leur affectation est variable.

En considérant le nombre de lignes d'adresse (20 lignes), le processeur peut gérer une mémoire de 2^{216}=1 Mo, divisible en 16 segments de 64 Ko contigüs.

La gestion mémoire de cette famille de processeurs n'est pas la plus simple et la plus souple, et implique une attention particulière dans la manipulation des segments (configuration "far" et "near").

Processeur de la famille Motorola 6800 (16 bits) [réf: VIE]

Les processeurs de la famille Motorola ne connaissent eux qu'un seul espace mémoire configurable à la demande par le concepteur du matériel et du logiciel, pour la mémoire et les périphériques. Cet espace unique n'est ni segmenté ni paginé. On ne retrouve donc pas la structure en registres dédiés des différents segments, du fait de la taille des registres.
La taille du bus de données, pour un 68008 (version 8 bits du 68000) est de 20 lignes autorisant un adressage de 1 Mo, et pour un 68000 ayant 22 lignes d'adresses, 4 Mo (de RAM en mode utilisateur) sont adressables. La taille des registres de 32 bits permet d'adresser logiquement 4 To. Une gestion virtuelle paginée est ainsi favorisée, pour une éventuelle extension mémoire.

La segmentation

C'est un mécanisme inhérent aux processeurs de la famille Intel, dû à la différence de taille entre;
-le registre d'adresse de segment, ne permettant d'accéder qu'à 64 Ko.
-et le bus d'adresses du microprocesseur permettant d'accéder à 1 Mo.
L'adresse transitant sur le bus étant obtenue par sommation du contenu du registre de segment et du registre d'index.

Les processeurs de la famille Motorola, dans la limite définie par la taille de leur bus de données ne connaissent pas ce phénomène de segmentation.

La largeur du bus d'adresse et la taille des registres internes définissent ainsi des limites d'utilisation du système de gestion quel que soit le type de microprocesseur utilisé.

La pagination

C'est un mécanisme de gestion mis en place lorsque l'on désire accéder à une position mémoire située au-delà de la limite définie par la largeur du bus de d'adresse. Ce mécanisme implique, au niveau du bus microprocesseur, un multiplexage des éléments constituant l'adresse appartenant à une autre page que la page courante(ensemble des adresses comprises dans la limite définie par le bus d'adresse). Ce mécanisme de pagination est celui utilisé, pour gérer les extensions mémoire (mémoire EMS).

Pour la 68008, la page courante a une taille de 1 Mo. Par ses registres d'adresses et de compteur programme, de 32 bits l'espace virtuellement adressable est de 2^{32} soit 4096 pages de 1Mo. L'extension mémoire réalisable est extraordinaire mais non utilisée car les temps d'accès deviennent très longs.

La structure du microprocesseur, ainsi que l'espace mémoire physiquement disponible, conditionnent le mécanisme de gestion mémoire. Dans le cadre d'une gestion multiprogrammée il reste à définir une méthode et un modèle de gestion des différents espaces processus, au niveau d'un segment ou d'une page, ainsi que les contraintes rencontrées.

Dans un but de généralisation, le terme de bloc mémoire, remplacera (si cela est possible) les termes de zone et de segment.

Problème de fragmentation

Fragmentation interne de bloc:

Lorsqu'un bloc est alloué l'ensemble des informations n'est pas un multiple entier de la taille d'un bloc. Un programme ou un ensemble de données peut utiliser plusieurs blocs. Un bloc non utilisé entièrement, produit un fragment interne au bloc (fig n°V.2). On considérera qu'il y a fragmentation interne lorsque la perte mémoire atteindra environ 1 Ko.

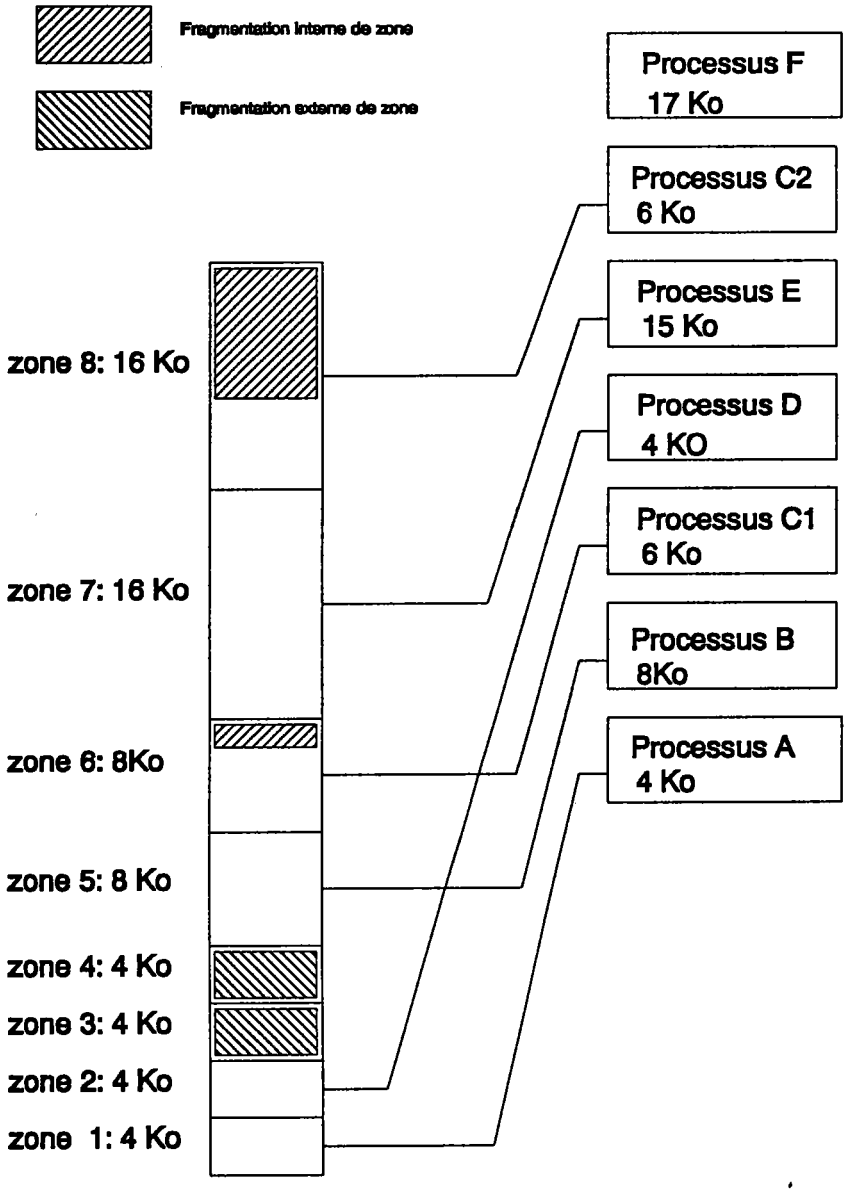


Fig n°V.2: Fragmentations dans le cadre d'un modèle d'allocation contigü des espaces mémoires (partionnement.fixe prédéfini).

Fragmentation externe de bloc:

Considérons un ensemble de données à placer en mémoire centrale et nécessitant un peu plus d'une zone mémoire (zone de taille fixe). Une deuxième zone mémoire est allouée à l'intérieur de laquelle des blocs (de taille fixe) restants libres (n°V.2). Ces blocs constitue un exemple de fragmentation externe de blocs (ou fragmentation interne de zone).

La gestion de la mémoire est souvent décrite à partir des seules opérations d'allocation mémoire et les mécanismes et stratégies envisagés ne traitent que de cette fonction. Cet état de fait peut s'expliquer d'une part par l'utilisation préférentielle du modèle d'allocation non-

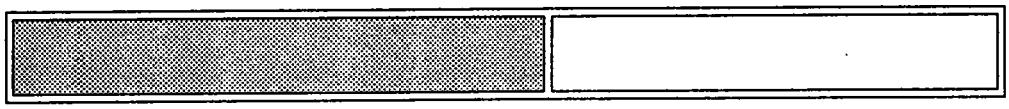
contiguë et d'autre part par l'importance de cette fonction au niveau de laquelle se fait le choix de placement des données en mémoire. Les opérations d'écriture et de lecture des données ne constituent pas l'ossature du modèle de gestion.

Chaque modèle met en oeuvre une structure de représentation, une répartition des fonctionnelles et privilégie des opérations à réaliser très spécifiques.

V.3: Le modèle d'allocation mémoire contigüe

Principe

Le espaces mémoire sont placés à des adresses consécutives. Le champ mémoire global est divisé en espaces d'un seul tenant comprenant un nombre fixe ou variable de zones entières,vides ou allouées. Le système maintient en permanence la liste des espaces libres et occupés (fig n°V.4).



Mémoire allouée à différents processus Mémoire libre (non allouée)

Fig n°V.4: Occupation mémoire en espaces contigüs.

Ce modèle, simple de mise en oeuvre, ne privilégie pas la fonction d'allocation qui est privée de recherche et de choix d'un espace libre. Par contre les opérations de réorganisation, de compactage des données présentes en mémoire sont les plus complexes et constituent le véritable système de gestion de la mémoire prise globalement. C'est un mécanisme intervenant à posteriori, sur l'existant.
Un espace mémoire affecté à un processus n'est pas figé dans le temps. Il peut être déplacé, étendu, réduit, désalloué, et chacune de ces fonctions participant au mécanisme de gestion intervient de façon équilibrée.

Allocation.

Lors d'une allocation mémoire le système prélève une zone libre à l'extrémité des zones occupées, dans l'espace libre restant. Avant toute allocation mémoire, opération qui ne consiste qu'à la définition des adresses de base des différentes zones (code, données, pile), un test est réalisé pour vérifier que l'espace demeuré libre est suffisant.

Désallocation.

Quel que soit le modèle utilisé, cette opération ne réalise que la modification d'un indicateur (en liste des enregistrements ou éléments descripteurs d'allocation mémoire) précisant que l'espace prédéfini est vacant. La désallocation doit s'accompagner de l'opération de compactage avec fusion de zone et translation, sous peine de se trouver confronté à des problèmes de fragmentation externe de zones.

L'extension d'un espace.

C'est l'opération la plus délicate de ce système car pour conserver l'aspect contigu des données appartenant à un même ensemble (et décrit par un seul élément descripteur) il est nécessaire d'opérer des déplacements. Les opérations d'extension d'un espace mémoire peuvent être réalisées de deux façons différentes, soit par déplacement vers le haut de la mémoire centrale des espaces supérieurs à l'espace à étendre, soit par réallocation d'un espace étendu et duplication des données de l'espace initial. Cette dernière solution impose ensuite l'exécution d'une procédure de compactage des données et des espaces. Le détail de ces mécanismes est proposé ultérieurement.

Réduction d'un espace alloué.

Cette opération implique une modification des paramètres d'utilisation de l'espace alloué (adresse de base, pointeur), et un marquage de l'espace devenu vacant (création d'un élément descripteur). Une procédure de compactage intervenant ultérieurement viendra rendre les espaces mémoire contigus.

La libération d'une ou plusieurs zones laisse des zones fragmentées. Le choix d'une partition en zone, simple ou double, fixe ou variable influe sur le phénomène de fragmentation. Cette fragmentation est de plusieurs ordres, interne ou externe aux zones mémoire, et interne ou externe aux espaces, mais dans tous les cas elle est synonyme de perte d'espace mémoire.

Le compactage

C'est un procédé permettant de regrouper des zones libres non adjacentes par déplacement pour constituer des espaces et zones libres contiguës eux mêmes regroupés en un seul espace. Il a le désavantage d'occuper l'unité centrale de façon importante, et diminue ainsi le temps consacré aux autres processus (processus utilisateur). De plus sa gestion est délicate car il nécessaire d'intervenir au niveau de l'adressage des données. Ce procédé est rarement utilisé dans les systèmes d'exploitation, et jamais dans des systèmes temps réel.

Allocation contigüe et systèmes monoprogrammés (mono-utilisateur)

Un seul programme utilisateur est présent en mémoire centrale.

L'organisation et la gestion de la mémoire sont très simples.

Un premier programme utilisateur dispose de toutes les ressources matérielles et logicielles disponibles. Le programme débute et prend fin sur décision de l'utilisateur. L'exécution d'un autre programme détruit toutes représentation en mémoire d'un programme utilisateur précédent. Cette technique est celle utilisée par les systèmes d'exploitation tels que CPM et MS-DOS.

La mémoire RAM y est divisée en deux zones de taille et de disposition variable (fig n°V.5) pour le système d'exploitation (zone système) et pour les programmes d'application utilisateur.(zone utilisateur).

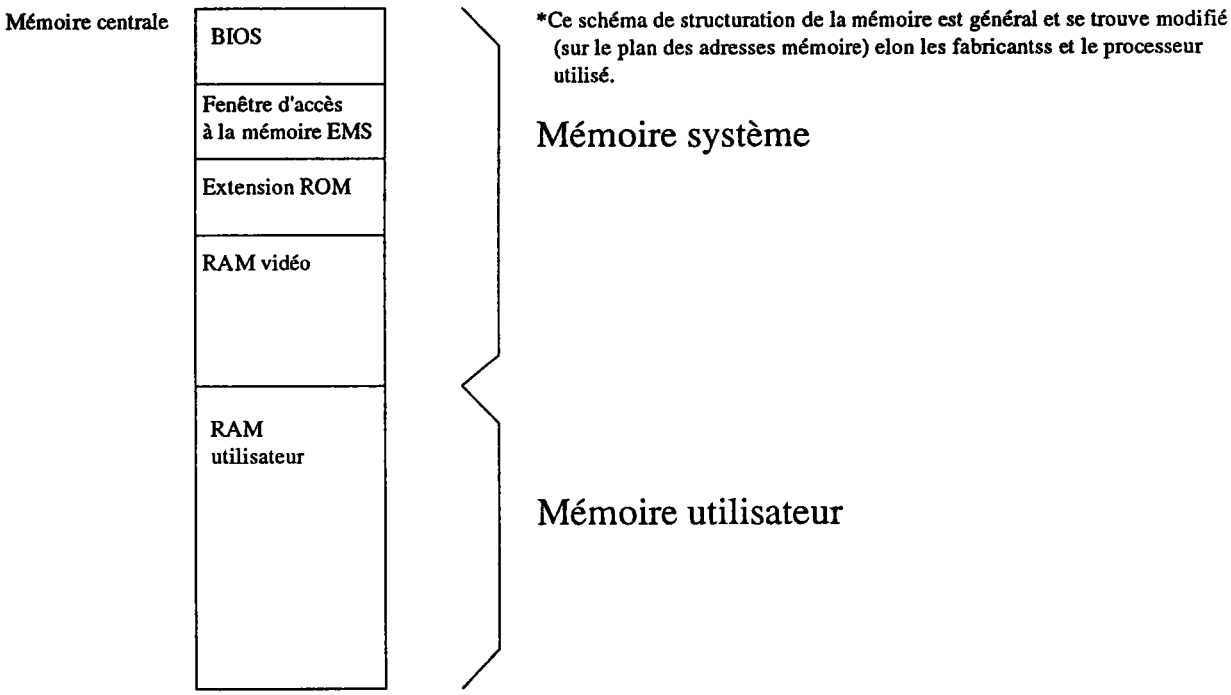


Fig n°V.5: Structuration générale de la mémoire centrale utilisée par MS-DOS.

La protection en écriture réside dans la comparaison entre l'adresse prévue en écriture, et l'adresse barrière. Celle-ci est:

-soit matériellement fixée de façon définitive.

-soit variable en fonction de la taille du programme d'application. L'adresse barrière est fixée en cours d'exécution de l'application.

-soit variable en cours d'exécution d'une application, que ce soit pour la zone système ou pour la zone utilisateur. On charge alors en mémoire centrale les éléments de programme

seulement nécessaires; c'est le procédé de gestion de mémoire virtuelle, ce procédé étant appliqué à la demande sur les fonctions système ou sur les fonctions constituant le programme utilisateur.

Allocation contiguë et systèmes en temps partagé (multi-utilisateur)

Chaque programme à exécuter se trouve en totalité en mémoire secondaire et lors d'un appel à exécution de l'un deux, son code exécutable est chargé en mémoire centrale en zone utilisateur durant une tranche de temps définie.
En quelque sorte on associe dans cette solution une gestion virtuelle de la mémoire sans partage de la zone utilisateur et un mécanisme de commutation d'exécution alterné des processus stockés sur disque. Le mécanisme d'allocation de la mémoire centrale reste identique à celui d'un système monotâche, mais le mécanisme de gestion intègre en plus des fonctions de restauration et de sauvegarde de contexte des processus et les fonctions de gestion de mémoire virtuelle. L'utilisateur peut avoir l'impression d'utiliser un système multitâche, mais ses performances sont très limitées par de fréquents accès au support de masse.

Allocation contiguë et systèmes multiprogrammés

Cette fois il est nécessaire de gérer plusieurs processus présents simultanément en mémoire centrale. Chaque processus occupe une partie de celle-ci, et une gestion de mémoire virtuelle peut être envisagée en complément. Un double système de partition est mis en place: partition de la mémoire centrale et partition de la mémoire de masse. Les mécanismes d'allocation/désallocation/extension/réduction d'aires mémoire peuvent être similaires ou différents entre les deux supports mémoire.

Il est possible de concevoir encore des solutions intermédiaires avec une allocation mémoire principale pour l'exécution courante d'un processus et deux zones mémoire tampon affectées l'une au déchargement du code de l'application précédente, et l'autre au préchargement de la tâche suivante, ce qui implique l'utilisation du mécanisme de transfert DMA (Direct Memory Accès). L'emploi de cette méthode est disproportionné par rapport au flux d'informations.

La coexistence de plusieurs programmes utilisateurs indépendants en mémoire centrale outre le fait qu'elle complique le mécanisme d'allocation/désallocation, extension/réduction

mémoire, introduit le paramètre taille des différentes zones allouées et donc le partitionnement de celle-ci, partionnement en taille fixe ou partitionnement en taille variable.

Le choix d'implantation du code de l'exécutif multiprogrammé en ROM, bien que réducteur de l'espace d'adressage physique, permet une utilisation presque exclusive de la RAM pour les programmes utilisateurs, et offre une protection absolue du code du moniteur contre des recouvrements accidentels d'espace mémoire affectés.

La possibilité offerte par certains processeurs évolués de protéger un espace RAM à utilisation exclusive du système spécialise les différents espaces mémoire physiques et apporte un niveau de sécurité supplémentaire non négligeable. Cette structure RAM système et RAM utilisateur laisse intacte la capacité d'adressage physique maximale (4000000 adresses définissables et 4Mo de RAM superviseur, 4Mo de RAM utilisateur).

Quelque soit la spécialisation de la RAM en différents espaces protégés ou non le problème du choix de la stratégie de gestion, de l'organisation et du partitionnement mémoire reste entier.

Les stratégies d'allocation, réallocation d'espace mémoire

Il n'y a pas de stratégie particulière d'allocation à choisir. On alloue tout ou partie de l'espace libre disponible et le partitionnement a lieu sur décision du concepteur fixe ou variable. Il est possible d'allouer à un nouveau processus toute l'aire mémoire disponible avec réduction ultérieure au risque de bloquer rapidement toute extension des espaces existant préalablement et appartenant à différents processus.
Le partitionnement fixe est simple à mettre en oeuvre, rationalisé et implique si besoin est des extensions mémoire dont la réalisation est rendue complexe par le placement de façon contigüe des espaces mémoire. La segmentation utilisée par le processeurs de la famille Intel s'accommode fort bien du partitionnement fixe. Les pertes mémoire sont conséquence de la fragmentation interne de zones ou de segments.

Partitionnement (fixe ou variable) ou non partionnement?

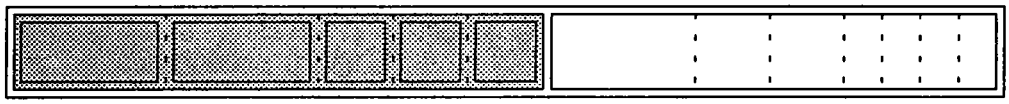
Non partitionnement

Aucun mécanisme de prédécoupage logique n'est mis en place et chaque nouvelle tâche vient occuper une partie de la mémoire restée libre. Les pertes mémoire sont dues seulement à une fragmentation interne de zone. Le gestionnaire de mémoire alloue de la mémoire jusqu'au moment où l'espace libre devient insuffisant. Sans mémoire virtuelle, il est impossible alors d'allouer et d'étendre une zone mémoire, si une zone suffisante n'a pas été libérée au préalable.

Partitionnement fixe

La zone utilisateur est divisée en zones de tailles différentes mais fixes et deux procédés de prédivision sont utilisables:

-1)Avant toute allocation et chargement en mémoire d'un quelconque programme utilisateur, cette zone est prédécoupée en zones de tailles décroissantes suivant une règle de partage définie (figure n°V.6). Ainsi par exemple une zone application de 64 Ko peut être prédivisée en 4 zones de 4Ko, 2 zones de 8 Ko, et 2 zones de 16 Ko.

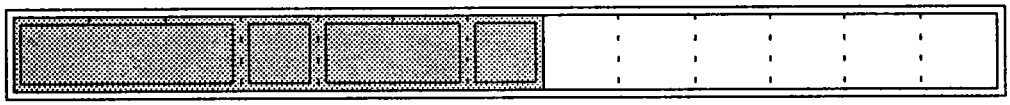


Mémoire libre (non allouée) prédécoupée.

Fig n°V.6: Partitionnement de taille fixe et décroissante, modèle d'allocation contigü.

Cette répartition des tailles de zones au cours des opérations multiples d'allocation, de libération , de réallocation, d'extension et de segmentation de zones mémoire disparaît au fur et à mesure. Cependant elle permet de mettre en oeuvre un mécanisme de gestion relativement simple. En effet pour placer des données en mémoire on recherche la zone libre se rapprochant le plus de la taille du code à charger. L'allocation d'un espace pour charger un programme de 6 Ko est réalisable si une zone libre de 8 Ko est disponible. Il y a malheureusement un problème de fragmentation interne de zone (2Ko non utilisés mais affectés) et de fragmentation externe.

Cette méthode de prédécoupage en taille fixe mais différenciée nous éloigne du modèle de gestion d'espaces contigus.



Mémoire libre (non allouée)

Fig n°V.7: Partionnement de taille fixe, modèle d'allocation contigü.

-2)Le partitionnement est de taille fixe et égal pour tous les espaces définis (fig n°V.7). Les pertes seront des fragments de zone, et son fonctionnement est compatible avec le modèle d'allocation contigüe. L'utilisation d'un mécanisme de récupération d'aires mémoire libres par un compactage et une fusion de zone limite ces pertes, mais consomme beaucoup de temps du microprocesseur.

Partition de taille variable

Les zones sont créées au fur et à mesure des allocations successives et en fonction de la taille des applications à charger et à gérer. Au cours du temps la taille de ces mêmes zones est susceptible d'évoluer selon les besoin, par extension ou réduction de zone. Les programmes utilisateurs importés sont nécessairement relogeables.

Le gestionnaire de mémoire travaille à partir d'une table qui cartographie l'ensemble des zones libres et occupées. Cette table contient pour chaque zone les informations qui la concerne, à savoir: -la taille de la zone, -l'adresse limite inférieure de la zone, -l'identificateur de contenu de la zone. Les zones libres peuvent être référencées dans une autre structure, plus adaptée à une gestion dynamique. Cette structure peut prendre la forme d'une liste circulaire doublement chaînée, pour faciliter et accélérer le parcours de la recherche, et les éventuelles modifications.

L'un des avantages immédiats d'un partitionnement variable est en principe l'élimination de la fragmentation interne, par ajustement sur mesure de la surface mémoire nécessaire. La fragmentation externe constitue la principale perte de mémoire. Le compactage n'est la solution de ce problème que dans le cas d'une gestion dynamique permettant le déplacement et la réimplantation des programmes en mémoire centrale.

La gestion de quelques systèmes

Le système pour mini-ordinateur VAX-VMS multi-utilisateur, utilise différentes structures et mode de partitionnement, pour allouer dynamiquement la mémoire. D'une part il gère par une liste chaînée les zones libres de tailles variables, et la stratégie d'allocation de celles-ci repose sur l'allocation de "la première zone libre". La fusion de zones adjacentes s'effectue automatiquement. Parallèlement, et par une autre liste doublement chaînée, VAX-VMS gère aussi des zones libres et de taille fixe. Cette solution permet d'accélérer l'allocation des zones, pour des fichiers de données, ou de programmes les plus souvent utilisés et de tailles connues (statistiques).

V.4: Le modèle d'allocation non-contiguë:

Principe

Pour être chargé en mémoire un fichier est fractionné, et les informations qu'il contient sont réparties en plusieurs zones non contigües (fig n°V.8). Les opérations de compactage et de fusion de zones ne sont plus utilisées, mais ceci implique de gérer une plus

grande masse d'informations sur la cartographie mémoire. Les différents fragments d'un fichier doivent être au moment adéquat reliés soit par chaînage, soit par indexation.

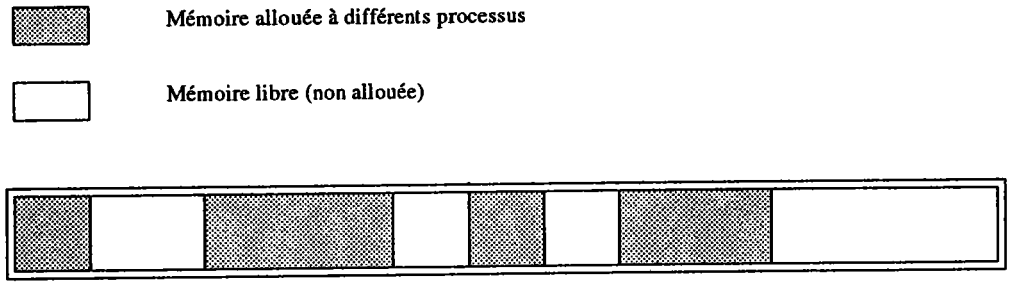


Fig n°V.8: Occupation mémoire en espaces non-contigüs.

Le chaînage

Chaque bloc mémoire affecté à un fichier particulier est affecté d'un numéro et la liste des fichiers présents en mémoire contient les numéros du premier et du dernier bloc utilisé pour ce fichier. Cette représentation est facilement modifiable par modification des numéros de blocs extrêmes référencés à la suite d'adjonction ou de réduction de blocs mémoire, donc de la taille du fichier.

Cette représentation présente différents inconvénients parmi lesquels un accès aux données uniquement séquentiel. Pour retrouver un bloc particulier il faut parcourir toute la liste des fichiers et des blocs mémoire. D'autre part, lors de la rupture de la représentation chaînée, il devient très difficile de retrouver l'accès aux informations contenues dans les blocs suivants. Enfin pour des opérations de réduction ou d'extension de fichier et donc du nombre de blocs, il est nécessaire à chacune de ces opérations d'effectuer une renumérotation des blocs affectés à un fichier.

L'indexation

On conserve dans la liste des fichiers présents et pour chacun de ceux-ci, tous les numéros de blocs utilisés. L'intérêt de cette représentation est d'accéder directement à un bloc, en consultant la liste contenant tous les numéros de blocs utilisés.Par contre ceci exige une liste très grande.

Cette méthode d'allocation est particulièrement adaptée à la gestion mémoire des unités de disque dur ou de disquette, car elle permet l'accès aléatoire aux données. Cette souplesse d'utilisation est le principal intérêt de ces supports de mémoire secondaire.

Dans ce modèle la gestion mémoire consiste essentiellement à réaliser les opérations d'allocation d'espace mémoire pour les différents processus. Son importance réside dans l'application d'un algorithme de choix pour définir l'implantation d'un espace processus en mémoire. C'est un mécanisme intervenant a-priori et dont le rôle est de gérer des "trous", c'est-à-dire des espaces libres, disséminés, et de tailles différentes.

Les autres fonctions n'interviennent que pour réaliser des opérations ponctuelles. Ainsi, la réduction de taille d'un espace mémoire se traduit par la création d'un nouvel élément (portant un indicateur de désallocation) dans la liste de cartographie de la mémoire centrale. L'extension n'est qu'une allocation mémoire pour un processus et un enregistrement existant.

Différentes méthodes de choix d'une zone libre sont envisageables lorsqu'un processus passe de l'état endormi à l'état actif. On parle aussi bien de stratégie d'allocation que de stratégie de placement. Pour choisir la stratégie adaptée on considère deux critères:

-la rapidité de choix d'un espace mémoire.

-les pertes d'espace mémoire par fragmentation (notamment par fragmentation externe).

Les stratégies d'allocation , et réallocation d'espace mémoire les plus couramment utilisées

-de la première zone libre ("first fit "):

Elle consiste à parcourir la liste des zones et à prendre la première ayant une taille suffisante. On optimise la rapidité de choix de l'espace, mais au prix d'une fragmentation éventuellement plus grande (fig n°V.9).

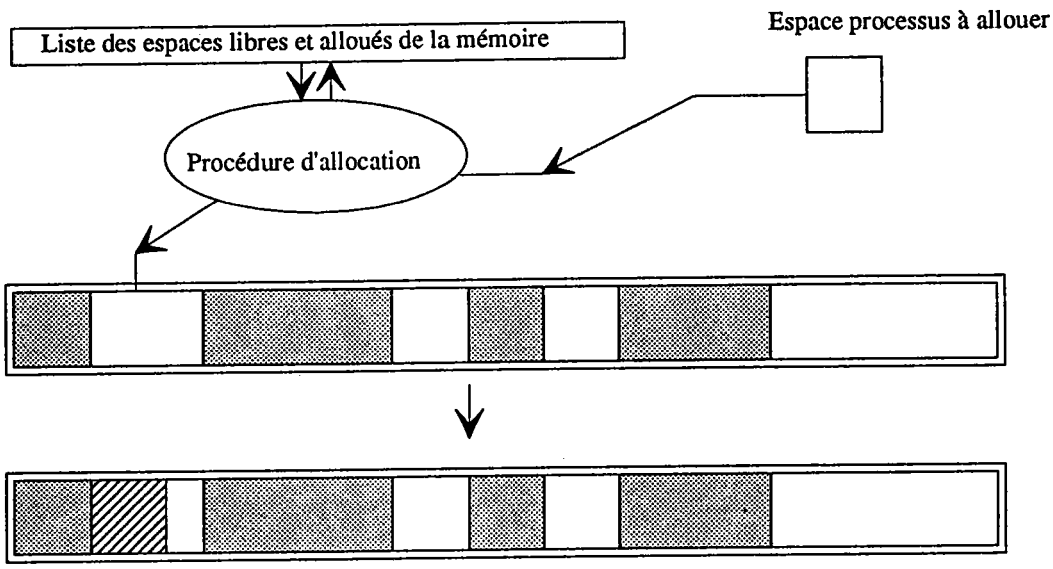


Fig n°V.9: Stratégie d'allocation de la premiére zone libre.

-de meilleur ajustement ("best fit"):

On sélectionne dans la liste des zones libres celle qui correspondra le mieux du point de vue de la taille à celle du fichier à charger, sans y être inférieure, ce qui revient à choisir la plus petite des zones de taille suffisante (fig n°V.10). On limite ainsi l'effet de fragmentation. Cette méthode est plus lente de mise en oeuvre car il faut examiner l'ensemble de la liste et effectuer un certain nombre de comparaisons relatives ou des réarrangements périodiques de la liste. Le réarrangement de la liste consiste à trier et à classer les zones selon un ordre croissant de taille des zones. En revanche l'occupation mémoire est optimisée.

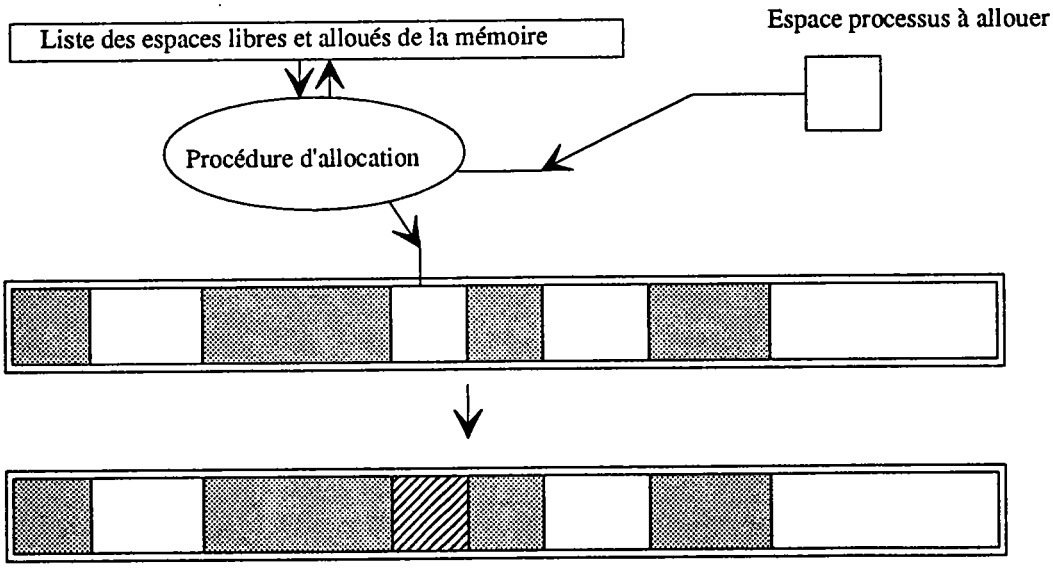


Fig n°V.10: Stratégie d'allocation du meilleur ajustement.

-de plus grand résidu ("worst fit"):

On décide pour simplifier d'utiliser la plus grande zone mémoire. L'opération d'allocation est rapide mais coûteuse en espace mémoire, ainsi on la complète par une opération de segmentation de la zone mémoire. On retire de la zone mémoire initiale l'excédent de mémoire non utilisée. Ainsi à partir d'une grande zone on crée une zone sur mesure pour le processus, puis une zone de taille variable à partir de la mémoire restante. On assiste alors à un morcellement des zones mémoire et donc à un effet de fragmentation (fragmentation externe de zone) (fig n°V.11). Pour limiter les pertes il faut alors appliquer la méthode de compactage, ce qui fait perdre une partie de son intérêt à cette stratégie.

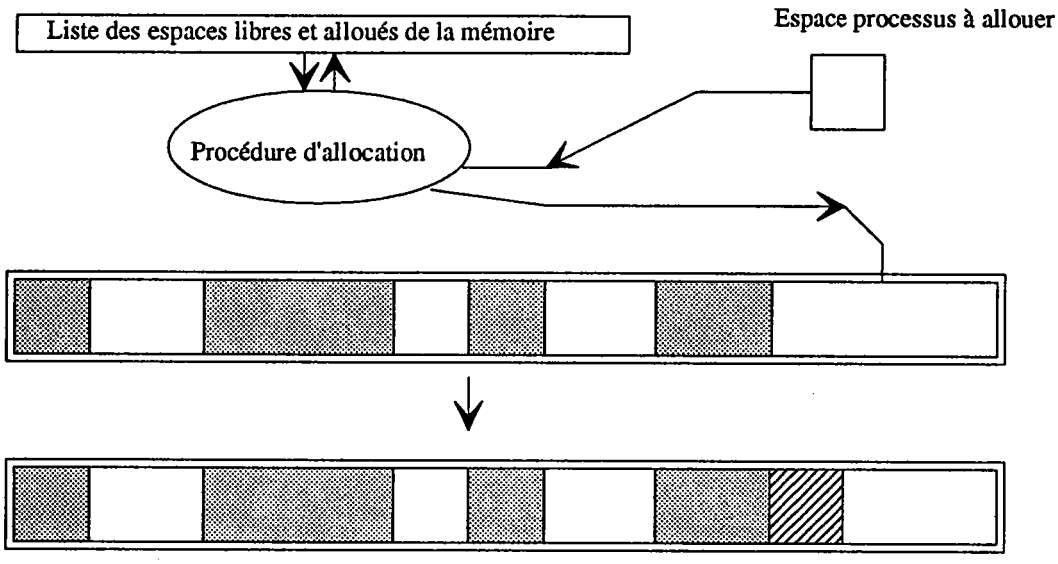


Fig n°V.11: Stratégie d'allocation du plus grand résidu.

Le modèle de l'allocation non contiguë en mémoire centrale est le modèle utilisé dans les systèmes d'exploitation plus récents. Un fichier peut être physiquement chargé à des adresses dispersées en mémoire centrale. Ceci implique alors de la part du système:

-de gérer des informations supplémentaires pour retrouver les différentes parties d'un même fichier.

-de gérer une nouvelle organisation mémoire sous forme de segments ou sous forme de pages, ce qui se traduit, de façon pratique, par une dissociation complète entre, d'une part les adresses référencées par un processus en cours d'exécution (pointeurs,étiquettes et symboles) et d'autre part l'ensemble des adresses physiques disponibles en mémoire centrale. Cependant pour réaliser l'exécution d'entités indépendantes que constituent les processus, un lien unique

et minimal doit être instauré pour créer la correspondance entre espace logique et espace physique seulement lors de l'exécution d'un processus. Cette correspondance est matérialisée par un nombre limité d'informations (de caractère logique), placées dans une table. La lecture des informations relatives à l'allocation mémoire d'un processus ne donne pas explicitement les adresses de pointage d'accès aux différentes zones du processus (pile, données,programme).

Organisation matérielle des systèmes paginés ou segmentés

Dans le cadre d'une gestion dynamique non-contiguë, se retrouve le caractère fixe ou variable de la taille des zones. Ces deux paramètres d'organisation apportent chacun leurs avantages et inconvénients. Pour un système géré de façon non-contiguë, on utilise une représentation différente, le terme de page correspondant à la définition de zones mémoire de même taille, et le terme de segment recouvrant des zones mémoire de taille variable.

Un système de gestion de zones non-contiguës, peut utiliser soit l'une ou l'autre approche du partitionnement, soit même utiliser les deux conjointement, de façon imbriquée (une page étant subdivisée en segment par exemple), ou de façon parallèle (différentes pages et segments coexistent en mémoire).

Conversion

Quel que soit le partitionnement adopté, en pages ou en segments, pour réaliser le lien provisoire mais nécessaire lors de l'exécution entre les représentations physique et logique d'un processus, on réalise un ensemble d'opérations de conversion entre les adresses logiques et les adresses physiques. Le mécanisme de conversion se doit de fonctionner dans les deux sens, lors du calcul de l'adresse physique en phase de lancement ou de relancement d'exécution du programme, puis lors de la phase de sauvegarde des pointeurs d'accès aux codes exécutables, aux données, à la pile, en déduisant les adresses logiques à partir des adresses physiques.

Chaque processus dispose d'une table de blocs référençant tous les numéros de bloc (page ou segment) du fichier exécutable (selon un ordre croissant) et les valeurs de déplacements de ceux-ci. La table comporte l'adresse physique de base du 1er bloc. L'adresse physique d'un mot dans un page ou un segment (un bloc) est alors calculée par sommation de l'adresse de base du premier bloc et du déplacement du bloc contenant ce mot.

Les deux types de partitionnement (pagination, segmentation) envisagés portent sur la représentation logique et s'envisagent à partir de la représentation et le partage physique de la mémoire (partage physique en boîtes de tailles fixes)

La pagination

Chaque processus dispose d'un espace logique constitué d'un nombre de pages référencées dans une table liée au processus. Ces pages (unités logiques) viennent s'intégrer à la structure physique fixe de la mémoire. La taille d'une page étant égale à la taille d'une boîte, l'adresse logique d'une donnée est obtenue par le numéro de la page et la taille de la page additionnée du déplacement dans la page.

AL=np*tp + dep AL: adresse logique

np: numéro de page

tp: taille d'une page

dep: déplacement dans une page

A chaque page logique correspond une adresse physique de la boîte associée. np <-> APP

AP=APP+dep

Ces informations sont stockées dans une table, en mémoire centrale.

Cette méthode implique des calculs complexes, et s'accompagne d'un effet de fragmentation physique externe. Souvent cette méthode est utilisée pour les processeurs de la famille Intel disposant d'une mémoire cache de type associative.

V.5: LA GESTION DYNAMIQUE DE LA MEMOIRE CENTRALE DU TERMINAL

La mémoire embarquée de l'équipement

Les contraintes d'utilisation et de conception

De par sa structure matérielle ce terminal ressemble à un microordinateur personnel. Son fonctionnement régi par le logiciel d'exploitation doit remplir (nous l'avons déjà vu) des fonctions spécifiques en tant qu'un automate et en tant qu'outil de communication.

Avec une taille mémoire limitée, et une configuration mémoire non optimale (ROM et RAM superviseur) il a fallu concevoir une gestion de ce périphérique en mode multitâche, constituant le deuxième grand mécanisme à mettre en place au sein de l'exécutif. D'une part la mémoire RAM, doit contenir les données reçues en provenance du port série dans un tampon de réception, dont le traitement doit conduire à un stockage définitif selon le service (messages inter-terminaux, messages d'urgence). On y trouve d'autre part, les variables et paramètres de fonctionnement des processus de services et des informations produites par ceux-ci.

Toutes ces informations doivent êtres structurées, agencées de telle sorte que le moins d'espace mémoire, ne soit perdu. Il faut enfin, qu'en cas de dysfonctionnement, les défauts ou dégâts constatés soient au moins en partie réparables, par une procédure de restauration. Cette restauration ne peut intervenir que si un niveau de redondance suffisant est instauré entre les différentes structure de description de processus.

Occupation de la mémoire

A partir de la configuration mémoire initiale, tous les codes programme constituant l'exécutif, l'interpréteur de commande et les processus utilisateurs, ont été chargées dans l'unique ROM de l'équipement (fig n°II.6). L'indépendance de ces entités programmées n'en est pas affectée, mais un certain niveau d'évolutivité matériel n'est plus assuré.

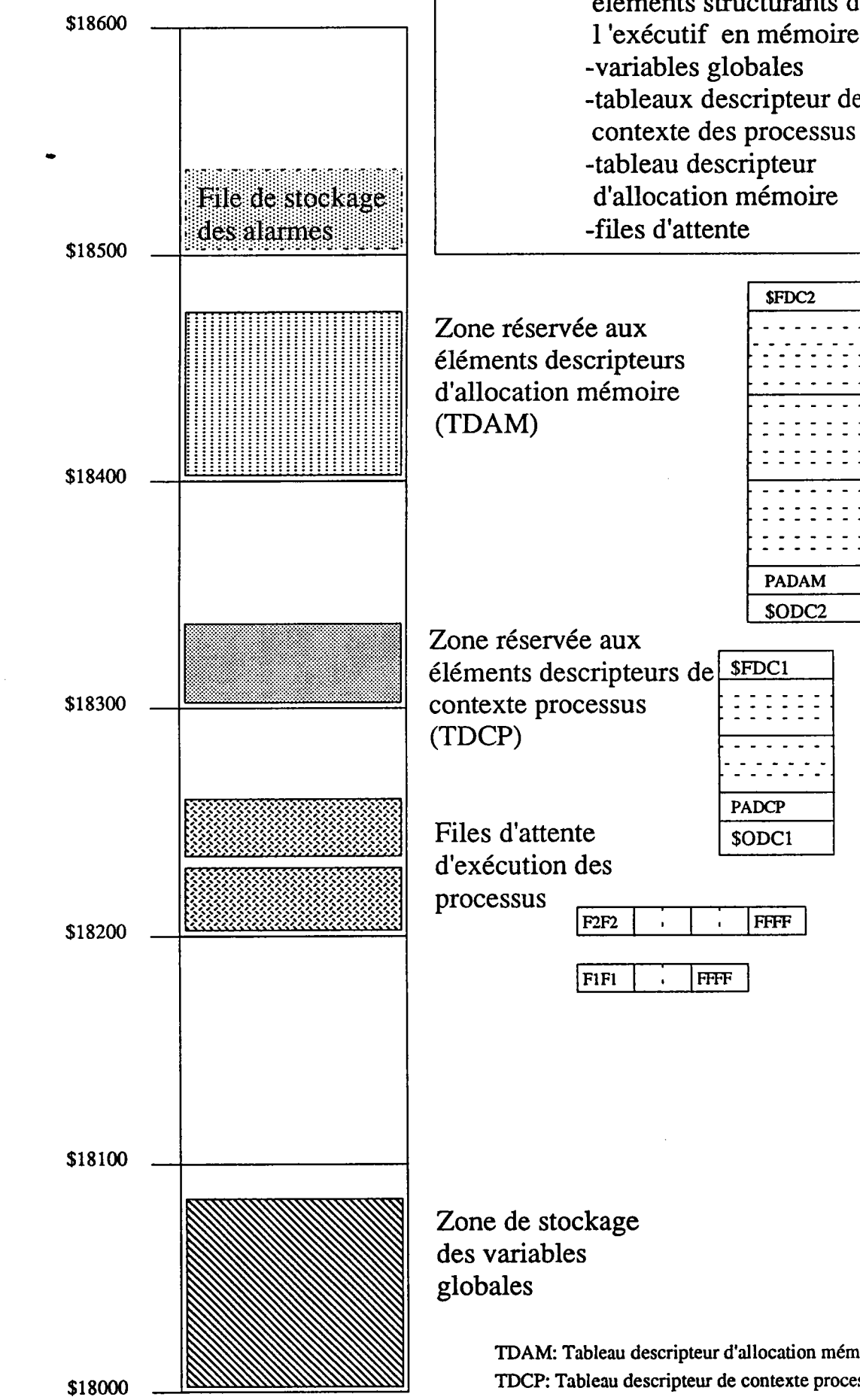
La RAM est utilisée pour contenir les différents tableaux, variables et éléments structurés de l'exécutif, qui sont installés en phase de démarrage de l'équipement (fig n°V.12). Une partie seulement de la RAM est utilisée, pour l'exécution des processus.

Une configuration matérielle, en trois ou quatre blocs mémoire serait nécessaire, pour assurer un fonctionnement à long terme sans risques majeurs (par séparation des données de différents types et des codes exécutables) pour le logiciel d'exploitation global.

-une ROM en espace superviseur pour l'exécutif.
-une ROM en espace utilisateur pour les applications (ces deux ensembles de programmes peuvent êtres chargés dans la même ROM en mode superviseur).
-une RAM système pour contenir les variables et la pile système.
-une RAM utilisateur affectée à l'exécution des seuls programme d'application.

L'espace mémoire superviseur est un espace protégé correspondant à un mode spécifique du microprocesseur. Dans ce mode le microprocesseur peut utiliser un jeu d'instruction étendu et accéder à la totalité de la mémoire (les zones ROMs et RAMs). Par contre en mode utilisateur le processeur ne peut travailler qu'avec les seules zones mémoire utilisateur (les zones superviseur ne lui étant pas accessibles). Le choix du mode de fonctionnement du processeur est réalisé par programmation en registre d'état du bit superviseur/utilisateur. Le microprocesseur passe automatiquement en mode superviseur lors d'une interruption matérielle ou de toute exception.

Fig n°V.12: Répartition des différents éléments structurants de l 'exécutif en mémoire:
-variables globales
-tableaux descripteur de contexte des processus
-tableau descripteur d'allocation mémoire
-files d'attente



TDAM: Tableau descripteur d'allocation mémoire
TDCP: Tableau descripteur de contexte processus

Les espaces mémoire concernés par le gestionnaire

La RAM en mode utilisateur est la seule concernée par le mécanisme de gestion mémoire, puisque son rôle est de gérer uniquement l'espace d'exécution des processus.

Le niveau d'occupation est évolutif, en fonction du nombre d'espace processus ou d'enregistrement présent, et en fonction de leur type. Rappelons que j'ai défini huit configurations de processus correspondant à des niveaux d'occupation mémoire différentes:

-processus résidents: ->enregistrant:2 zones mémoire nécessaires
zone données non initialisées.
zone pile locale.
->non enregistrant:1 zone mémoire utilisée
zone pile locale.

-processus importés: ->enregistrant:3zones mémoire nécessaires
zone code programme+ données initialisées.
zone données non initialisées.
zone pile locale.
->non enregistrant:2 zones mémoire
zone code programme+données initialisées.
zone pile locale.

Ces données qui devraient résider en RAM utilisateur, sont gérées par un ensemble de fonctions implantées en ROM superviseur, telles que les fonctions d'écriture et de lecture en RAM (en zones de données non initialisées).

Les orientations de gestion mémoire: allocation contigüe, dynamique,et compacte.

Adopter une gestion dynamique de la mémoire (RAM), c'est prévoir et réaliser des opérations de translation (et recalcul des adresses relatives), d'allocation, d'extension, possible sur tous les processus présents, tout en conservant et à tout moment une image de l'occupation mémoire [ref:]. Elle ne peut s'effectuer qu'à partir de la définition d'une stratégie globale d'allocation, et du choix de la représentation et paramètrisation des processus ainsi que de l'implantation.

La pratique courante

Il est difficile de faire un parallèle entre la gestion mémoire d'un terminal intelligent, et celle d'un micro-ordinateur personnel ou d'un système informatique plus puissant mettant

en oeuvre un système d'exploitation et des unités de stockage de grandes contenance (disque dur, enregistreur à bande magnétique). En effet sur ces systèmes à dispositif de mémoire de masse, le manque de mémoire centrale peut être pallié par une extension mémoire préalable, ou par échanges de contenu entre la mémoire centrale et le disque dur par exemple. Les données et, programmes déjà utilisées et donc devenues inutiles sont chargées sur le disque dur , et remplacées immédiatement par des données en provenance de ce moyen de stockage. On ne se préoccupe pas de savoir comment s'opère la gestion interne de la mémoire, mais seulement de la présentation des données en mémoire sous forme de fichier et de ces échanges entre mémoire vive et mémoire de masse en cours d'exécution du programme.

De du fait de cette pratique de mémoire virtuelle, le modèle d'allocation utilisé est non-contigüe, car on cherche à privilégier la paramètre de rapidité des opération mémoire. Quelle que soit la politique de placement des données, le gestionnaire mémoire gère des "trous", c'est à dire des espaces libres de tailles différents et disséminés.

La plupart des systèmes d'exploitation et notamment les plus récents utilisent un stockage des données non contigü associé à un mécanisme de gestion dynamique, et d'une structure de représentation sous forme de liste (fig n°V.13). Cette représentation est très pratique car elle permet en peu d'instructions de désallouer un ensemble de blocs mémoire, et elle regroupe dans cette liste tous les blocs mémoire disséminés qui constituent un fichier. Cette structure est aussi celle utilisée dans la gestion du disque dur et disquette.

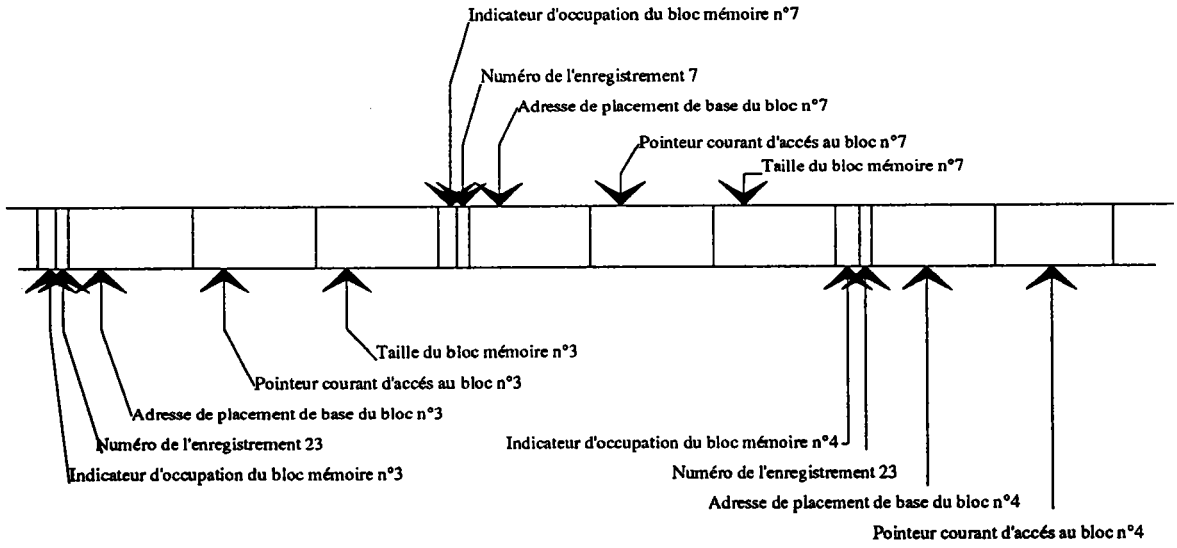


Fig n°V.13: Cartographie mémoire réprésentée sous forme de liste.

Le tableau descripteur d'allocation mémoire

Structuration d'un élément descripteur d'allocation mémoire

A tout moment, que cela soit pour allouer un espace mémoire à un nouveau processus, étendre la mémoire d'un autre en cours d'exécution, libérer totalement ou partiellement l'espace mémoire occupé par un programme, il est nécessaire de se référer à un tableau contenant les renseignements importants de tous les enregistrements présents en mémoire; c'est le tableau descripteur d'allocation mémoire (TDAM).

Tout processus a au moins un enregistrement en mémoire et donc au moins un élément descripteur d'allocation mémoire structuré en six mots longs de 32 bits (fig n°V.14):

-1er mot long: il renferme les informations générales à caractère dynamique sur l'enregistrement et le processus, tels que: numéro du processus, numéro d'enregistrement, nombre de blocs de 10 octets constituant la pile, commande de zone à désallouer de façon automatique, type et nature du processus, état du processus, et enfin le nombre de tranches de temps d'exécution utilisées par le processus (fig n°V.15).

-2éme mot long: il est utilisés pour la sécurité des accès aux différents espaces mémoire alloués et concernant principalement la RAM utilisateur (clé d'accès, et indicateur lecture/écriture autorisée).

-3éme mot long: c'est une adresse de base RAM

-4éme mot long: il contient l'adresse courante de la prochaine instruction à exécuter. Cette adresse est rafraîchie à chaque interruption ou suspension d'exécution de programme.

-5éme mot long:structuré en deux mots de 16 bits affectés au pointeur courant d'accès à la zone données, et au pointeur de base fixe de cette même zone données. Ces deux valeurs figurent ici comme déplacement par rapport à l'adresse de base RAM. Un déplacement codé sur 16 bits permettant de couvrir 64 Ko de mémoire (taille mémoire actuelle disponible 32Ko).

-6éme mot long:lui aussi organisé en deux mots représentant des valeurs de déplacement pour le calcul des adresses d'accès de base et accès courant de la zone pile. Cette structure et cette représentation sous forme de déplacement a été choisie pour éviter une trop grande consommation d'espace mémoire par le descripteur (8 mots longs=32 octets par élément descripteur d'allocation mémoire, par une structure de description classique).

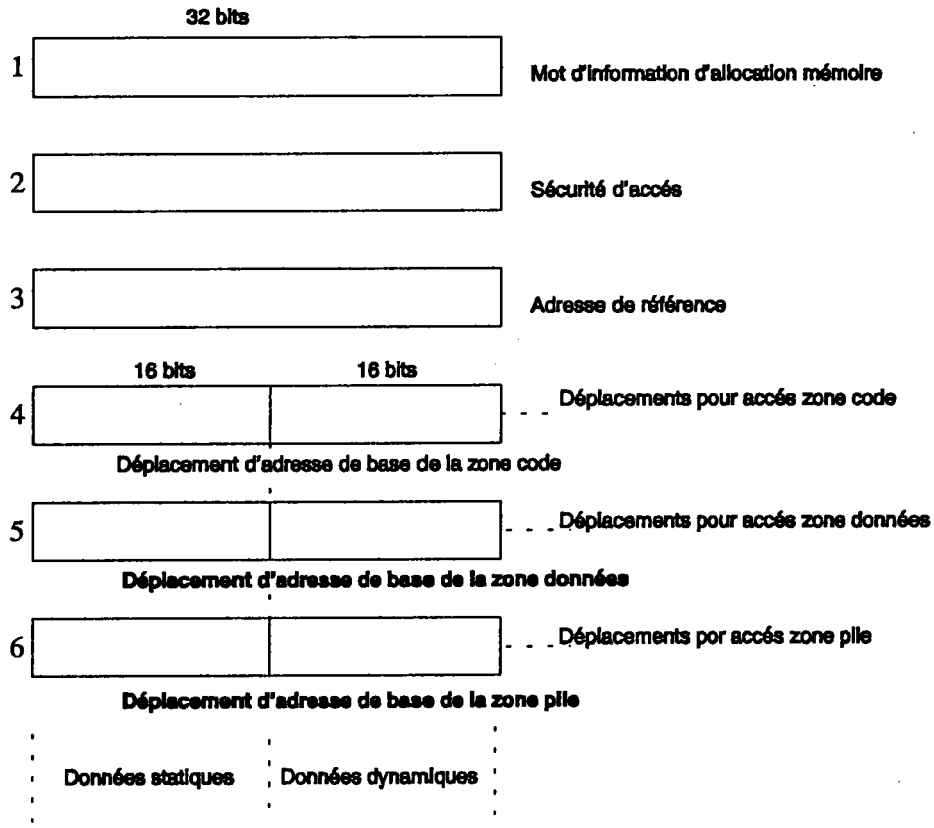


Fig n°V.14: Structuration d'un élément descripteur d'allocation mémoire.

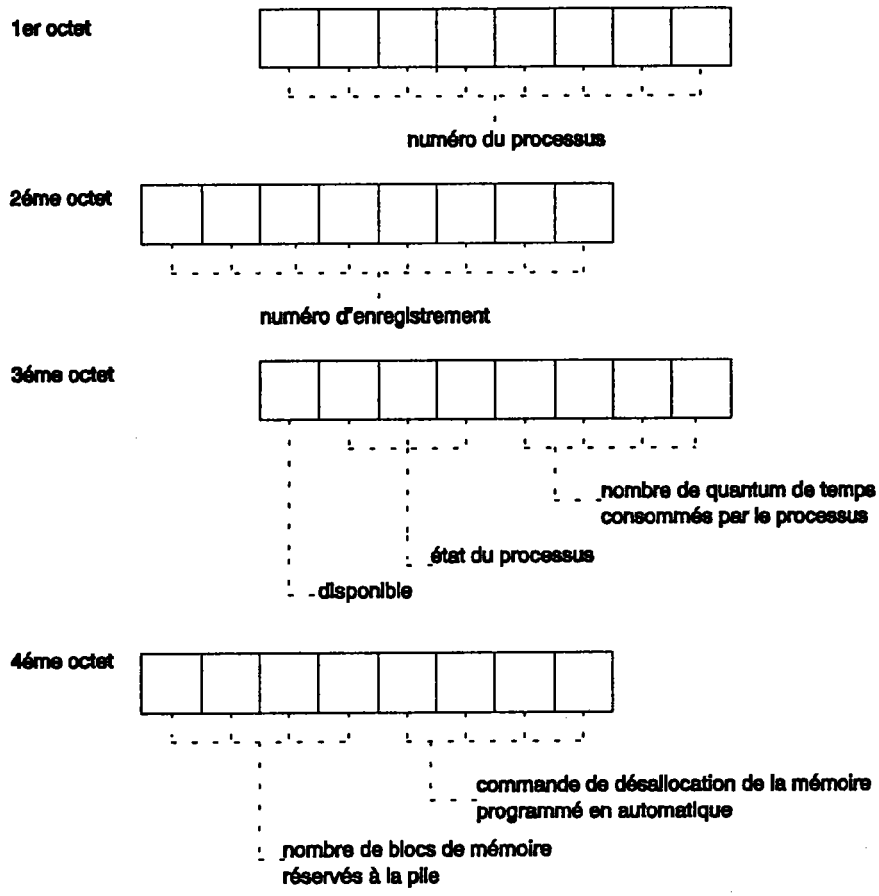


Fig n°V.15 :Structure détaillée du premier mot long (mot d'information) d'un élément descripteur d'allocation mémoire (paramètres dynamiques).

Les informations portant sur le processus

L'état du processus (3 bits): est une information évolutive en fonction des événement internes ou externes.

Le processus peut se trouver dans l'un des cinq état suivants (fig n°V.16):

-endormis: aucune ressource matérielle n'est affectée à ce processus pour permettre son exécution.

-activable: le processus dispose des espaces mémoire nécessaire à son exécution, mais ne s'exécute pas encore. Cet état correspond à un processus en début d'exécution ou de réexécution. Les autres ressources ne lui sont pas spécialement affectées.

-actif: le processus s'exécute effectivement et dispose de toutes les ressources matérielle (à l'exception des interfaces non partageables déjà affectées) et notamment du processeur.

-interrompu: c'est l'état du processus à la suite d'une interruption horloge système. Ce processus étant rechargé en fin de file. Son contexte processus est sauvegardé en tableau descripteur. Son exécution reprendra ultérieurement à l'endroit même où il a été arrêté dans son exécution.

-suspendu: est l'état caractérisant un processus arrêté en attente de la survenance d'un événement.

Le nombre de quantum de temps consommés par le processus: cette information n'est utilisable que par le module de gestion multitâche (ordonnanceur). Un processus dispose initialement d'un nombre limité de quantum de temps pour s'exécuter. Ce nombre de quantum est fixée actuellement à quatre unités. Un processus long , dépassant ce seuil, se verra pénalisé en laissant passer une fois son tour, puis se verra allouer deux nouvelles unités de données.

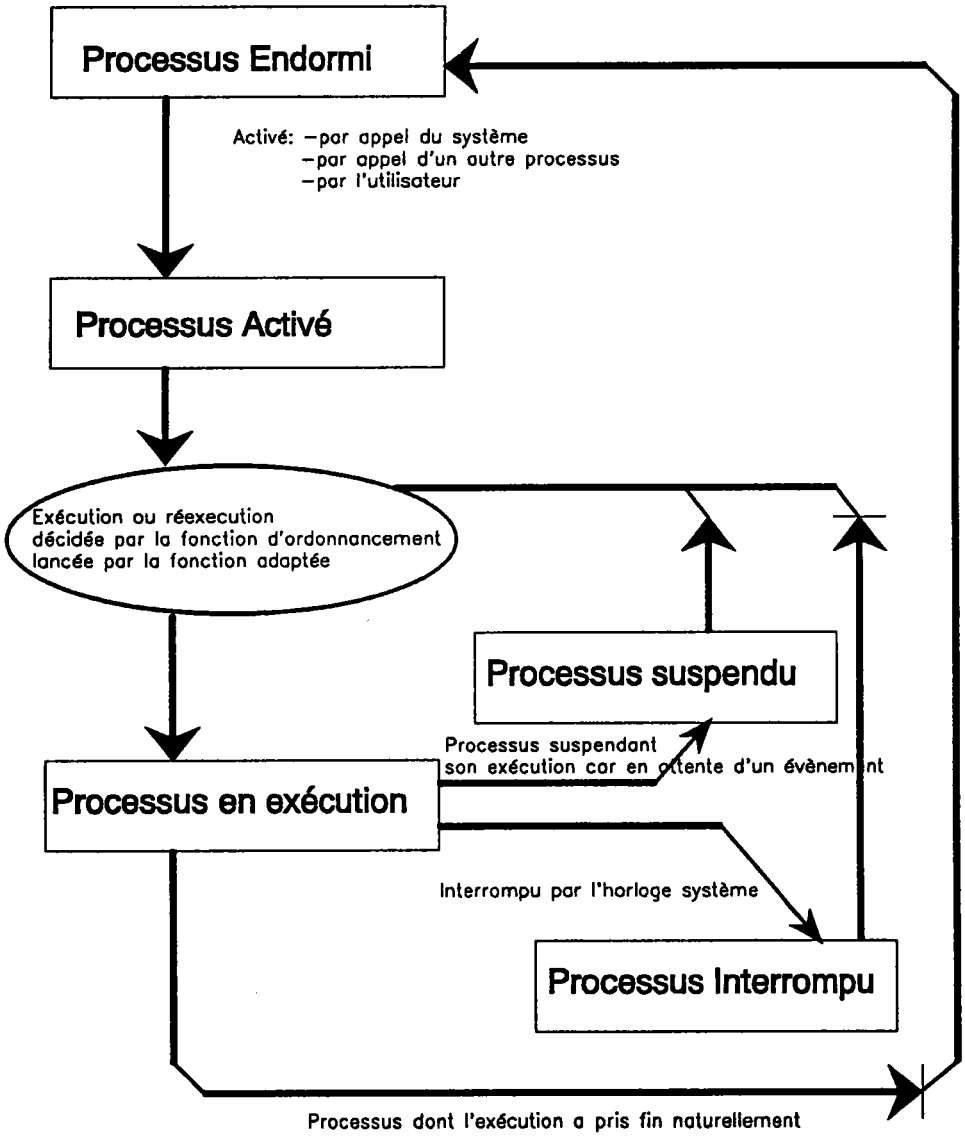


Fig n°V.16: Evolution de l'état d'un processus.

Deux tableaux descripteurs; les raisons

La description globale de l'exécutif montre que celui-ci utilise deux tables principales affectées aux deux mécanismes de base du moniteur. On aurait pu concevoir de ne travailler qu'avec une seule table regroupant tous les éléments nécessaires à chaque processus; regroupant aussi bien les informations constantes sur le processus, que les données dynamiques de gestion du processus et de sa mémoire. Cette solution n'a pas été retenue, pour raison de sécurité, et pour rendre indépendant (modularité) le mécanisme de gestion de la mémoire envisagé. Au niveau de la sécurité, les deux tableaux descripteurs étant implantés à des endroits différents ,une altération des données descriptives ne concerne alors que l'un des descripteur. Les dégâts seront donc relativement limités d'autant plus qu'il existe un certain niveau de redondance entre les descripteurs. L'élaboration d'un mécanisme de sécurité du système est possible.

Un processus n'est reconnu du système que si celui-ci dispose d'un élément de contexte de processus.

L'exécution d'un processus, est marquée par l'existence d'un élément descripteur d'allocation mémoire. Cet élément contient les éléments nécessaires à la gestion mémoire du processus, mais aussi un certain nombre de paramètres évolutifs propres à l'exécution ordonnée courante du processus.

La perte d'informations d'un élément descripteur d'allocation mémoire se traduira par: la réinitialisation d'exécution du processus. L'élément descripteur en cause sera reconstruit à partir des informations de base de l'élément descripteur de contexte et des résultat de la fonction de restauration des espaces processus. Une nouvelle allocation mémoire sera effectuée, et le processus sera réexécuté par le début.

L'altération d'informations contenues dans un élément descripteur de contexte, pourra être compensée par le chargement de certains paramètres présents dans l'élément descripteur d'allocation mémoire. Les éléments manquant seront évalués par défaut selon un cadre restrictif.

L'allocation mémoire automatique différenciée

Le modèle d'allocation contigüe

Après une période critique durant laquelle les programmes de plus en plus perfectionnés faisaient l'objet d'une optimisation forcenée du fait de la limitation de la taille mémoire gérable par les systèmes d'exploitation et du coût élevé des composants mémoire, ces deux obstacles sont aujourd'hui levés. Les logiciels d'application n'ont pas cessés d'être de plus en plus gourmands en mémoire; une mémoire centrale de 1 à 2 méga octets (Mo) est devenue chose courante, et disposer de 8 Mo est considéré comme confortable, mais pas

exceptionnel. La taille des support de masse suit la même évolution et le concept de mémoire virtuel se généralise sur tous les matériels.

Parallèlement à l'augmentation de la masse des données et la taille des programmes, la vitesse d'exécution a elle aussi augmentée soit par des artifices techniques (mémoire cache) soit par de nouvelles technologies (mémoires plus rapides).

Le paramètre vitesse d'exécution n'étant pas déterminant, et la taille mémoire étant très limitée, sans possibilité de recours à une technique de mémoire virtuelle, il ne peut être question de reprendre l'organisation et les mécanismes de gestion de la mémoire de ces systèmes.

L'allocation mémoire est de type contigüe (simplicité), non segmentée, non paginée, de taille fixe pour un nouveau processus, mais évolutive au cours de la vie du processus, et enfin compactable (optimisation). Le défaut principal des opérations de compactage est d'être grand consommateur en temps d'exécution. Pour le pallier, les opérations de compactage sont réalisées par un processus système, spécialisé. Le compactage ne doit pas nécessairement être effectué à chaque désallocation, il est généralement différé et porte sur un ensemble d'enregistrements ou de zones à désallouer. Rappelons que la désallocation s'opère en deux phases dont la première consiste seulement à positionner un indicateur précisant au système que l'espace mémoire défini peut être affecté à une autre application.

La maîtrise de l'implantation mémoire

L'exécutif est le maître de la gestion mémoire. Les processus utilisateurs ne sont pas autorisés à utiliser les fonctions mémoire. C'est une différence importante avec les systèmes de gestion des micro-ordinateurs, qui par l'intermédiaire des fonctions (Malloc, Free...de la bibliothèque de gestion mémoire d'un compilateur C) de gestion de l'ensemble de développement logiciel, donnent au développeur la possibilité d'intervenir sur l'organisation mémoire. Le terminal devant présenter des mécanismes de gestions transparent par rapport à l'utilisateur, c'est le noyau qui se charge de toutes les opérations, et notamment l'implantation des zones de travail d'un processus en mémoire centrale (fig n°V.17). A chaque commande d'exécution d'un processus (par le système, ou par l'utilisateur) un nouvel espace mémoire est réservé et divisé en zones selon les caractéristiques du processus stockées en élément descripteur de contexte processus. Cette division n'est pas à proprement parler, une coupure. La fonction d'allocation ne définie que trois adresses d'accès (pour les processus nécessitant ces trois zones) et procède d'une part à une réinitialisation du contenu de l'espace concerné et d'autre part un positionnement des bornes de marquage d'un enregistrement. Enfin au niveau de l'élément descripteur d'allocation mémoire les paramètres et code d'accès peuvent êtres positionnés.

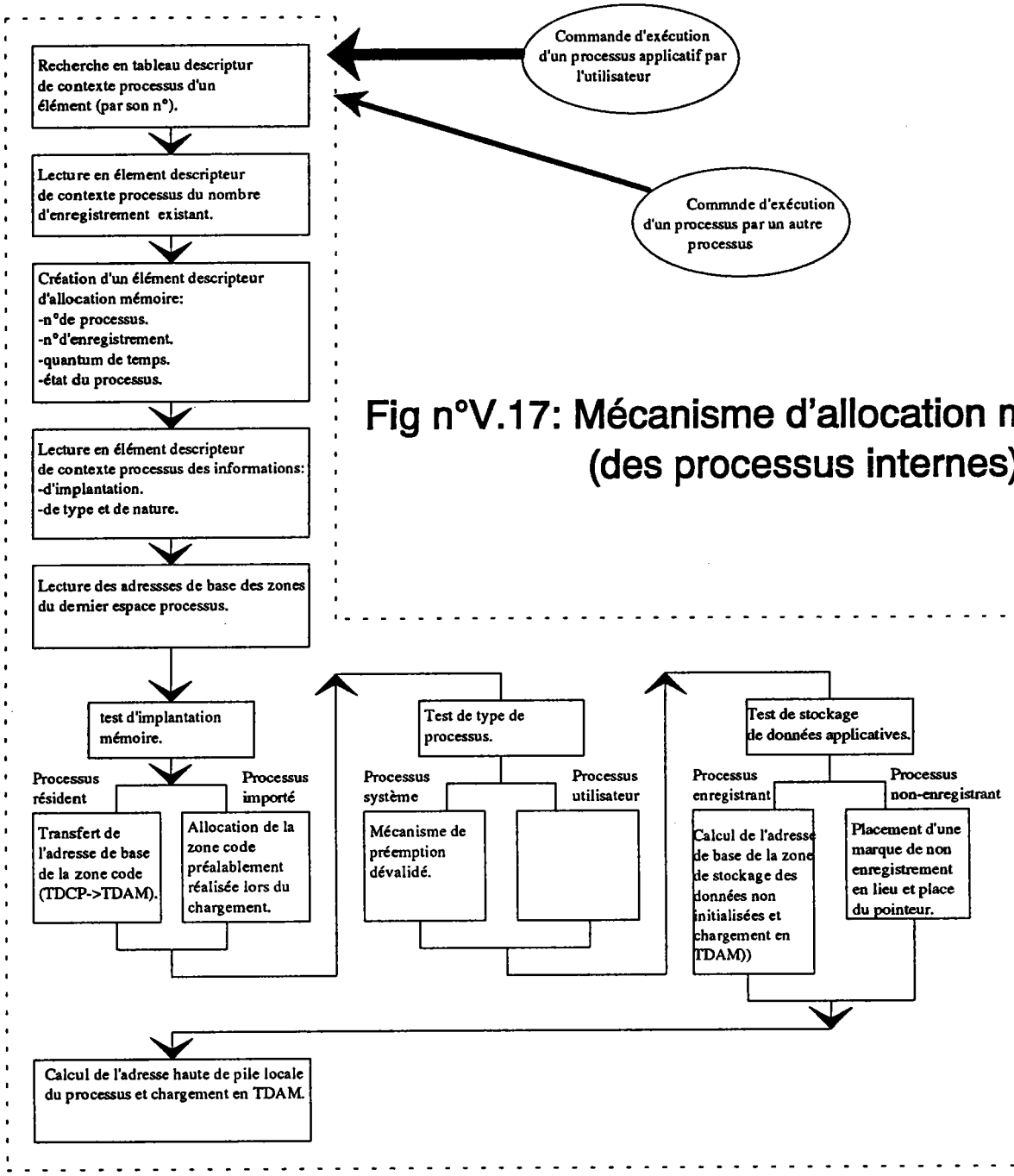


Fig n°V.17: Mécanisme d'allocation mémoire (des processus internes).

Informations présentes en mémoire, dans un espace alloué

Un processus utilisateur, enregistrant, importé (par exemple) se voit affecté (provisoirement) un espace mémoire divisé en trois zones de tailles différentes (4 Ko pour le code, 2 Ko pour les données non initialisées, 500 octets pour la pile) et évolutives (fig n°V.11).

Les premiers octets de la première zone porte la marque de début de l'enregistrement.

La zone pile étant la dernières zone de l'espace mémoire, à son extrémité se trouve la marque de fin d'enregistrement (pour un processus en activité). La pile porte à son extrémité supérieure l'adresse de la prochaine instruction à exécuter, le contenu des registres internes du

processeur, et les adresses de retour de fonctions. Cette zone pile est désallouée, seulement en fin naturelle du processus.

D'autre informations sont présentes immédiatement après la marque de début d'enregistrement, telles que les numéros de processus et d'enregistrement. Ces informations permettrait en cas de destruction totale ou partielle des éléments descripteurs d'allocation mémoire, de les reconstituer par une fonction spécialisée.

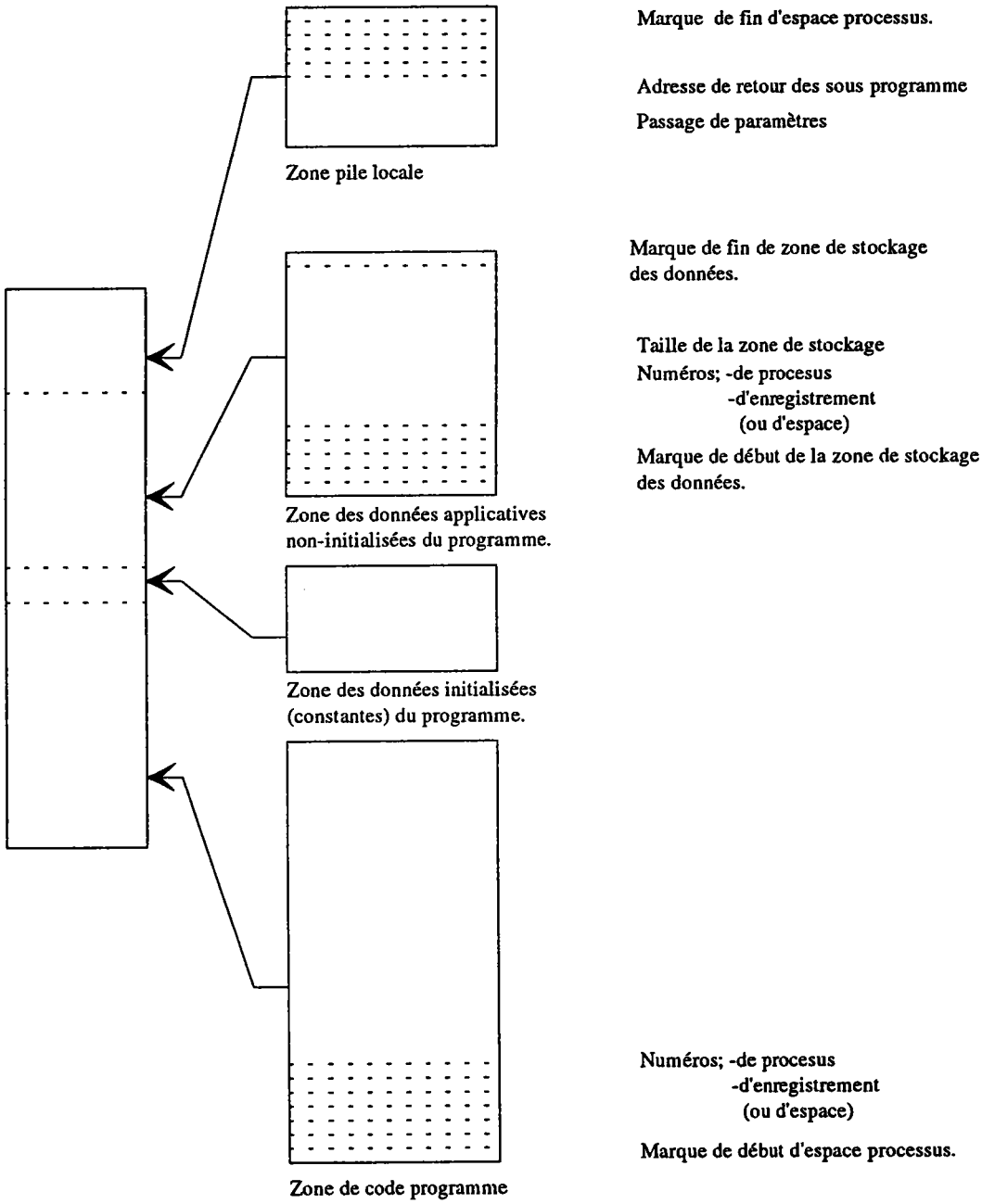


Fig n°V.18: Structuration d'un espace processus.

Cette fonction a pour but, de rechercher les marques de début et de fin d'enregistrement, puis par les numéros caractéristiques de processus et d'enregistrement et les

séparateurs de zone de redéfinir les adresses d'accès à celles-ci (fig n°V.18). Il ne faut pas espérer cependant de cette fonction de remèdes miracles. Elle est comparable aux fonctions de restauration des fichiers perdus, incluses dans les logiciels utilitaires tels que PCTOOLS. Pour d'autres éléments propres au systèmes et subissant des altérations, tel qu'une perte de marque de début ou de fin d'une file d'attente d'exécution le système devra engager une structure de réinitialisation sélective.

La procédure d'allocation

Phase d'attribution d'un espace mémoire à chaque processus

Un espace mémoire n'est attribué à un processus que lorsque le système a reçu un ordre d'exécution de celui-ci. La reprise d'exécution n'entraîne pas l'attribution d'un autre espace distinct car il n'y a pas de nouvelle exécution. Chaque processus est référencé de façon unique par un élément descripteur et chaque exécution entraîne la création d'un élément descripteur d'allocation mémoire (fig n°V.17).

Après que l'ordre d'exécution d'un service des processus associés ait été interprété, le gestionnaire de processus scrute le tableau descripteur de contexte général des processus pour en vérifier sa présence. Le défaut de référence du processus en descripteur entraîne l'affichage d'un message d'erreur. Une fois le processus reconnu certaines informations extraites, l'opération d'attribution d'un espace mémoire débute. Elle consiste essentiellement à créer un élément descripteur d'allocation mémoire, qui va contenir toutes les informations évoluant dynamiquement au cours de la phase d'activité du processus. L'état du processus en fonction des événements survenant au cours de l'exécution ; le processus passe de l'état endormi, à l'état actif (file d'attente). Le signal d'horloge fait passer le processus de l'état activé à l'état interrompu. Au bout d'un temps indéterminé le processus est réactivé, jusqu'à être suspendu en attente d'un événement déterminé. L'exécution pourra reprendre soit de façon infinie soit de façon naturelle.

Le descripteur d'allocation renferme les valeurs de pointeurs d'accès aux différentes zones. Ces valeurs ne sont pas constituées d'adresses explicites mais de déplacements par rapport à une adresse de référence fixe et stockée elle aussi dans l'élément descripteur. Utiliser des déplacements permet de réduire la taille du descripteur. Ainsi au lieux de huit mots longs nécessaire pour constituer un élément, l'utilisation des déplacements permet d'économiser deux mots longs (32 bits chacun). La réduction de taille pour la représentation trouve sa contre partie dans l'introduction de calcul de conversion; calcul de l'adresse à partir de l'adresse de référence et du déplacement ou inversement. Les pointeurs d'accès aux zones données et pile suivent cette représentation. L'accès à la zone code dépend elle de la nature du processus. Un processus résident a son adresse d'accès présente telle que, "en clair". Pour les processus importés, les trois pointeurs sont représentés sous forme de déplacement (fig n°V.14).

Chaque mot long de pointeur d'accès à une zone, est structuré en deux mot de 16 bits correspondant à deux valeurs de déplacement. Le mots long de poids le plus fort correspond au codage en déplacement du pointeur de base à la zone concernée; valeur qui reste inchangé au cours de l'exécution du processus. Par contre le mot de poids faible porte le déplacement du pointeur courant de la zone concernée, et sa valeur est mise à jours à chaque interruption de gestion des tâches. Cette représentation double dans un même élément descripteur a été conçu pour faciliter le travail du processus système de compactage mémoire.

Taille des espaces mémoire alloués aux processus

Travailler avec des espaces mémoire de taille variables est plus efficace même si cela augmente la complexité de la gestion mémoire; mais cette notion de taille doit s'envisager différemment selon la nature des opérations qui concerne la mémoire (allocation, libération, extension...), selon le type de processus utilisant l'espace. Il n'est pas question de faire que tous les processus tout au long de leur exécution aient même taille mémoire, car tous n'ont pas les mêmes besoins que cela soit par le nombre de zones (résident et enregistrant: 2 zones; importé et enregistrant: 3 zones)ou que cela soit par la taille des zones qui peuvent être de même nature (zone données du processus 1: 4 Ko; zone données du processus 2: 6Ko) ou de nature différentes (données: 4 Ko; pile 512 octets).

Lors de l'allocation d'un espace pour un processus la taille alloué est fixe car on ne peut prédire le volume des données à stocker. Ce volume évolue en fonction du temps, et en fonction de l'utilisation du processus. Si besoin est l'une des zones peut être augmentée, à n'importe quel moment de la vie d'un processus. Ceci ne préjuge pas de la politique d'extension d'une aire mémoire.

L'extension d'une zone mémoire sera réalisée par blocs de taille fixe mais réduite (2 Ko pour la zones code,1Ko pour la zone données, 128 octets pour la zone pile).

L'absence de segmentation, de pagination pour le 68008 permet d'allouer à un processus n'importe quelle taille mémoire sans contrainte matérielle forte. Il existe bien quelques contraintes telle la nécessité que la taille soit de nombre paire liée au fait que le stockage d'une donnée nécessitant plus d'un octet (mot et mot long) ait une adresse paire.

Compactage des espaces mémoire

Un processus de compactage des zones mémoire

C'est un processus système, c'est à dire un processus ne nécessitant aucune représentation graphique, jouant un rôle d'organisation et de gestion dont l'unique bénéficiaire est le système. Son but est de regrouper tous les espaces mémoire libres disséminés pour constituer une aire mémoire libre et unique,la plus vaste possible. A partir de la structure

descriptive de la mémoire en place un certain nombre d'opérations séquentielles interviennent pour réaliser le compactage:

-le tri et la réorganisation des éléments descripteurs selon l'ordre d'occupation mémoire .

-la recherche des zones libérables.

-la translation des données (code exécutables, données, pile).

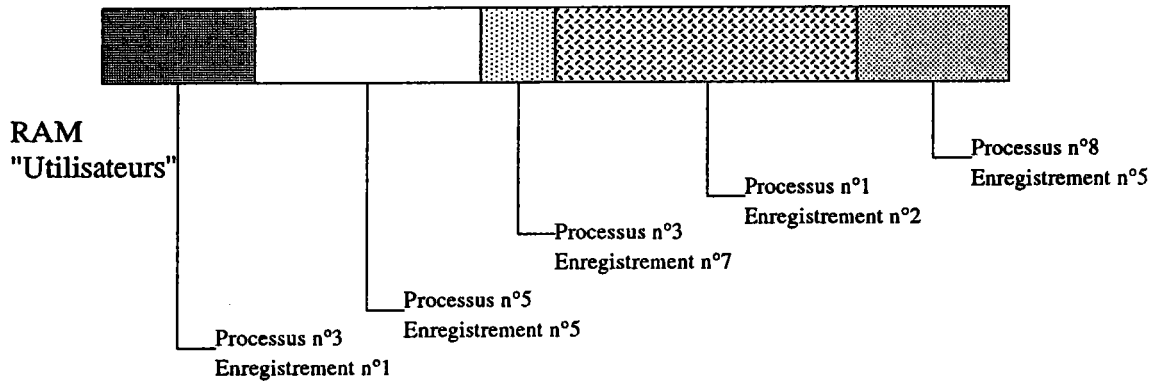
-la réduction et la fusion des éléments descripteurs.

-l'adaptation des liens pour les variables et symboles.

Une opération de compactage n'est pas réalisée à chaque désallocation mémoire automatique, mais elle est commandée par un seuil de commande. Le principe et les différentes phases du processus de compactage sont détaillés par la figure n°V.19.

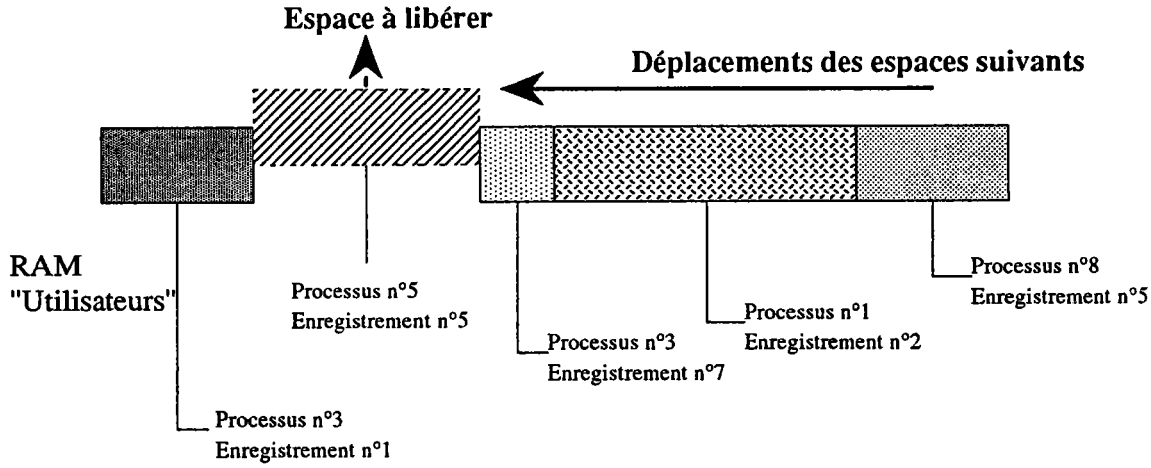
Fig n°V.19: Procédure de compactage des enregistrements mémoire

Schéma d'occupation initiale de la mémoire

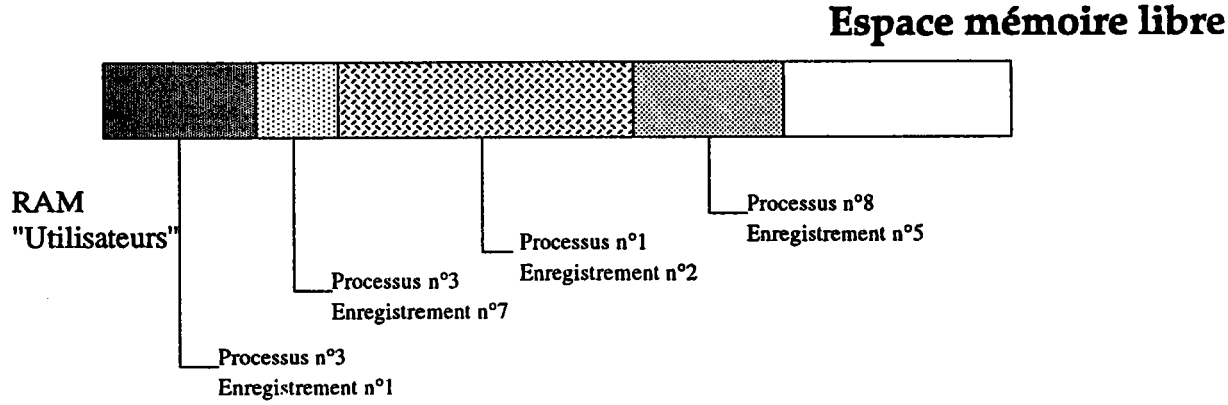


1ére étape: désallocation d'un espace mémoire affecté à un processus

2éme étape: réorganisation interne du tableau descripteur d'allocation mémoire



3éme étape: déplacement des espaces mémoire alloués supérieurs (réinitialisation de la mémoire et déplacement du contenu des espaces).



Une désallocation automatique différenciée

La procédure de désallocation est s'envisage en deux étapes. En premier lieu , une fonction recherche en tableau descripteur d'allocation mémoire, l'élément descripteur correspondant aux numéros de processus et d'enregistrement, puis y positionner le code de commande de désallocation. Il s'agit en fait d'une fonction de commande de désallocation dont le dernier rôle (si le nombre de commande a atteint un certain niveau) est de charger en file d'attente le numéro du processus de compactage. Cette fonction de commande intervient soit lorsque le processus vient de terminer son exécution naturellement (la zone à désallouer automatiquement est la pile), soit lorsque par sa télécommande l'utilisateur en demande l'ordre au système pour effacer des données, ou pour détruire un programme importé.

La deuxième étape comme on peut s'en douter consiste à compacter les espaces processus. Elle ne rentre en action que lorsqu'un seuil de commandes de désallocation est atteint.

La désallocation ne touche pas obligatoirement tout l'espace processus. Cette désallocation est paramètrable et différenciée par un code de désallocation défini (fig n°V.20).

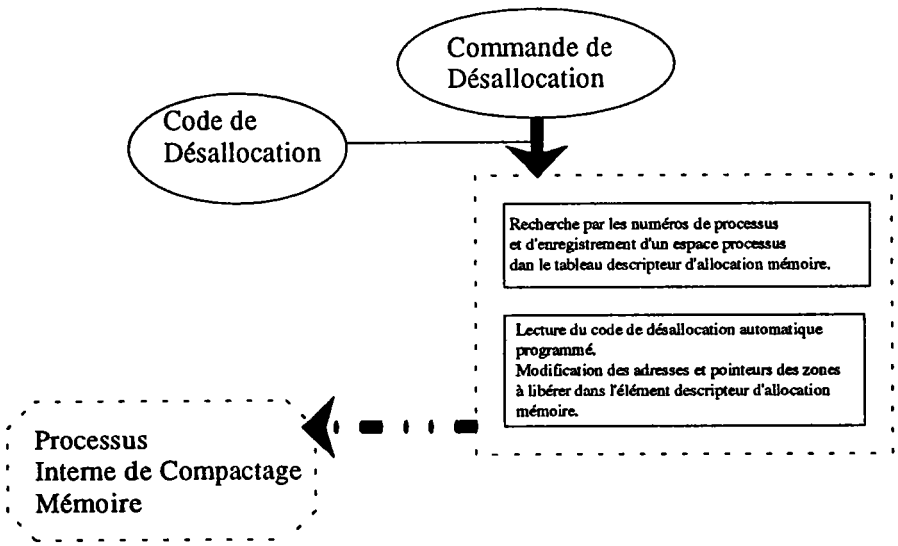


Fig n°V.20: Désallocation automatique différenciée de la mémoire.

L'extension mémoire

Principe

L'extension d'un espace mémoire alloué à un processus est elle aussi attribuée à un processus système spécialisé dans l'extension d'une ou plusieurs zones mémoire.

Le mécanisme d'extension est particulier du fait du type d'allocation contigüe utilisé. Par définition l'espace alloué à un processus est d'un seul tenant. Pour augmenter cet espace en gardant son unicité il faut soit déplacer le contenu des espaces supérieurs, soit (et c'est la solution retenue) allouer au processus réclamant une extension , un espace étendu pris dans l'espace libre restant, puis y translater le processus et enfin lancer une opération de compactage (fig n°V.21). En cas d'insuffisance de l'espace libre, l'opération peut être interrompu, sans dommage pour le processus demandeur.

Prise de décision d'extension

Le processus d'extension d'une zone mémoire est lancé de façon automatique lorsqu'un seuil de remplissage de la mémoire allouée est franchie. Le mécanisme de détection de dépassement du seuil est différent selon les zones concernées.

Pour une zone donnée, la détection est réalisée à chaque enregistrement en mémoire, par la fonction de chargement d'une donnée, et ceci par comparaison entre l'adresse de stockage courante (valeur du registre d'adresse d'accès indirect avec la mémoire) avec le seuil de remplissage calculé de cette zone. En cas de dépassement du seuil, le numéro de processus interne correspondant aux opérations d'extension est chargé en file d'attente de niveau 2. Une autre possibilité de mise en route immédiate était concevable; le recours à une exception de type logicielle (un trappe), ce qui ne change rien a-priori aux opérations d'extension et ne recèle pas en principe de risque de blocage en zone critique.

Pour les autres zones et notament la zone réservée à la pile locale, la prise de décision d'extension est effectuée par les fonctions du noyau qui comparent les adresses de base de la pile ou adresse d'extrémité de zone code, et les adresses courantes correspondantes.

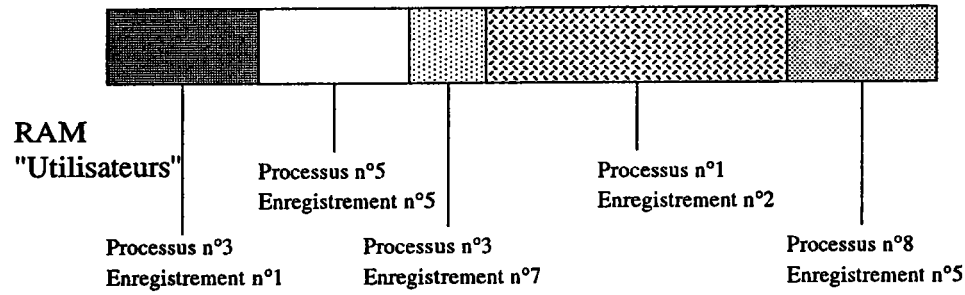
Le processus d'extension mémoire

Ce processus est un processus interne, ne faisant intervenir en principe aucun périphérique (graphique, communication), puisque sa seule action porte sur la mémoire, et son contenu. Une particularité réside dans la non préemption. Ce processus une fois lancer (son tour étant venu) s'exécute sans interruption (horloge système) jusqu'à la fin du traitement. Seules les interruptions issues du réseau de communication Médiabus sont prises en compte. Le fait que ce processus ne comporte pas de traitement liés aux périphériques, exclus en théorie tous blocage du à des zones critiques.

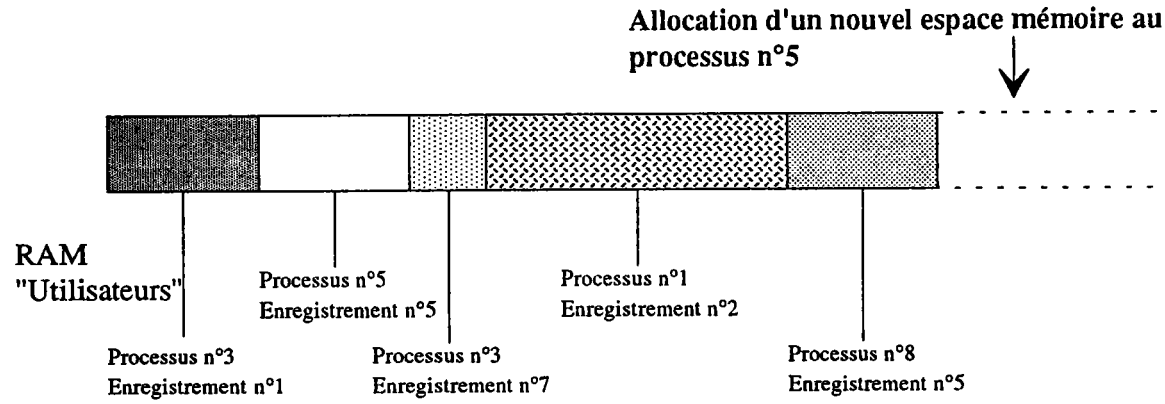
En mémoire, les processus sont placées de façon contigüe sans zone libre entre les différents espaces.

Fig n°V.21 : Procédure d'extension d'un enregistrement mémoire

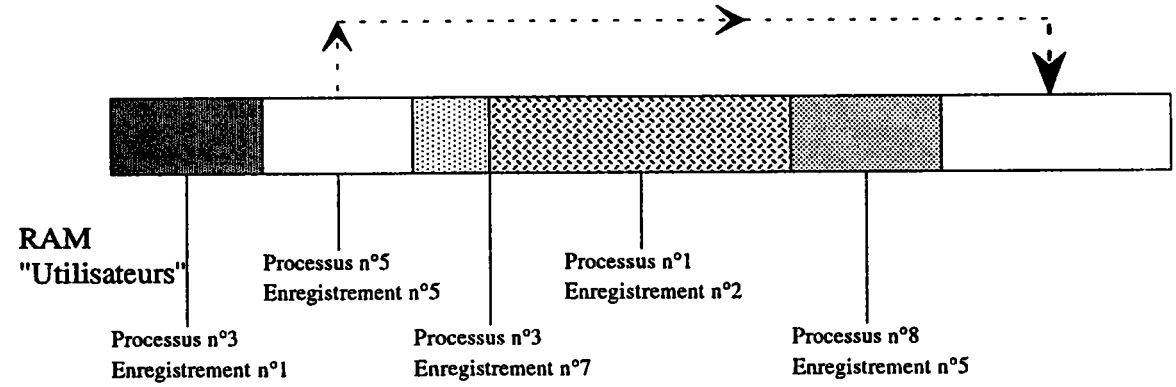
Schéma d'occupation initiale de la mémoire



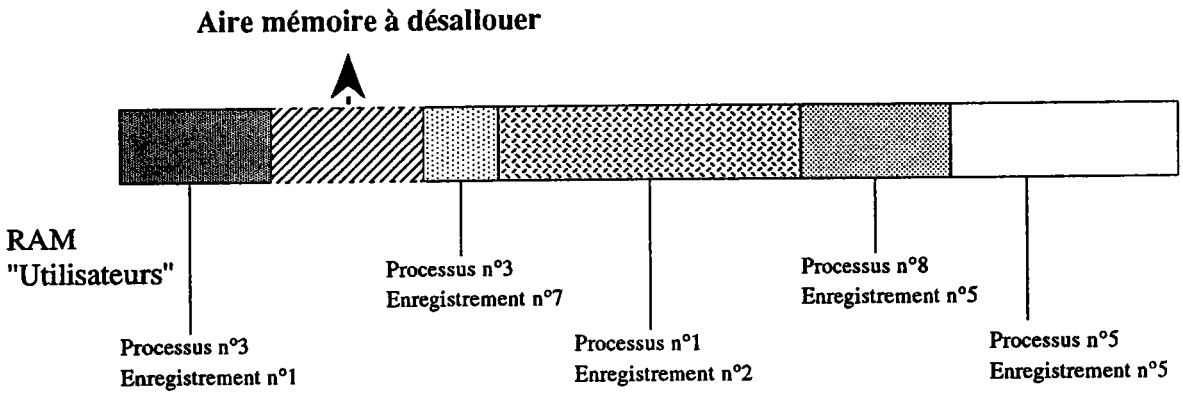
1ére étape: allocation d'un espace mémoire étendu affecté au même processus



2éme étape: déplacement du contenu de l'espace mémoire initial dans le nouvel espace étendu

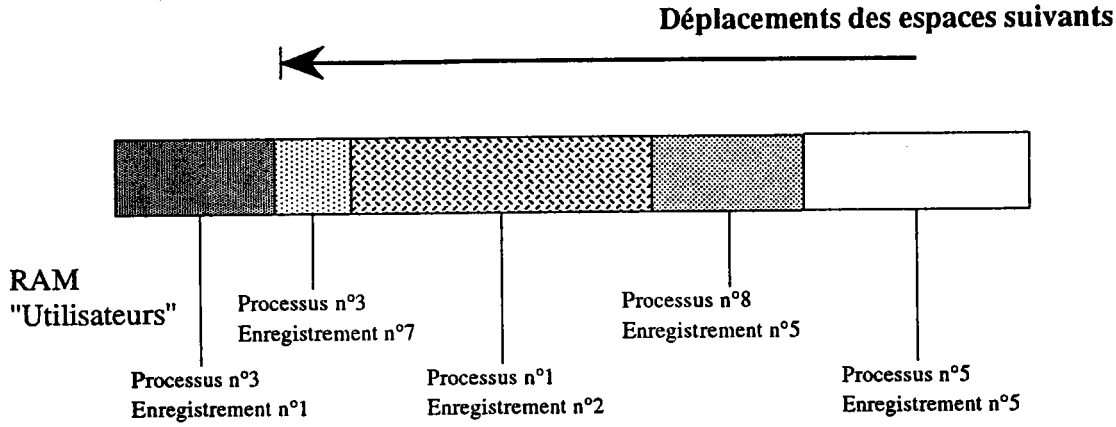


3éme étape: libération de l'aire mémoire précédement allouée au processus



4éme étape: restructuration du tableau descripteur d'allocation mémoire.

5éme étape: déplacement des espaces mémoire alloués supérieurs



Sécurités d'accés à chaque espace mémoire

Sécurités d'accés aux données

Il existe deux niveaux de sécurité pour l'accès des données en mémoire, en lecture ou en écriture. Le premier concerne le mécanisme de gestion de la mémoire et vise à éviter l'écriture de données hors des zones mémoire affectées à ces données. Sont concernées, en premier lieu la pile locale du processus, puis les données non initialisées. Le système de sécurité a pour but d'allouer de la mémoire supplémentaire pour stocker de nouvelles données en évitant de perdre des données par chevauchement.

Le second niveau de sécurité porte sur l'accès sélectif en lecture de certaines zones mémoire selon l'interlocuteur et ses droits d'accès. Ainsi le réseau se verra interdire l'accès en

lecture de zones mémoire attribuées de façon personnelle à l'utilisateur du terminal (exemple: la zone mémoire de messages reçus).

Toute solution logicielle de gestion sécurisée de la mémoire présente des défauts et se révèle être moins efficace qu'une solution matérielle (unité de gestion mémoire MMU, différenciation Superviseur/Utilisateur).

Un problème: le débordement de la pile

Lorsque la pile empiète sur des zones allouées aux processus, l'espace processus est considéré comme perdu. La catastrophe, peut provenir soit de la pile locale d'un processus, soit de la pile système. Pour limiter de tels problèmes et les effets catastrophiques qui en découle, j'ai penser à titre préventif (mais non miraculeux !) d'impliquer une fonction de surveillance de l'évolution de la pile. Cette fonction teste si le pointeur de pile se trouve à proximité d'un espace mémoire alloué à un processus (pour la pile système). La solution est brutale et consiste à revenir en haut de la pile. Pour la pile locale le test porte sur une comparaison entre une adresse de seuil d'occupation de la zone pile et la valeur du pointeur . De cette façon j'essai de prévenir le débordement de pile. Si le seuil d'occupation de pile est atteint, la fonction de test charge en file d'attend le numéro du processus d'extension pour la zone de pile locale au processus concerné.

En somme...

J'ai essayé de doter le terminal de mécanismes de gestion mémoire, adapté à la gestion de la seule mémoire centrale de l'équipement, en utilisant des techniques d'allocation contigüe, et de fusion, compactage, extension par recopie des espaces processus pour assurer une utilisation optimale du support. Pour l'avenir l'utilisation d'une mémoire périphérique permettrait l'utilisation de technique de gestion de mémoire virtuelle, sans remettre en cause la politique d'allocation contigüe des espaces mémoire.

VI AUTRES ELEMENTS LOGICIELS DE L'EXECUTIF

VI.1: Interactions entre les différentes couches constituant le logiciel d'exploitation

Les trois couches logicielles constituent trois programmes indépendants ayant chacun leurs fonctions. Le moniteur multitâche, à partir des caractéristiques de chaque processus lance l'exécution de celui-ci. Le processus prend alors le contrôle de toutes les ressources matérielles du terminal jusqu'au moment où un événement extérieur au processus survient sous forme d'une interruption logicielle ou matérielle. La nature de l'événement conditionne les échanges entre les couches mais quel qu'il soit, il implique l'intervention de la couche système en faisant appel aux mécanismes de traitements des interruptions en mode superviseur du processeur.

Interactions processus-exécutif

Cette interaction s'effectue dans les deux sens et de deux façons différentes. A la suite de tout événement le processeur quitte l'exécution du processus pour traiter l'une des routines de traitement d'interruption. C'est un déroutement d'exécution provisoire ou définitif n'introduisant aucune transmission de donnée, autres que la sauvegarde des registres internes du processeur en pile locale du processus.

Si l'événement concerne une interface de communication la donnée est reçue, traitée (superficiellement) et le processus interrompu (couche application) reprend son exécution sans échanges de données entre le processus externe et le processus interne (fig n°VI.1). Le contenu des registres internes est restauré à cette occasion.

Lorsque l'événement pris en compte est le signal de l'horloge préemptive, le traitement de la routine d'interruption conduit soit à rester au niveau superviseur pour réaliser une nouvelle opération d'ordonnancement, soit l'exécution du processus est reprise jusqu'à la fin de la zone critique précédemment interrompue. Outre la restauration du contenu des registres internes du microprocesseur, une variable d'état du système est positionnée. Cette variable globale est consultée lors de toute fonction de décrémentation du sémaphore. Cette fonction système permet soit de poursuivre l'exécution linéaire du processus avec dévalidation du sémaphore, soit conditionne le retour à la couche système et la clôture du processus en cours, par utilisation programmée d'une exception (trap #1).

Les échanges de données entre les deux couches extrêmes, sont donc très limités, mais par contre leurs interactions définies dans un cadre strict sont très fréquentes.

Interactions processus-interpréteur

A la suite de la réception du code de télécommande sous interruption (de niveau 5), le processeur ne retourne pas immédiatement au processus en couche applicative mais passe par l'interpréteur de la couche d'interfaçage logicielle (protocole réseau, gestionnaire graphique, interpréteur de commande). Le code télécommande est la valeur d'interaction unidirectionnelle entre la couche basse et l'interpréteur. Suivant le code reçu, des actions sont engagées par le processeur et conduisent, soit à un retour au moniteur pour l'activation d'une nouvelle application (fig n°VI.2), soit au retour à la couche application après exécution de la routine graphique (déplacement du curseur). En cas de passage au moniteur, le flux d'informations porte sur le numéro de processus correspondant au cryptogramme le représentant.

Interactions processus-processus

Sur le plan des échanges de données entre deux processus, la seule méthode envisagée consiste à définir pour l'occasion un tampon. Cette procédure est rudimentaire dans son principe et dans sa réalisation, et elle mériterait un effort de développement. Chaque tampon de communication est délimité par des marques de début et de fin, et renferme les coordonnées du processus (numéros de processus et d'enregistrement) ayant ouvert et chargé d'une part le tampon, et d'autre part la taille de celui-ci.

L'interaction entre processus porte aussi sur l'aspect des appels de processus entre eux. Comme vu précédemment un processus commande l'exécution par le système (et par son numéro) de l'exécution d'un autre processus après la fin du premier (par exemple; un ensemble de données en provenance du réseau est stocké en tampon puis repris par un processus de décodage aprés détection d'une fin de message). Mais un processus peut aussi commander l'exécution d'un autre processus sans avoir pour autant terminer son exécution. Le deuxième processus n'ayant pas toutes les données issues de la première tâche sera en position d'attente correspondant à l'état suspendu du processus en attente de l'événement de réception des informations nécessaires.

Figure VI.1 : Schéma d'interaction entre les trois couches constituant le logiciel d'exploitation lorsqu'une interruption est générée par le port série RS232C.

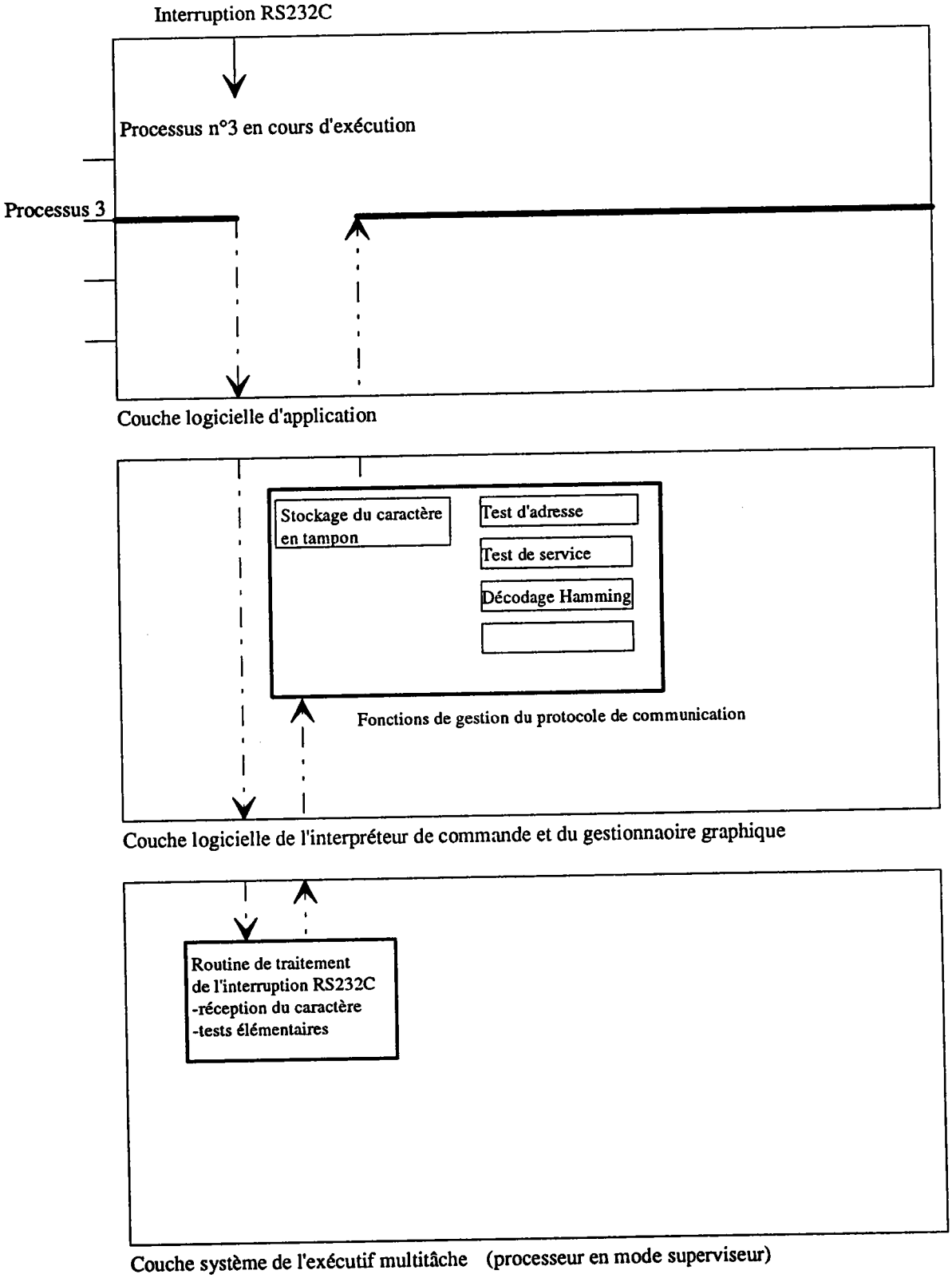
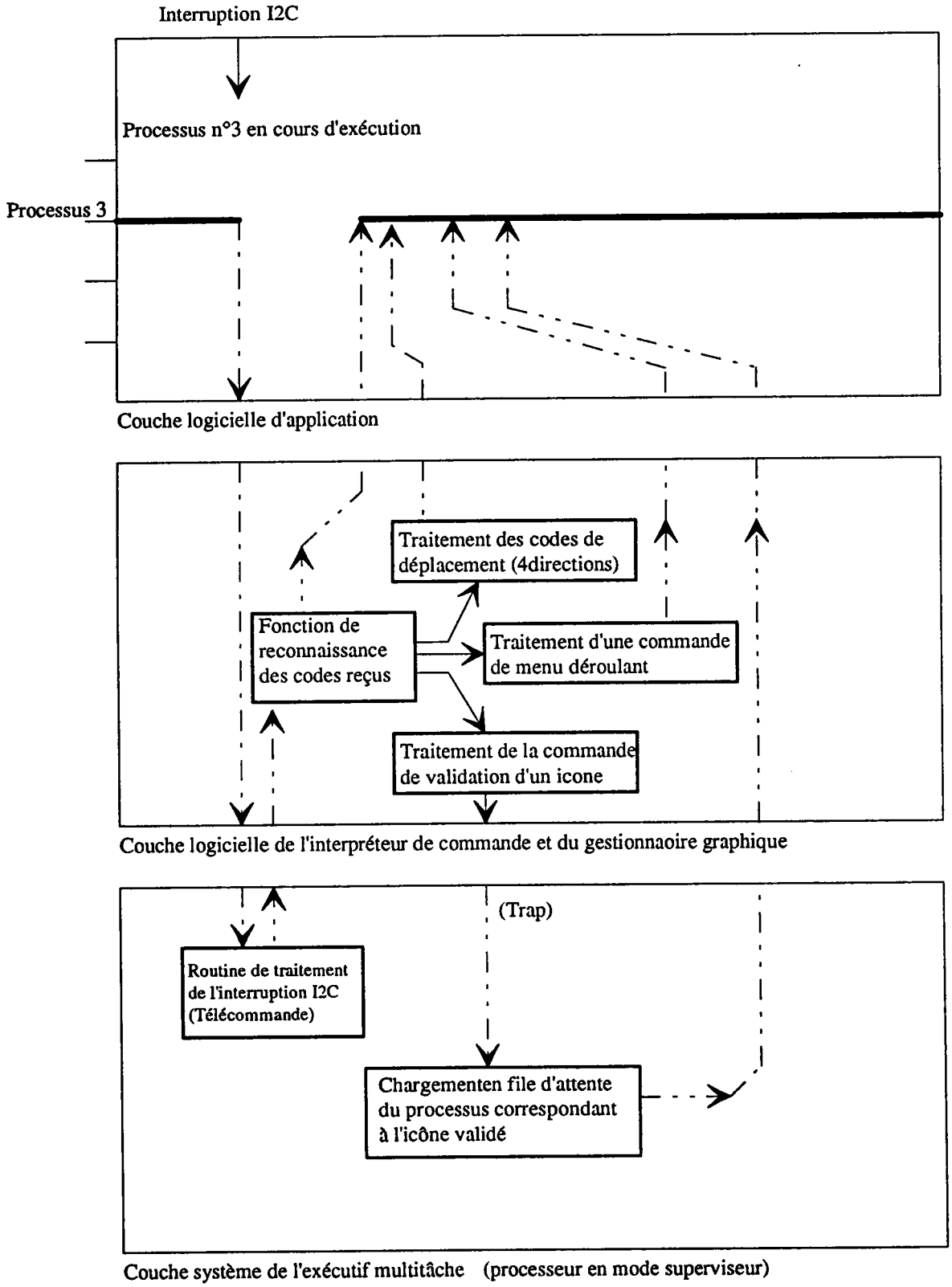


Figure n°VI.2 :Schéma d'interaction entre les trois couches constituant le logiciel d'exploitation lorsque survient une interruption générée par l'un des composants connecté au bus de communication I2C.



VI.2: Les interruptions

Une interruption est une procédure à partir de laquelle certains événements intérieurs ou extérieurs peuvent arrêter le déroulement logique et linéaire du programme principal pour répondre à une sollicitation ou réagir à un événement a priori aléatoire. Selon les microprocesseurs les interruptions et leurs nombres sont différents.

Evénements déterministes ou aléatoires

Les événements susceptibles d'impliquer la génération d'une interruption surviennent soit de façon aléatoire soit de façon déterministe. La nature de ces événements et leur périodicité ont une incidence sur la méthode de captation des données et sur la structure des traitements liés ou non aux interruptions. Ainsi les données en provenance d'un réseau constituent des événements périodiques donc déterministes. Par contre l'interruption générée par l'ensemble récepteur-transcodeur de télécommande IR est de type aléatoire. Considérant la fréquence de survenance des données sur le port série (période de 520 µs) et pour éviter de mobiliser trop longtemps l'unité centrale en scrutation des données en registre de réception, il est nécessaire de réaliser les opérations de réception des caractères sous interruption. De même, et bien que la survenance d'un code télécommande soit aléatoire, sa réception sous interruption est la condition d'une prise en compte certaine de celle-ci.

Les composants d'interface utilisent les interruptions pour attirer l'attention du processeur, mais tant que le niveau de cette interruption est inférieur ou égal au masque d'interruption, la requête est ignorée. Si cette requête est temporellement limitée, elle a de grande chance de n'avoir jamais existée. Ce n'est pas le cas des interfaces série et I2C pour lesquelles il est nécessaire de lire le registre de données pour que le signal d'interruption généré par le composant soit désactivé. Ceci garantit la prise en compte par le processeur du signal d'interruption mais ne peut pas définir le temps au bout duquel cette reconnaissance sera effectuée. Cette dernière difficulté peut être levée indirectement en considérant la hiérarchie des interruptions (fig n°VI.3).

Les exceptions logicielles et matérielles

Il existe deux types d'interruption:

-les interruptions matérielles qui comme le nom l'indique sont générées par un périphérique quelconque voulant intervenir sur le microprocesseur, ceci pour demander le transfert en mémoire d'une donnée reçue par un périphérique, ou pour signaler au processeur qu'une demande de réinitialisation du système a été commandée par l'utilisateur. Ce sont généralement des interruptions externes, et leur réponse ainsi que leur prise en compte est programmable. Ces interruptions sont hiérarchisées et prioritaires.

-Les interruptions logicielles ou exceptions sont des interruptions générées en interne, à la suite soit de la détection d'un défaut de programmation (erreur d'adresse, division par zéro par exemple) générant un blocage au niveau du processeur, soit d'une demande d'accès au niveau superviseur (déroutement) pour exécuter une fonction spécifique et protégée. Au nombre de quinze, ces exceptions appelées aussi TRAP (instruction trap # de demande d'exception) sont programmables. Seules les exceptions système prennent le pas sur les interruptions matérielles.

Priorité des groupes	Groupes	Exception	Priorité au sein d'un groupe	Moment où l'exception est reconnue	Moment où commence le traitement de l'exception (pouvant éventuellement être retardé par HALT, BR)
↑	0	RESET ERREUR BUS ERREUR ADRESSE	↑	Fin du cycle horloge L'instruction en cours est avortée	Cycle horloge suivant
↑	1	TRACE INTERRUPTION INST. ILLEGALE INST. NON IMPLÉMENTÉE INST. PRIVILÉGIÉ	↑	Fin de l'exécution de l'instruction en cours Fin du cycle bus Instruction en cours avortée.	Après l'instruction en cours et avant la prochaine instruction
↑	2	TRAP TRAPV CHK DIVISION PAR ZÉRO	Pas de priorité car chaque instruction est effectuée séparément	Pendant le déroulement de l'instruction, après son décodage	A la fin de l'instruction en séquence et avant la suivante

Fig n°VI.3: Hiérarchie des exceptions logicielles et matérielles.

Hiérarchie et mécanisme des interruptions

Pour le processeur 68008, les entrées d'interruption, au nombre de deux (IPL2/0, IPL1) autorisent ainsi quatre niveaux d'interruption (niveaux 0, 2, 5, 7). Trois de ces niveaux sont masquables. Par programmation du registre d'état (fixation du niveau minimal d'interruption) on autorise ou non la prise en compte du signal d'interruption par le processeur.

Sont prises en compte toutes les interruptions de niveau strictement supérieur au niveau programmé dans le registre d'état c'est à dire à la valeur du masque d'interruption. Le niveau 7 qui a la plus haute priorité, est non masquable, et est muni d'un système matériel de réarmement automatique interne. Une interruption de niveau 7 peut interrompre une interruption de même niveau. Le niveau 0 permet la prise en compte de toute interruption (fig n°VI.3).

La hiérarchie des interruptions fait qu'une interruption incidente ne peut prendre le pas sur une interruption en cours de traitement que si son niveau est supérieur. Le traitement de l'interruption n'est pas immédiat car elle est détectée en fin d'exécution de l'instruction en cours.

Pour garantir la prise en compte quasi immédiate d'une interruption il est nécessaire que son niveau soit supérieur au niveau en cours du registre d'état. Cela suppose d'une part de hiérarchiser les événements et les traitements afférents, et de définir les zones de code interruptibles.

Répartition des priorités d'interruption selon les interfaces du terminal

Cette hiérarchie de connexion des entrées correspond aux choix de traitements prioritaires des événements et donc aux finalités de l'équipement. L'orientation du terminal pour l'écoute permanente du réseau Médiabus implique que l'interruption du port série RS232C soit de niveau 7. Ce positionnement permet au système de ne perdre théoriquement aucune donnée véhiculée sur le réseau. Comment répartir ensuite les deux dernières sources d'interruption sur les deux entrées disponibles?

En positionnant l'horloge système au niveau 5 et l'ensemble I2C au niveau 2, on rend prédominant l'aspect multitâche sur la deuxième interface (l'interface de commande par l'utilisateur).

L'autre option consistant à placer l'interface I2C sur le niveau 5 et l'horloge de synchronisation des tâches sur le niveau 2 est plus homogène et correspond plus à la connotation d'outil de communication que l'on veut lui donner. Cette notion d'homogénéité porte aussi bien sur le groupement des interruptions d'interfaces sur les niveaux de priorité que sur le choix de classification en processus interne ou externe.

Ainsi donc la répartition des interruptions est la suivante:

-niveau 7 (priorité maximale):	port série RS232C connecté sur le réseau Médiabus.
-niveau 5:	interface bus I2C.
-niveau 2 (priorité minimale):	horloge système de préemption.

Note:
Les symboles représentant des signaux électriques et comportant le signe "\" fonctionnent selon une logique négative (ex: VPA\ :=0 état actif; =1 au repos).

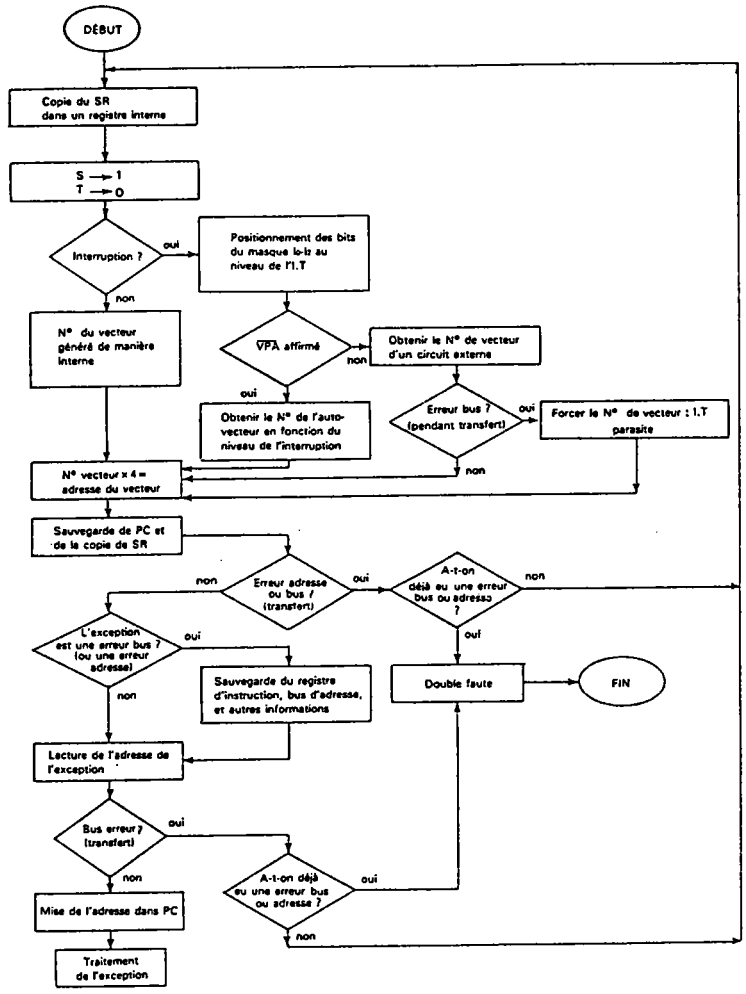


Fig n°VI.4: Organigramme du traitement des exceptions (sauf le reset).

Vectorisation automatique et vectorisation non-automatique

La nature du composant ayant généré l'interruption matérielle influe sur le mécanisme de reconnaissance de celle-ci. Le traitement microprogrammé intervenant dans la phase de reconnaissance de l'interruption d'un composant de la famille 6800 est de type à vectorisation automatique (fig n°VI.4). Ceci a été mis en place pour garder la compatibilité avec les interfaces de cette autre famille de processeur 8 bits. C'est ce type de vectorisation qui est utilisé pour traiter les événements issus du port série RS232C (MC 6850) et de l'horloge système (MC 6840). Par contre les autre composants sources d'interruption sont traités en vectorisation non-automatique, et font intervenir un mécanisme de reconnaissance microprogrammé légèrement différent.

En vectorisation automatique on réalise l'économie de la demande du numéro de vecteur par le composant interrompant. Le nombre de vecteurs automatiques est limité par le nombre de niveaux d'interruption (au nombre de quatre).

Opérations microprogrammées de traitement d'une interruption

Entre le moment où le processeur perçoit le signal d'interruption (quel que soit son niveau) et le traitement logiciel conçu par le développeur, un certain nombre d'opérations microprogrammées non modifiables interviennent (fig n°VI.4). A la suite de la réception d'un signal d'interruption le processeur termine l'exécution de l'instruction en cours, puis entre en phase de reconnaissance d'interruption. Cette phase consiste à comparer la valeur du masque en registre d'état et le niveau d'interruption.
Si la valeur du masque est supérieure à celle de l'entrée d'interruption, le processeur reprend l'exécution linéaire du programme en cours. L'interruption est inhibée. Aucun registre n'est affecté par ces opérations.
Par contre avec un masque inférieur au niveau de requête présenté par l'interruption, le processeur adapte les bits d'état du processeur (utilisateur->superviseur) et du masque en registre d'état, et recherche le numéro de vecteur correspondant. Ce numéro de vecteur est (à ce moment) l'entrée d'interruption activée, qu'il "affiche" sur les lignes d'adresse A1-A2-A3, pour adresser le composant interrompant. A ce stade se détermine le type de traitement auto-vectorisé ou non-auto-vectorisé, à partir du signal utilisé par le composant en accusé de réception. Avec un signal VPA\ validé (voir Note p 124), le processeur suit un protocole de traitement d'une interruption type famille 6800, auto-vectorisée (fig n°VI.4). Un accusé de réception sous forme d'un DTACK\ conduit à un traitement non-auto-vectorisé. Sont disponibles avec le 68008 trois interruptions matérielles auto-vectorisées, et 192 possibilités de vecteurs interruption non auto-vectorisés configurables par l'utilisateur.

Auto-vectorisation (ou vectorisation automatique)

A la suite de la réception du signal VPA\ , de façon interne le processeur calcule la nouvelle valeur du vecteur d'interruption; n° du vecteur= niveau de l'IT+ 24 (base décimale) qui représente une valeur d'entrée dans une table de vecteurs. Ce calcul marque la fin de la phase de reconnaissance de l'interruption, puis la sauvegarde en pile du pointeur ordinal et du registre d'état. La procédure de traitement d'interruption peut alors commencer par le branchement à l'adresse correspondant au n°de vecteur dans la table.

Non-autovectorisation

Lors de la génération de l'accusé de réception par le composant, celui-ci place sur le bus de données le n°de vecteur correspondant qui lui a été précédemment attribué lors de son initialisation. L'adresse de branchement à la routine de traitement de l'interruption peut alors être déterminée par correspondance entre le numéro de vecteur et adresse physique d'accès au

code de traitement, via le tableau des vecteurs, placé à la base de la mémoire RAM (par utilisation du même calcul qui précédemment).

Limitation du nombre d'entrées d'interruption

Le 68008 ne comporte que deux entrées de signaux d'interruption ce qui est actuellement suffisant pour l'équipement de base. Pour des développements matériels ultérieurs faisant intervenir de nouvelles sources d'interruptions (réseau Batibus, capteurs...) il sera nécessaire de connecter plusieurs sources différentes sur une même entrée. Dans ce cas, les premières opérations à réaliser par la routine de traitement de l'interruption seront d'interroger tour à tour les composants connectés à cette entrée pour déterminer lequel est la source de l'interruption. Cette solution est d'ailleurs retenue pour l'interruption générée par les composants I2C sur un seul fil. La situation est homogène car tous les composants connectés à cette entrée appartiennent au même ensemble.

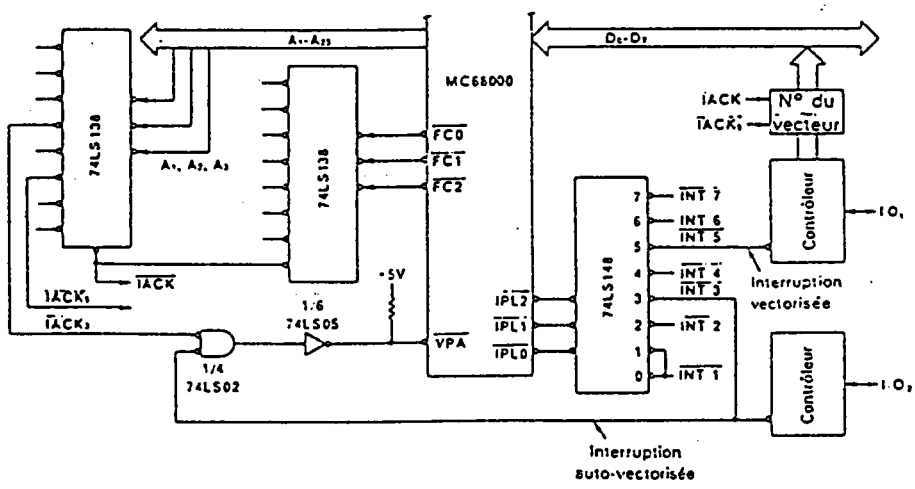


Fig n°VI.5: Ensemble logique des interruptions auto-vectorisées et non auto-vectorisées.

Détermination des composants source d'interruption I2C

Après détection d'un signal d'interruption I2C , les opérations de traitement consistent à interroger successivement tous les composants comportant un registre d'état mémorisant leur état (générant ou pas une interruption) tels que le calendrier/horloge/alarme, compteur d'événements (figure n°VI.10). Si aucun de ces composant n'a marqué le désir d'attirer l'attention du processeur et s'il existe d'autres composants ne disposant pas d'un registre d'état que l'on peut consulter, mais susceptible de générer un signal d'interruption (tel que transcodeur IR-I2C) la logique veut (par élimination) que ces composants soient les sources

du signal et les traitements correspondants sont alors mis en route. Cette situation implique qu'il vaut mieux alors ne disposer que d'un seul composant tel que le récepteur/transcodeur de télécommande.

Cette procédure de tests successif à réaliser avant tout traitement n'est qu'un palliatif au manque d'entrées d'interruption car ce système est consommateur de temps. Sont placés en tête des tests, les composants interrogeables et générant un signal le plus fréquemment:

-1)le composant horloge/calendrier/alarme.

-2)le compteur d'événements.

-3) le récepteur/transcodeur IR-I2C.

Définition des niveaux d'interruption dans le code exécutable.

Nous avons vu que le positionnement du niveau d'interruption constitue l'une des politiques de sécurisation de certaines zones de code contre la survenance d'événements intempestifs. En considérant le fonctionnement linéaire du logiciel, les deux premières phases d'initialisation matérielle et d'installation du système multitâche ne doivent pas être perturbées, et donc le masque d'interruption est positionné au niveau 7. Seule l'écoute du réseau Médiabus est autorisée.

Par la suite, en phase d'exploitation normale, au cours de l'exécution d'un processus toute interruption est autorisée (niveau 0, 2, 5, 7), et seuls quelques éléments de code présentent une exception à cette règle, pour protéger l'action de certains mécanismes (exemple: dévalidation d'un sémaphore).

Entre chaque phase d'exécution des processus le processeur exécute les fonctions d'ordonnancement avec des valeurs de masque d'interruption évolutives. Bien que se voulant être un moniteur temps réel, il est impossible de réaliser un code exécutable interruptible à tout moment. Certaines opérations délicates, telles que les opérations:

- de lancement d'exécution,
- de déroutement,
- de sauvegarde et restauration du contexte d'un processus,

exigent un positionnement du niveau d'interruption le plus élevé car elles correspondent, par exemple à des instants de manipulation de la pile. La figure n°IV.21 présente au long du code exécutable de l'ordonnanceur le niveau d'interruption du masque du registre d'état et donc les interruptions autorisées. L'interruption de niveau 7, reste cependant toujours en activitée.

Les fonctions de traitement des interruptions comportent toutes des segments de code interdisant la prise en compte d'une interruption. L'opération de lecture d'un caractère du port série sous interruption n'autorise pas d'autres interruptions que celle de son propre niveau, et l'opération de lecture (unique) du code télécommande interdit tout autre interruption.

Exceptions système: nature et traitement associés.

Un certain nombre de dysfonctionnements internes ont été prévus au niveau du processeur, faisant l'objet d'une hiérarchisation des priorités des traitements (réf [VIE]).

->reset

->erreur de bus: une telle erreur est détectée lorsque par le signal BERR\ qui signale une erreur de transfert de données (un défaut de "timing" des signaux intervenant au cours d'échange). L'exécution de l'instruction en cours est avortée, et le traitement de l'exception est lancée pour détecter le défaut si cela est possible. Cette interruption est plus particulièrement adaptée en phase de mise au point de l'ensemble matériel-logiciel, pour le développeur,pour reprendre le contrôle des opérations (sous émulation).

->erreur d'adresse: utilisable aussi bien en phase de développement qu'en phase d'exécution d'un logiciel, cette interruption détecte les accès mémoire irréguliers tels que l'écriture (données) ou lecture (instructions ou données) d'un mot ou d'un mot long à une adresse impaire. Le traitement consiste à réaliser le calcul d'une nouvelle adresse d'accès en utilisant un déplacement.

->instruction illégale: cette interruption entre en action lorsque le processeur ne reconnait pas le code de l'instruction à exécuter, par rapport au jeu d'instruction du processeur. Le traitement à effectuer dépend de la nature du code erroné. Un code système erroné implique la réinitialisation complète du système, alors qu'une erreur en code de processus provoque la désallocation de la mémoire correspondante, et le marquage du défaut en élément descripteur de contexte processus.

->division par zéro: elle intervient après détection d'une impossibilité d'exécution de l'instruction DIV ou DIVU par un diviseur nul conduisant à un résultat indéterminé. La réponse à apporter est la génération d'un message d'erreur et le branchement des opérations d'ordonnancement.

->interruption CHK: elle est activée à la suite de l'exécution d'une instruction CHK (test d'appartenance à un intervalle) et d'une réponse négative à celle-ci.

->violation de privilège: elle apparaît lorsque qu'une instruction privilégiée est exécutée et que le processeur est en mode utilisateur. La réponse à apporter est le passage en mode superviseur, puis la réexécution de l'instruction en cause si le processeur exécute une instruction du noyau du système ou de l'interpréteur de commande.

->interruption non-initialisées: c'est une interruption générée par un composant n'appartenant pas à la famille 6800 et non initialisée. De ce fait en phase de reconnaissance de l'interruption non auto vectorisée, le composant ne peut placer sur le bus de données le numéro de vecteur, ce qui implique cette exception matérielle spécifique. Le traitement à effectuer en conséquence est la réinitialisation de tous les périphériques non auto-vectorisés.

->interruption parasite: un signal d'interruption a été perçu et pris en compte par le processeur, et par manque d'accusé de réception sous forme d'un VPA\ ou d'un DTACK\ permettant de qualifier le type de vectorisation utilisé et de rattacher l'interruption à un

composant précis, celle ci est qualifiée de parasite et n'exige aucun traitement particulier. Le programme en cours est repris à l'endroit où il avait été interrompu.

Exception utilisateur: affectation et traitement

Les exceptions ou interruptions programmées logicielles sont utilisées dans le cadre des interactions entre couches pour effectuer une transition entre couches, pour utiliser une fonction appartenant à une autre couche selon le schéma d'interaction défini.

-les trappes #1, #2 et #3 sont réservées aux accès systèmes.

-les trappes #4 et #5 sont réservées aux fonctions de l'interpréteur de commande et de l'interface graphique.

VI.3 Les périphériques

Par principe d'indépendance du logiciel par rapport au matériel, les fonctions de gestion des périphériques n'appartiennent pas au noyau du logiciel mais elles sont intégrées dans la premiére couche de celui-ci. Le noyau est indépendant des périphériques, même si ceux-ci participent à la visualisation des résultats (processeur graphique), à la captation des informations provenant de différentes sources (RS232C, télécommande), et à la synchronisation de l'exécution des processus applicatifs.

Chaque interface est programmée, contrôlée par des fonctions, les pilotes (ou "drivers") de périphériques répartis en deux niveaux selon le degré de complexité réalisé et le langage de programmation utilisé. Les fonctions de base, généralement développées en assembleur, réalisent des opérations simples telles que le positionnement du curseur à l'écran ou la commande d'affichage d'un caractère pour le processeur vidéo. Les fonctions de deuxième niveau sont des fonctions évoluées écrites en langage C et agglomèrent souvent des fonctions de base. Ce genre de fonctions effectuent l'affichage d'un texte semi-graphique dans une fenêtre avec des couleurs de fond et de caractères différentes de celles du reste de l'écran.

En utilisant une structure de description par bibliothèques, on définit ainsi trois bibliothèques de programme de gestion des périphériques (bibliothèques graphiques, bibliothèque de communication du port série, bibliothèque des fonctions I2C).

Traitements logiciels liés au port série:ACIA 6850

L'ACIA 6850

L'utilisateur dispose fondamentalement de deux modes de fonctionnement pour émettre ou pour recevoir des données (ces deux modes sont plus particulièrement définis pour la réception des données):

-un mode fonctionnement sous interruption.

Il est utilisé pour éviter que le processeur ne perde son temps à attendre l'arrivée de données par le port série. Lorsqu'un signal d'interruption se présente le processeur suspend l'exécution de la tâche en cours et effectue le traitement concernant le port série et entre autre la lecture de l'octet de donnée disponible dans le registre de réception, seule opération permettant de dévalider le signal d'exception matérielle. La procédure de fonctionnement sous interruption est similaire en émission, mais présente dans notre cas moins d'intérêt. Ce mode de fonctionnement permet de capter des événements (informations) aussi bien aléatoires que déterministes et de façon quasi-instantanée.

-un mode de fonctionnement par scrutation

Ce mode de réception ou d'émission est simple de programmation est correspond bien à un système monotâche pour une application de transmission. Le processeur lit en permanence l'état du registre de réception des données et lorsque l'indicateur est positionné, le microprocesseur effectue la lecture puis reprend ses opérations de scrutation jusqu'à réception de la prochaine donnée.

Utilisation des différents indicateurs d'état:

Ces indicateurs servent au processeur en cas de dysfonctionnement lorsqu'une donnée ne correspond pas à celle attendue ou lorsqu'une opération de transmission n'est pas réalisable. Ces indicateurs sont au nombre de cinq.

L'indicateur (bit n°2) TDRE signale la perte de porteuse de données, ce qui ne permet aucune transmission. La tâche d'émission en cours doit être interrompue et reprise ultérieurement à son début lorsque l'indicateur est dévalidé. Il n'y a pas en principe de répercussion en réception, cependant ce port série dédié à une réception permanente d'informations signale à l'utilisateur le défaut constaté.

L'indicateur (bit n°4) signale une erreur de format, c'est à dire une non correspondance entre le format des données reçues et le format programmé au niveau du port. Une adaptation automatique par tests successifs doit permettre de résoudre le problème. La détection du défaut ne peut s'effectuer que par scrutation périodique de l'état de cet indicateur.

Le drapeau de surcharge (bit n°5) signale une désadaptation de vitesses entre la vitesse de transmission sur la liaison série et la vitesse d'échanges programmée au niveau du composant; là aussi la détection du dysfonctionnement ne peut se faire que par scrutation.

L'indicateur d'erreur de parité(bit n°6) porte sur une incohérence du schéma de contrôle par parité entre l'émetteur et le récepteur de données. Ce choix de contrôle implique une évolution de la taille de la trame d'acheminement d'une donnée.

Le réseau collectif domotique Médiabus

La gestion des échanges sur le réseau Médiabus est réalisée en alternance, à une vitesse de 19200 bits/s selon un protocole défini au CERLOR par Ms KOPP, LANCELOT. Ce protocole implique que le gérant du réseau interroge ou adresse chaque terminal ou équipement esclave (le terminal en est aussi un) périodiquement et attende durant un temps limité la réponse ou un accusé de réception de la part du terminal. L'adressage du terminal a pour objectif soit l'interrogation de l'équipement pour que celui-ci génère une réponse (par exemple la lecture immédiate de la consommation d'eau), soit le passage d'une commande à effectuer par l'esclave (qui doit générer en retour un accusé de réception), soit pour vérifier la présence et le bon fonctionnement de l'équipement, soit enfin pour consulter le terminal sur une éventuelle requête (accès à un service disponible sur le gérant ou un serveur) de celui-ci.

La sécurité du système repose sur l'interrogation périodique de tous les éléments du réseau. Lorsqu'un équipement ne répond pas il est réinterrogé immédiatement. Si aucune réponse n'est finalement reçue par le gérant, alors une procédure d'interrogation ultérieure est programmée avant de déclarer le boîtier en défaut.

Procédures d'échanges et de traitement des données entre le réseau et le terminal

Lorsqu'une trame d'informations est reçue par le terminal par le port série via le modem, les données sont stockées dans le tampon de réception. La routine de traitement de l'interruption est conçue de façon à être la plus courte possible car elle intervient durant le temps de traitement d'un processus interne et consomme une partie de la tranche de temps allouée. Il est nécessaire de corréler la durée de la tranche de temps avec la vitesse de transmission des informations sur le réseau externe. Considérons une tranche de temps (quantum) de 30 ms allouée pour l'exécution d'un processus, et la vitesse de transmission de 19,2 Kbits/s. Une interruption est générée par le port série toutes les 520 µs, ce qui correspond à 58 interruptions survenant durant cette tranche. Sachant que la durée de traitement de l'interruption est de 150 µs, la durée totale passée en interruption est de 8,7 ms, soit 30 % du temps total d'exécution du processus. Pour faire évoluer ce pourcentage, seuls deux paramètres interviennent; la durée de traitement de la routine d'interruption, et la périodicité de survenance de celles-ci (fréquence de transmission des données de la liaison série). L'amélioration de ce pourcentage peut être obtenu soit par augmentation de la périodicité des interruptions et donc diminution de la fréquence de transmission, soit par diminution de la durée de traitement de l'exception matérielle, seul variable sur laquelle il est possible d'agir. Encore cette seule action possible, trouve sa limite (une centaine de µs) dans le temps de prise en compte de l'interruption, de lecture de la donnée, d'un éventuel test, et du retour d'interruption. Avec une durée de traitement d'interruption de 103 µs, le pourcentage de temps consacré à la réception des données passe à 20%, ce qui est encore beaucoup; 10% serait une bonne valeur. Alors que faire?

A partir de la structure matérielle actuelle, la solution réside dans le déport de certains traitements hors de la routine d'interruption. Les traitements à opérer portent sur la trame de communication du réseau Médiabus, c'est-à-dire l'extraction, le test et la mémorisation des informations d'adresse, de service, de longueur de la trame informative et naturellement des données. Ceci étant réalisé, et comme le terminal est un équipement esclave, il doit nécessairement généré un accusé de réception compris dans un intervalle de temps défini par le concepteur du protocole de transmission sur le réseau. Les choix de gestion et donc de structure du logiciel portent sur le mode de traitement (traitement parcellisé ou global), sur les opérations réalisées au sein même de la procédure de traitement de l'interruption (nombre et nature des tests), et enfin sur le comportement à tenir lors d'acheminement d'informations ne concernant pas le terminal. Une solution extrême peut consister à capter et à stocker tous les octets reçus dans un tampon de réception circulaire.

Si on adopte une procédure de traitements parcellisés, la fonction de traitement de la trame renferme au moins une partie des tests à effectuer, voir même tous, et l'opération de stockage en mémoire. A chaque interruption sont réalisés de un à quatre tests (début, adresse, accès, fin) donc une partie des opérations de traitement. Cet ensemble constitue un prétraitement, car il ne peut être question d'y inclure la génération d'un accusé de réception ou même d'une réponse. Le temps passé dans cette phase est alors assez grand.

En procédant de façon globale on déporte le maximum d'opérations hors routine d'interruption, qui dans ce cas comporte un seul test de fin et le stockage en tampon de réception. Le test de fin de trame déclenche par sa détection, le lancement des opérations de déstructuration de la trame, de conservation et de stockage protégé des données en mémoire et ceci hors routine d'interruption. L'exécution de la routine d'interruption est alors rapidement exécutée, et le traitement des données comporte toutes les opérations d'extraction des informations utiles de la trame, le stockage éventuel et défini des données selon la nature du service en zones mémoire prédéfinies, et enfin la réponse du terminal.

Quelle que soit la structure de traitement retenue, il existe un impératif de temps de lecture du registre de réception du port série à respecter; 104 µs est le temps alloué pour effectuer complètement la phase de lecture du registre, qui comporte aussi la sauvegarde des registres internes du microprocesseur en pile, et le masquage de toute interruption. Ce temps correspond à la réception d'un bit de "stop" et d'un bit de "start". Si la lecture du registre est effectuée après cette limite, la donnée initiale présente en tampon, est perdue en tout ou partie par écrasement de celle-ci par les nouvelles données qui arrivent.

Les essais de réception et de traitement des données en provenance du port série n'ont été réalisés qu'avec une seule configuration de trame différentes. Une trame allégée de même structure que celle utilisée sur le réseau Médiabus, est proposée au terminal par un micro-ordinateur jouant le rôle de serveur.

La nature et le nombre de traitement à réaliser dépendent de la trame utilisée sur le réseau qui pourrait être différente de la trame Médiabus actuelle. Cependant on peut définir les opérations de base à effectuer pour tout protocole:

-détection du début de trame.
-reconnaissance dans la trame de la propre adresse du terminal.
-lecture du service faisant l'objet de la trame.
-enregistrement de la longueur de la trame informative et comptage du nombre d'octets reçus.
-stockage en tampon de réception.
-détection de fin de trame et lancement d'opérations complémentaires

Opérations complémentaires:

-l'ensemble des opérations de décodage et de protection des données (codage Hamming et CRC).
-transfert des données du tampon en mémoire sous forme d'enregistrement selon le service, puis éventuellement programmation d'un processus interne de traitement de ces données.
-génération d'un accusé de réception lorsque la trame ne porte pas une information de service ou de requête, ou génération d'une réponse complète avec génération d'une trame.

Traitement de la trame générée par un PC serveur

Cette trame a été définie à partir des travaux préliminaires de constitution de la trame de communication sur le réseau Médiabus (alors que celle-ci n'était pas encore constituée en totalité) pour tester la capacité du terminal à capter et à répondre à des informations. La trame de test, tout en étant de taille variable dispose d'octets de début (FB) et de fin (FF) constants.

Deux systèmes de traitement de la trame incidente ont été mis en place. L'un, décomposable en trois étapes, comporte la phase de lecture du port et la phase de prétraitement de la trame effectuée sous interruption. La troisième phase réalise la fin des traitements hors interruption. Le modèle représentatif est celui des traitements parcellisés. Il ne convient pas, du fait du temps d'exécution sous interruption qu'il requiert et les pertes de données occasionnées par bourrage. Son avantage étant de ne mémoriser que les données concernant le terminal.

L'autre mécanisme testé, ne réalise que l'opération de test de fin de trame (le FF) et le stockage des données en tampon (qui doit être de taille suffisante) sous interruption. Il n'y a plus d'opérations de prétraitement, et le test de détection de fin de trame est utilisé pour lancer à la suite de la dernière interruption le traitement global et déporté de la trame. L'inconvénient d'un tel mécanisme est le non filtrage des informations n'intéressant pas le terminal. Le taux d'occupation du processeur par rapport au port série est élevé et constant. Un autre avantage de ce mécanisme est son mode d'intervention . Le traitement de la trame est déporté et répond à la condition de modularité. Il est ainsi plus facile de programmer une commutation de traitement entre trames différentes.

Pour limiter le taux d'occupation du processeur en interruption lorsque la trame ne porte pas d'informations concernant le terminal, un comportement type est à définir. Soit il est décidé que toutes les informations sont captées et étudiées, soit on établit une porte d'entrée au stockage, soit le mode de réception sous interruption est désactivé si le drapeau d'entrée et l'adresse transmise sont reconnus.

Si toutes les informations sont captées et étudiées, tous les processus sont également perturbés, mais par contre aucune trame concernant le terminal ne peut être perdue. On assure le taux maximal de détection des messages intéressant l'équipement.

Par contre en choisissant de déprogrammer le mode de réception sous interruption on réalise une pleine utilisation de la tranche de temps CPU par les processus non interrompus. La déprogrammation étant effectuée à la suite de la lecture du début de trame, des octets de service, et de longueur de trame et enfin de la non reconnaissance de l'adresse acheminée. La durée de déprogrammation étant évaluée à partir du nombre d'octets comportant la trame ignorée. Tous les débuts de trame sont lus et étudiés. La reprogrammation du mode interruption est conditionnée par la détection de la fin de trame. Un problème de resynchronisation peut apparaître, lorsque le terminal juge que la ou les trames ne le concernant pas sont acheminées.

En utilisant un système de porte, on définit une solution intermédiaire. Le mode interruption n'est jamais dévalidé, mais lorsque le début de trame et l'adresse sont reconnus (donc testés) une variable est positionnée à un et la réception se déroule tant que cet indicateur reste en cet état. Son passage à zéro qui correspond à la détection du drapeau de fin de trame, marque la fermeture de la porte. A chaque interruption au moins deux tests sont effectués; le test de début et le test de la porte, mais le traitement est réduit lorsque la porte est fermé.

Sémaphore

Que ce soit pour l'émission ou la réception de données, deux sémaphores protègent l'utilisation unitaire du port série et des deux tampons , contre toute utilisation concurrente et

destructrice de la part de deux processus. Un sémaphore concerne l'ensemble réception (code et tampon) SEM_RS232_R, et un autre SEM_RS232_E le code des fonctions et le tampon de réception.

Traitements logiciels liés à l'horloge système

Le PTM 6840

C'est l'élément important de régulation de l'exécution des processus puisque la méthode utilisée pour interrompre un processus dans son exécution est celle de la préemption.

Au cours de la phase d'initialisation des composants cette horloge est programmée pour fournir périodiquement un signal de fréquence et de rapport cyclique (état 0/état 1) égal à 1/30 environ sur l'une des trois sorties générant un quelconque signal. Ce signal est celui appliqué sur l'entrée d'interruption de niveau de priorité 2, et une telle valeur du rapport cyclique s'explique par le rôle de métronome que l'on fait jouer à cette horloge. L'interruption horloge ne peut rester pendante, c'est à dire que ce signal ne peut rester à zéro tant qu'un registre quelconque ne soit lu pour que l'interruption soit dévalidée. D'autre part un tel rapport cyclique ne peut être obtenu en utilisant la sortie d'interruption du 6840.

Le traitement de l'interruption de l'horloge système est simple en apparence. Lorsque le signal est pris en compte le traitement consiste d'abord à se prémunir contre toute interruption de niveau inférieur ou égal à cinq par masquage, puis à sauvegarder le contenu de tous les registres internes du µp en pile locale du processus interrompu, et enfin à sauvegarder en élément descripteur correspondant les informations nécessaires à la reprise ultérieure d'exécution du processus, c'est à dire les pointeurs d'accès courant aux différentes zones et certaines informations d'état du processus mis à jour.

Avant l'entrée en phase d'ordonnancement pour l'exécution ou la reprise d'exécution d'un autre processus, il faut vérifier si l'interruption est intervenue en zone de code partageable. Si ce n'est pas le cas, c'est à dire si l'interruption est survenue en zone critique, il est nécessaire de reprendre l'exécution du processus jusqu'en fin de la zone critique concernée et seulement elle. Un indicateur est alors positionné marquant la survenance d'une interruption au cours du traitement d'une zone critique. Il n'y a pas à proprement parler de réallocation de temps au processus en cours. Pour éviter un blocage, ou au minimum un dysfonctionnement vis à vis d'une ressource non partageable, on termine l'exécution de la zone critique. En fin de zone critique, une fonction teste l'indicateur. Une valeur positive de cet indicateur, marque alors le déroutement en phase d'ordonnancement. Ainsi le traitement programmé par l'horloge système est effectué de façon différée.

La structure de génération programmée d'un signal périodique, connecté à une entrée d'interruption auto-vectorisée permet de faire l'économie sur le plan programme, en routine d'interruption du réarmement de l'interruption avant toute nouvelle utilisation. Ce qui

représente un gain de temps appréciable. Par recours à une sortie horloge, il peut aussi être possible de modifier la fréquence de génération d'un signal d'interruption en cours de fonctionnement du logiciel, soit à l'initiative de l'utilisateur (ou du gérant de réseau) ou soit par le système lui-même en suivant l'évolution d'un paramètre significatif.

Traitements logiciels liés à l'interface I2C

Il est nécessaire avant toute tentative de programmation d'un composant I2C, d'initialiser le composant réalisant l'interface entre le bus parallèle et le bus série I2C, le PCD8584.

L'interruption I2C est multisource, c'est à dire que plusieurs composants I2C peuvent générer une interruption sur la même entrée, ce qui implique un ensemble d'opérations complexes et ordonnées. Après la phase de sauvegarde des registres internes du microprocesseur (en pile locale), et la phase de sauvegarde et de mise jour du contexte du processus, une phase de scrutation séquentielle des registres de contrôle des composants I2C est entammée: le PCD8583 utilisé en calendrier/horloge/alarme, puis le PCD8583 compteur d'événements. Si aucun de ceux-ci n'a appelé l'attention du processeur il reste le récepteur de télécommande dont la probabilité de produire une interruption est la plus importante.

Si l'interruption a été générée par une alarme, il est impératif de réarmer l'alarme, ce qui se traduit par dévalidation du signal d'interruption (remise à 1 de la ligne IPL1). Il est possible alors de reprogrammer pour une nouvelle alarme, les registres adéquats (deuxième groupe de registre du PCD8583) par les caractéristiques de l'alarme contenues dans un tableau d'alarme géré comme une file d'attente selon la règle horaire. La dernière opération consiste alors à remettre en route de compteur du calendrier. Une interruption d'alarme est générée par le composant lorsqu'il y a coïncidence entre les registres d'alarme et les registres d'horloge.

Avec une interruption générée par le compteur d'événement, l'ensemble des traitements à effectuer est similaire et porte seulement sur des registres différents.

A la réception d'un code télécommande par IR le transcodeur SAA3028, génère une interruption, qui n'est dévalidée que par lecture via le bus I2C (par le contrôleur de bus I2C (ou par tout composant maître)) du code. Cette lecture sur le bus est commandée par le microprocesseur. Le format de transmission Infra Rouge est du type RC5, comprenant quatre octets, dont un seul, le dernier porte l'information utile. La phase de lecture du code implique de régénérer la trame de lecture sur le bus série de quatre octets par le maître.
A la suite du stockage en tampon du code télécommande, on fait appel à l'interpréteur de commande dont le rôle est de lancer les actions correspondantes. L'appel peut être

explicitement inclu dans la routine de traitement, ou suivre de peu la fin de l'interruption. Avant le retour au processus, le contexte et l'environnement de processus est restauré.

Traitements logiciels liés à l'interface graphique

Aucun traitement sous interruption n'est à réaliser pour l'interface de visualisation, et c'est là plus qu'ailleurs que se trouve représentée la structuration en deux couches logicielles des programmes de gestion; les fonctions de base opérant sur un caractère ou sur un élément, et les fonctions évoluées de présentation tel que le tracé d'une fenêtre.

Exemples de fonctions élémentaires:

-programmation d'un jeu de caractère alphanumérique.
-programmation d'un jeu de caractères semi-graphique.
-affichage d'un caractère.
-programmation des couleurs de caractère et de fond.
-initialisation des registres du processeur vidéo.
-effacement de l'écran.
-commande de clignotement d'un caractère.
-commande d'inversion vidéo d'un caractère.
-effacement d'un caractère.
-codage de caractère UDS pour icône.

Exemples de fonctions évoluées.

-affichage d'une ligne de caractères.
-affichage d'un icone quatre caractères.
-tracé d'une fenêtre.

Les fonctions élémentaires comportent toutes des zones critiques et utilisent un sémaphore simple mais spécifique sous forme d'une variable globale SEM_VDP.

VI.4: L'interface réseaux de communication

Les systèmes de communication et notamment les systèmes téléinformatiques sont très complexes quand à leur structure (étoile, boucle) et aux équipements qu'ils mettent en oeuvre (modems, multiplexeurs, déconcentrateurs...). Un réseau, à la différence d'un bus, doit être de par sa taille géré par un système d'exploitation spécifique pour assurer le bon fonctionnement de celui-ci, doit éviter si possible et résoudre les problèmes de conflit d'accès, doit régulariser

les flux d'information, doit veiller au bon acheminement des informations et enfin détecter les dysfonctionnements matériels. L'ensemble des tâches effectuées par le système d'exploitation d'un réseau est identique à celui d'un système d'exploitation de microordinateur. Elle doit assurer la bonne gestion des différents éléments qui le constitue. Il n'est pas question ici de décrire un système d'exploitation de réseau, mais simplement de voir dans la structure du logiciel que je conçoit comment le réseau intervient. Une normalisation des systèmes de communication permet de décrire en sept couches la structure logicielle et matérielle d'un réseau. Tout matériel de communication évolué doit dans son fonctionnement et selon sa nature obéir à cette structure de fonctionnement.

Généralités sur les réseaux

L'une des similitudes de fonctionnement avec un bus de communication, réside dans le mode de transmission des informations, en mode synchrone ou en mode asynchrone.

Pour assurer la communication uniquement sérielle entre deux éléments d'un réseau on utilise une seule voie physique. Tout support physique quel que soit sa nature (câble coaxial, fibre optique, paire téléphonique...) présente par ses caractéristiques des défauts, des contraintes et des limites de fonctionnement en fonction de son utilisation. Le support est choisi en fonction de la nature, de la représentation des informations à transmettre (tansmission analogique/numérique, modulée/en bande de base, niveaux de tension, type de modulation, type de perturbation...)
Les données transmises peuvent être altérées, perdues, dupliquées, parasitées selon l'usage fait de ce support (désadaptation des lignes->duplication et phénomène d'écho).

Pour remédier aux défauts survenant dans la sphère d'utilisation des supports, il est nécessaire:
-de structurer l'information transmise, en informations de contrôle et en informations de données: on définit ainsi une ou plusieurs trames de communication.
-de tenir compte des altérations possibles sur ces deux types d'information.

Pour détecter et corriger les erreurs de transmission, on instaure un certain niveau de redondance des informations, par utilisation de codes cycliques pour reformater les données et d'identificateurs de messages qui viennent s'insérer dans la trame. Le but étant par un système de codage adapté, de rendre compte le plus précisément possible du ou des altérations survenues et de permettre ainsi leur correction. Un système de codage/décodage et de correction fonctionne de façon permanente et comme aucun mécanisme n'est jamais entièrement fiable, il introduit lui aussi des erreurs en détectant et corrigeant de fausses erreurs.

Le codage

Le choix du code est fonction des données (type, importance) de la taille maximum de la partie informative de la trame, et de la vitesse de transmission. Le dernier paramètre à prendre en compte porte sur les caractéristiques propres de chaque codage; le nombre maximal d'erreur détectables et corrigible, le taux de fausses erreurs introduites par le décodage.

La correction

La stratégie d'utilisation de l'ensemble codage/décodage dépend de l'ensemble du système de transmission utilisé. Un système simple:a chaque erreur détectée une alarme peut être signalée. La détection d'une erreur est suivie d'une correction. La correction est:
-soit directe (Correction d'Erreur Directe)de façon automatique par le décodeur lui-même.
-soit par retransmission (ARQ). Lorsque le décodeur est inefficace (nombre trop élevé d'erreur), il faut alors retransmettre tout ou partie des informations.
Selon la stratégie choisie, on retransmet:

-avec arrêt d'attente (méthodes BSC,TMM).

-de façon continue (HDLC utilisant un REJ).

-avec répétition sélective (HDLC utilisant un SREJ).

Correction d'erreur par retransmission

Retransmission avec arrêt et d'attente:

L'émetteur après transmission d'un bloc de données (tout ou partie de la trame) attend un accusé de réception positif (bonne réception) ou négatif (erreur détectée en réception). Dans ce dernier cas, il réémet le bloc ou la trame. Une temporisation ("time out") laisse au destinataire un délai de réponse en cas de surcharge de celui-ci. Une défaillance matérielle du destinataire peut ainsi être détectée, et provoque l'arrêt du mécanisme de réémission. Cette méthode est celle retenue dans le système Médiabus.

Retransmission en continue:

La transmission se poursuit bloc par bloc sans arrêt, jusqu'à ce que le destinataire par un accusé de réception négatif, signale la détection d'une erreur. Tous les blocs de données à partir du bloc portant l'information erronée sont réémis.

Retransmission continue à répétition sélective:

A la suite de la réception d'un accusé de réception négatif, seul le bloc portant l'information erronée est retransmis. Cette méthode ainsi que la précédente nécessitent une liaison bidirectionnelle simultanée.

L'opération de correction est permanente et trouve sa limite dans l'introduction d'erreurs (de correction) par la détection de fausses erreurs.

Le choix de la méthode de correction est imposé par les contraintes techniques. Il est possible d'estimer les taux d'erreur admissibles et prévisibles mais c'est la réalité,c'est à dire la qualité de la transmission des informations qui porte le verdict sur les options prises.

Ces mécanismes de codage/décodage, et correction d'erreurs dans la transmission des informations constituent une procédure de communication. Ils ne se limitent pas à ces aspects, et se trouvent caractérisés par d'autres éléments:

-le sens de transmission des données: unidirectionnel (simplex) ou bidirectionnel (duplex).
-l'organisation des échanges en mode bidirectionnel: simultané (full duplex) ou alterné (half duplex).
-le débit binaire utilisé sur la ligne.
-les temps d'établissement et de libération des composants en commutation lors des transferts de données.
-le temps de propagation.

En résumé la procédure de communication prend en charge:
-l'ensemble des opérations de transfert.
-la structuration des données: mise en forme, reconnaissance des données informatives.
-la détection et la correction des erreurs par réémissions.
-la gestion des échanges; la génération de différents types de messages (interrogation d'un terminal, envoie de messages), d'accusés de réception positifs ou négatifs.
-l'adressage individualisé ou collectif des équipements placés sur le réseau.
-la gestion des temporisations.
-une gestion adaptée des différents équipements placés sur le réseau (possibilité de déconnexion de certaines voies ou branches).
-un dispositif de gestion des dysfonctionnements matériels sur le réseau (terminal esclave ne cessant d'émettre en permanence hors sollicitation de la part du gérant).
-un dispositif d'accès réglementé au réseau pour éviter les conflits d'accès.

Les procédures se classent en trois familles selon les équipements connectés, l'implantation des mécanismes de gestion du réseau et la taille de celui-ci. Les procédures dites minimales sont connues pour des systèmes très centralisés, de matériels homogènes, où les terminaux sont de simples esclaves (réseau minitel). Un autre type de procédure défini pour des liaisons entre ordinateurs de status différents, laisse une plus grande marge d'initiative aux terminaux, et permet des vitesse de transmission plus élevées. La dernières famille concerne les réseaux d'interconnexions d'ordinateurs ou d'équipement à intelligence répartie. Ces réseaux sont de taille évolutive et utilise des liaisons spécialisées en bidirectionnel simultané.

Bien d'autres éléments caractérisent un réseau tel que le type de multiplexage des données, les canaux d'émission et de réception...et naturellement leur architecture.

L'ensemble des tâches effectuées par la procédure fait intervenir différentes fonctions que l'on peut répartir en couche selon le modèle OSI organisé et proposé par l'Organisation Internationale de Standardisation: ISO. Ce modèle comporte sept couches:

-1)La couche physique qui assure le transport de l'information.

Les caractéristiques physiques de la liaison et le formatage des données font l'objet d'une normalisation (exemple avis V24).

-2)La couche liaison renferme les fonctions de réception et d'envoi des données ainsi que les fonctions chargées du codage/décodage des informations et de toutes les fonctions ayant pour but de limiter les erreurs de transmission. Cette couche est différente selon la hiérarchie de l'équipement; maître ou esclave, terminal ou gérant.

-3)La couche réseau est implantée dans les unités de gestion principales du réseaux (les noeuds) pour effectuer le routage des informations suivant le chemin le moins encombré.

-4)La couche transport renferme les fonctions de découpage ou de réassemblage d'ensemble d'informations (un fichier par exemple) en blocs pour le transport; chaque trame ayant une longueur maximale.

-5)La couche session contient les fonctions de gestion des accés simultanés entre différents équipements en communication.

-6)La couche présentation: est responsable de la présentation (types de caractères reconnus) et du stockage des données.

-7)La couche application:

Tout équipement connecté à un réseau est connu en tenant compte de cette structure en reprenant une ou de plusieurs couches de cet ensemble. Ce modèle permet de définir rapidement le système d'exploitation du réseau ainsi que le niveau d'intervention des fonctions constituant le système de gestion embarqué de l'équipement.

Tout équipement ne renferme pas ces sept couches. Le terminal domotique n'a pas de couche réseau, ni de couche session.

Les fonctions de gestion réseau implantées au sein du terminal

Nous avons vu que pour tout périphérique, les fonctions de gestion de celui-ci était regroupées en deux ensembles, pour les fonctions de base et les fonctions évoluées. Le port série RS232 obéit à cette règle de structuration, mais l'ensemble de toutes ces fonctions ne peut assurer d'une part que la réception, l'envoi de données isolées ou groupées et leur stockage en tampon, et d'autre part la programmation des différents paramètre de transmission en registre de contrôle ainsi que la détection et la correction de défaut de programmation ou de transmission au niveau du port (vitesse ou format de transmission inadapté, effet de bourrage).

A ce niveau le terminal est incapable de traiter un ensemble de données dans son ensemble et de réagir en fonction de ce qu'elles renferment; c'est à dire vis à vis du réseau reconnaître la nature de l'information. Est-ce un message à caractère général n'impliquant de la part du terminal que la génération d'un accusé de réception? Ou plutôt est-ce une demande de renseignement sur une donnée nécessitant l'envoi d'une réponse complète par le terminal? Cette réponse pouvant être soit immédiate sous la forme d'une trame de réponse à générer durant le délai de "time out", soit être différée lors d'un prochain échange. Dans ce cas le terminal doit produire un accusé de réception.

Faisant toujours partie de la couche liaison (comme les précédentes) les fonctions de détection et de correction des erreurs de transmission en réception par décodage Hamming des champs de données, et de contrôle, sont complétées par les fonction de codage des informations produites par le terminal dans la génération d'une réponse.

Les fonctions des couches transport et session ne sont pas implantées en raison de l'inexistence de services exigeant l'emploi de plusieurs trames de 256 octets de données, ainsi que le dialogue entre périphériques esclaves.

Les couches présentation et application sont elles bien présentes mais propres au terminal.

Les fonctions de gestion du protocole sont intégrées dans la deuxième couche avec les fonctions de gestion de l'interface graphique (gestion d'écran). L'équipement pour être polyvalent, sur le plan des possibilités d'accès réseau devrait en l'absence (à ce jour) de normes, soit intégrer en ROM différents protocoles de communication, soit donner à l'utilisateur un outil de définition de protocole pour son terminal, en fonction du réseau sur lequel il est connecté. La première solution étant la plus réaliste.

Les opérations de réinitialisation

Les opérations de réinitialisation sont constituées de deux groupes selon qu'il s'agit de solutionner un dysfonctionnement au niveau du système multitâche, ou qu'il soit nécessaire de régler un problème touchant à une périphérique.

VI.5: La prévention et la correction des dysfonctionnements du terminal et de son logiciel

Dérives logicielles

Les dysfonctionnements touchant le système sont du type logiciel, et portent sur des atteintes des différents éléments de l'exécutif, les piles, tables et tableaux supports de gestion. Il ne peut être question de définir exactement la cause d'un dysfonctionnement (souvent du à un débordement de pile catastrophique), qui n'est d'ailleurs détecté que lors de l'utilisation de ces ensembles de références du système.

Il ne s'agit pas d'implanter dans le logiciel des points de test que l'on consulte régulièrement comme cela se passe lorsqu'on teste le matériel en des endroits stratégiques. La détection des dysfonctionnements se fait incidemment lors de l'accès à ces tables de mémorisation structurées utilisées pour l'exécution des processus; accès aux files d'attente, au tableau descripteur de contexte des processus référencés, au tableau descripteur d'allocation mémoire, à la table de correspondance icône-processus-fenêtrage. Toutes ces tables sont bornées par des indicateurs de début et de fin, dont la seule utilité, est de cerner l'ensemble structuré. Le système de détection des dysfonctionnements utilise ces bornes. Ainsi pour les files d'attente la non reconnaissance en des endroits fixés des octets de début et de fin lance la procédure de détection du défaut. Le défaut de non reconnaissance de l'octet de fin de file, se corrige par positionnement d'autorité d'une fin de file.

Le niveau de correction est variable, et il peut aller jusqu'à la réinstallation complète de l'exécutif multitâche, (les périphériques ne sont pas concernés). Par exemple la non

reconnaissance des bornes extrèmes d'une file ou d'un tableau provoque la réinstallation de cette file. La structure altérée est la seule à être corrigée.

Après détection d'un défaut toutes les structures sont testées. Une table de défaut enregistre les codes correspondants aux éléments altérés et le mécanisme de réinstallation opère en fonction des résultats de ce tableaux.

Le mécanisme de détection des dysfonctionnements ne peut malheureusement vérifier l'intégrité de ces tableaux quand à leur contenu.

La réinstallation d'un ensemble structuré a des conséquences variables selon l'altération de celle-ci.

Les files d'attente:

Pour une file d'attente, il faut éviter de perdre le contenu de cette file. Cela peut se faire, par vérification de la cohérence des processus référencés en descripteur. La non reconnaissance d'une structure numéro de processus et numéro d'enregistrement, provoque l'élimination des deux octets référencés. Le contenu de la file peut ainsi être vérifié.

Le tableau descripteur d'allocation mémoire

Par contre l'altération du contenu du descripteur d'allocation mémoire, ne peut être détecté du fait de son fonctionnement dynamique. La majeure partie des données présentes dans ce descripteur ne sont pas dupliquées ni redondantes (déplacements d'adressage indirectes). Une solution aléatoire pourrait être développée à partir de tests de reconnaissance de marques de début et de fin d'enregistrement en mémoire et le marquage dans chacun de ceux-ci des numéros d'enregistrement et de processus correspondant. Cette précaution permettant en cas de destruction totale ou partielle du tableau descripteur de reconstruire celui-ci, par une fonction spéciale à développer.

Par contre l'altération du contenu du descripteur d'allocation mémoire, ne peut être détecté du fait de son fonctionnement dynamique, en cas de disparition de ces marques.

Le tableau descripteur de contexte processus

Lors de l'installation des éléments de l'exécutif, les éléments descripteur de contexte des processus résidents (père) sont transférés en mémoire vive. Une altération de ces éléments n'aura pour conséquence que leur réinstallation à partir de la ROM.

Par contre toute altération des éléments descripteurs de contexte des processus importés est catastrophique. Là aussi une solution empirique (non développée) consisterait à dupliquer et à conserver à l'extrémité de la pile locale l'élément descripteur correspondant.

Dysfonctionnements matériels

De la panne franche d'un composant à un dysfonctionnement (VDP ou composant en attente infinie, horloge désynchronisée) souvent la résolution du problème impose une solution matériel. Avant de signaler à l'utilisateur le défaut, une tentative de solutionnement se fait par logiciel. L'idéal étant de disposer d'un mécanisme de réinitialisation matériel pour chaque interface, et non pas seulement d'une initialisation générale.

Dans le cas du VDP, qui est un périphérique lent, la majorité des blocages se sont produits au niveaux de la fonction d'attente intervenant entre différentes opérations de programmation des registres internes aux composants. Cette fonction est complétée par un mécanisme de comptage des tests de disponibilité (du VDP) effectués permettant de réaliser une procédure d'échappement et d'enchaîner une reprogrammation des registres de base du composant (R0,R1,R2,R3). Cette méthode ne constitue pas pourtant la solution à tous les blocages constatés, et la seule alternative actuelle est de réinitialiser l'ensemble du système.

l'horloge système est un composant sur lequel on réalise peu d'opérations (en phase d'initialisation des périphériques) car il est utilisé en générateur de signaux. Les quelques problèmes constatés l'ont été au démarrage du système. Le défaut est détectable et il suffit alors de reprogrammé le temporisateur.

Le port série RS232C, ne pose en apparence aucun problème de programmation dont la procédure est connue ainsi que celle de correction des défauts de transmission.

Le contrôleur de bus I2C, actuellement ne pose pas de problème de programmation de ses registres, mais il est nécessaire cependant de faire attention à ne pas perturber de façon anarchique la transmission sur le bus. Une transmission sur le bus I2C doit être impérativement terminée.

VII INTERFACE DE COMMANDE GRAPHIQUE ET PROCESSUS APPLICATIFS

VII.1: L'interpréteur de commande et le gestionnaire d'écran

Notions élémentaires d'ergonomie

Il s'agit de définir ici, les principes fondamentaux de conception de la représentation externe qui correspond à ce que voit l'utilisateur de l'application interactive (et éventuellement du système de gestion de la machine), puis de concevoir cette interface.
Rappelons que le représentation externe intègre:

-la représentation conceptuelle (définie et réalisée au préalable).
-l'ergonomie et la psychologie cognitive.
-les contraintes techniques du matériel (résolution, couleurs, gestion de l'écran).

La psychologie cognitive

La psychologie cognitive s'intéresse aux différents modes de perception des informations par l'homme dans son environnement. Elle illustre son utilité dans le cadre de la représentation externe en fournissant des données précises sur les mécanismes de perception (et les limites de celle-ci) par l'homme d'une ou d'un ensemble d'informations.

Mémoire à court et long terme

La perception de notre environnement fait intervenir cinq sens différents et complémentaires. Les informations collectées sont stockées soit en mémoire à court terme, soit en mémoire à long terme, selon le type de données le processus de mémorisation et d'autre facteurs psychologiques.

La mémoire à court terme est utilisée pour la perception d'évènements évolutifs nécessitant un traitement ou une action immédiate.

La mémoire à court terme est une mémoire mettant en oeuvre un mécanisme d'acquisition rapide mais limité en capacité. Dans une image il n'est possible de prendre en compte que sept évènements (ou unités d'information significatives, ou items) pendant un délai de deux secondes. Après deux secondes l'information s'altère rapidement et cette

altération peut s'accélérer si d'autre informations arrivent durant le processus de mémorisation et interférer avec les premières.

La mémoire à long terme correspond plus à un mécanisme d'acquisition de règles, de données, c'est-à-dire un mécanisme d'apprentissage.

La mémoire à long terme nécessite un effort de mémorisation et donc de concentration qui nous rend indisponible momentanément pour percevoir d'autres évènements.

Les phénomènes de passage des informations de la mémoire court terme à la mémoire long terme, ou celui de mémorisation directe en mémoire à long terme est favorisé si le cadre de présentation des données est déjà connu. Ceci met en oeuvre un raisonnement par analogie.

Ce mécanisme est d'autant plus utilisé, et conduit à des automatismes (permettant d'exécuter des tâches très rapidement et avec une mobilisation minimale de ses processus conscients) que la personne utilise, pratique de façon répétée une structure bien définie.

Pour faciliter le mécanisme d'apprentissage et donc d'acquisition d'automatismes de manipulation, une structure de présentation a été définie avec un souci de simplicité qui se traduit aussi dans l'utilisation d'un nombre très limité de touches (cinq ou six) de fonction définies et invariables.

Dans le cadre de notre application, les mécanismes de perception et d'acquisition en mémoire visuelle à court et long terme ont un impact important sur l'ergonomie du logiciel, par la définition de paramètres, de contraintes de conception de l'interface graphique.

De façon générale, lors de la conception d'interfaces les ergonomes prennent en compte sept aspects, ou paramètres non homogènes entre eux mais ayant des impacts complémentaires:

-le séquencement des opérations.
-le langage d'interaction.
-les dispositifs d'entré.
-les dispositifs de sortie.
-le temps de réponse.
-le guidage, l'aide à l'utilisateur.

1) Le séquencement des opérations:

Lors de la conception du logiciel d'interfaçage, il est nécessaire de prévoir un certain nombre d'applications en considérant, les flux d'information entre interfaces et

applications, la représentation des résultats, la structure de l'application et d'éventuels aléas de fonctionnement.

Le séquencement des opérations du logiciel recouvre différents aspects de manipulation du logiciel:

->le type d'enchaînement des actions:

Libre:à la guise de l'utilisateur(pas d'interdit de commande).

Guidé: l'utilisateur dispose d'un nombre de commande évolutif en fonction de sa position dans le logiciel.

Automatique:l'utilisateur n'intervient que pour indiquer un choix.

->les possibilités d'enchaînement variées selon l'expérience de l'utilisateur:

Raccourcis de commande.

Représentation.

->Les différents modes d'intervention:

->les possibilités: -de quitter définitivement ou provisoirement une application.
-d'annuler le travail effectué sur une application.
-de reprendre une application en cours pour la poursuivre.
-de retour à des opérations ou à des choix précédents.
-d'inversion des opérations.

Ces propriétés concernent aussi bien les dispositifs d'entrée que les dispositifs de sortie (graphique) des données, messages et guides à l'utilisateur.

2)Le langage d'interaction:

Le langage d'interaction est l'outil qui va permettre à l'utilisateur d'exprimer au moyen d'un vocabulaire et d'une syntaxe les opérations qu'il désire faire effectuer par l'équipement.

Par le choix du mode de commande et de représentation choisi pour le terminal, et en utilisant conjointement des pictogrammes et des menus déroulants, les problèmes de vocabulaire et de syntaxe sont réduits à leur plus simple expression.

Les pictogrammes et menus déroulants sont très intéressants car sans ambiguïtés, simple d'apprentissage et de mise en oeuvre par l'utilisateur (même novice). Par un seul item il est facile de commander l'exécution d'une action ou d'un groupe d'actions complexes.

Le principe d'homogénéité impose que la représentation des actions et des processus se fasse sous forme de pictogrammes et menus déroulants s'inscrivant dans des zones géographiques prédéfinies, et ceci pour toutes les applications (voir fig n°VII.1).

3)Les dispositifs d'entrée:

L'action de l'utilisateur, sur le terminal ne consiste pas seulement déplacer un curseur, ou à effectuer et à valider des choix de processus, de paramètres parmi ceux qui sont proposés, mais aussi à transmettre des données alphanumériques.

Le terminal domotique ne disposant pas de clavier mais d'une télécommande au nombre de touches limité, un problème particulier se pose pour l'entrée des données alphanumériques dans le cadre de service tel que la messagerie.

Une solution déjà utilisée sur certains équipements de mesure (analyseur logique) consiste à représenter à l'écran un clavier. Le choix des touches se faisant par positionnement du curseur par les quatre touches de déplacement et la validation du choix par une cinquième affectée à cet effet.

Pour des applications impliquant l'écriture de texte étoffés (>50 caractères, application en messagerie) les mots et expressions les plus courantes seront à choisir et à valider dans un dictionnaire.

Pour avoir tester cette approche, elle n'est pas la plus adaptée car trop éloignée de notre pratique habituelle d'écriture d'un texte.

Des casque HIFI, des télécommandes de magnétoscope complexes utilisent la liaison infrarouge pour acheminer les commande utilisateur à l'équipement principal; l'autre solution serait de réaliser un clavier alphanumérique (40 touches) de télécommande IR de taille réduite mais permettant d'entrer des textes dans le terminal.

4)Les dispositifs de présentation:

A la suite d'expériences portant sur les aires de perception visuelles privilégiées, il a été montré que les deux modes de perception, privilégiées et uniformes sont liés à deux modes d'exploration de l'écran (systématique et sélective) et en fin de compte de la connaissance que l'utilisateur a du logiciel.

En recherche systématique l'utilisateur scrute tout l'écran à partir du coin supérieur gauche, puis cette recherche se poursuit dans la zone centrale et se termine au bas de l'écran. Les éléments placés dans les extrémités droite et gauche sont mal perçues.

En recherche sélective l'utilisateur cherche une structure de représentation déjà connue et à l'intérieur de celle-ci les informations qui l'intéressent particulièrement à des positions prédéfinies. Aucune information n'est ignorée mais certaines sont sélectionnées

prioritairement. Aux extrémités de l'écran les limitations de perception sont plus rapidement atteintes .

Cette recherche conduit à une manipulation, à une utilisation plus rapide, plus automatisées, ce qui conduit à une standardisation du positionnement des différents types d'éléments et donc des réactions. Cette structure "normalisée" apporte à l'utilisateur un confort de travail, même si la contrepartie en est une certaine uniformisation des interfaces.

La présentation des données peut se faire de deux façons différentes:

-en mode question/réponse: (réponse oui/non ou similaire). Aucune latitude d'évolution n'est laissée à l'utilisateur et ce mode est à utilisé en guidage total de celui-ci par le système.

-en mode par remplissage de forme. Plus souple il permet à l'utilisateur d'entrer des données dans un ordre différent de celui proposé, mais impose le cadre général de la saisie.

L'interpréteur de commande du terminal

L'interpréteur de commande est un élément essentiel dans la chaîne de commande d'exécution de processus ou d'actions. L'interpréteur doit assurer la coordination entre différents gestionnaire; gestionnaire d'écran, gestionnaires des périphériques, gestionnaire d'exécution des programmes utilisateurs.

Le rôle essentiel de l'interface graphique est de nous proposer un outil de visualisation et de manipulation agréable et facile d'utilisation. Cette interface graphique travaille sous le contrôle étroit de l'interpréteur de commande qui vérifie la pertinence de l'action correspondante demandée à partir d'un code de commande reçu, et lance l'exécution de celle-ci. L'interpréteur effectue deux vérifications préalables avant de choisir l'action à effectuer. En premier est effectué le test de validité du code reçu (reconnaissance d'un code prédéfini), puis vient le test d'accessibilité à la fonction à l'opération demandée qui peut provisoirement ne pas être réalisable. Une opération peut ne pas être réalisable du fait de l'incompatibilité de fonctionnement avec une opération en cours, ou par l'incohérence du type d'action et son environnement. Ce point sera revu en détail ultérieurement. Une signalisation (en grisée) est couramment utilisée dans les interfaces graphiques pour signaler leur indisponibilité provisoire (Windows, Macintosh). Enfin dans l'ensemble des opérations, actions et commandes disponibles l'interpréteur effectue celle qui correspond au code reçu et validé.

Le choix d'utilisation de cinq touches de la télécommande seulement implique un mécanisme spécifique de gestion du code télécommande. Ce choix d'un nombre réduit de touches est fait dans un souci de simplicité pour l'utilisateur qui n'aura plus à vérifier la signification de la touche. Ne subsistent que quatre touches de direction et une touche de validation. Optionnellement les touches numériques pourraient être utilisées.

Cette simplification de manipulation pour l'utilisateur se traduit par contre par un alourdissement de la charge de travail par le processeur qui doit considérer la position du curseur à l'écran, en plus du traitement du code de la touche de validation . Ce paramètre de positionnement du curseur intervient dans le test d'accessibilité à une opération.

Ainsi l'interpréteur n'a à reconnaître que cinq codes (au maximum 15 codes) correspondant à deux ensembles très distincts de traitements par le système:

-les codes opération de déplacement du curseur au nombre de quatre (déplacement vers le haut ou le bas, à droite ou à gauche).

-le code validation d'une option, ou d'une commande.

Un code de déplacement n'implique que l'exécution d'une routine de déplacement du curseur à l'écran (voir fig n°VII.). C'est un traitement simple et court effectué sous interruption. Par contre l'utilisation de la touche de validation valide une action représentée par un symbole occupant une aire géographique précise (et évolutive). Ainsi le test du code s'accompagne de test de localisation du curseur.

Le nombre et la position des icônes, fenêtres, et commandes étant évolutif, ces objets sont représentés dans un tableau de correspondance (à deux entrées) icône, processus, fenêtre (TCIPF). Ce tableau renferme pour chaque objet disposant d'une représentation graphique, toutes les informations nécessaires à sa manipulation.

pour un icône:

-n°d'icône
-n°de processus
-n°d'enregistrement
-état d'activabilité
-coordonnées de position
-n° de fenêtre

pour une commande:

-n°de commande
-coordonnées de position
-état d'activabilité
-adresse de la fonction de traitement de la commande

Cette structure de représentation des informations utiles permet d'être indépendante de la position des différents objets à l'écran. De plus toutes les informations sont regroupées en une seule structure de taille évolutive en fonction du nombre d'objets.

A la suite de la réception d'un code de télécommande, l'interpréteur vient tester la validité du code reçu et le type d'action à réaliser, effectue un déplacement ou passe par une série de tests définis pour enfin scruter le tableau de correspondance Icône-Processus-Fenètre (TCIPF). Ce tableau constitue le lien dynamique entre trois gestionnaires; le gestionnaire d'écran , le gestionnaire de processus, l'interpréteur de commande.

Ce test est important car il implique deux types de traitements d'importances inégales tant sur le plan action à réaliser, que sur le plan des ensembles logiciels concernés et les interactions engendrées (fig n°:VI.2).

Ainsi pour un code correspondant au déplacement du curseur, les seules opérations à réaliser avec les fonctions graphiques consistent à déplacer le curseur selon la direction choisie. Aucun appel de fonction du noyau n'est effectuée. Après déplacement du curseur, le processus en cours reprend son exécution un instant suspendu.

Un code de validation implique l'utilisation du TCIPF pour effectuer les tests de positionnement, et les tests d'activation des différents objets. Le résultat positif à ces tests conduit à un déroutement vers le noyau de l'exécutif pour charger les numéros de processus et d'enregistrement en file d'attente.

L'inexécution d'une action a trois raisons; soit le code reçu ne correspond pas à l'un des codes actifs prédéfinis, soit l'action demandée est indisponible temporairement et c'est un indicateur d'indisponibilité qui en marque l'état. L'indisponibilité étant dûe à une limitation imposée (quatre fenêtres ouvertes au maximum, ou déplacement en dehors du cadre de l'écran), ou l'incompatibilité entre deux fonctions (lecture et écriture d'un enregistrement), soit l'action demandée et l'environnement de service (ouverture et consultation d'un fichier message en dehors du processus de traitement de la fonction de messagerie) présente une incohérence manifeste.

La structure graphique

Définir une structure graphique, c'est définir des zones de travail spécialisées, les zones de validation (boutons, curseur, menus déroulants), les fenêtres et leur zones actives.

Les différentes zones de l'écran

L'écran est divisé en trois zones assurant le principe d'homogénéité de l'interface; une zone de travail, une zone de commande (barre horizontale en haut de l'écran), une zones d'affichage des icônes (fig n°:VII.1).

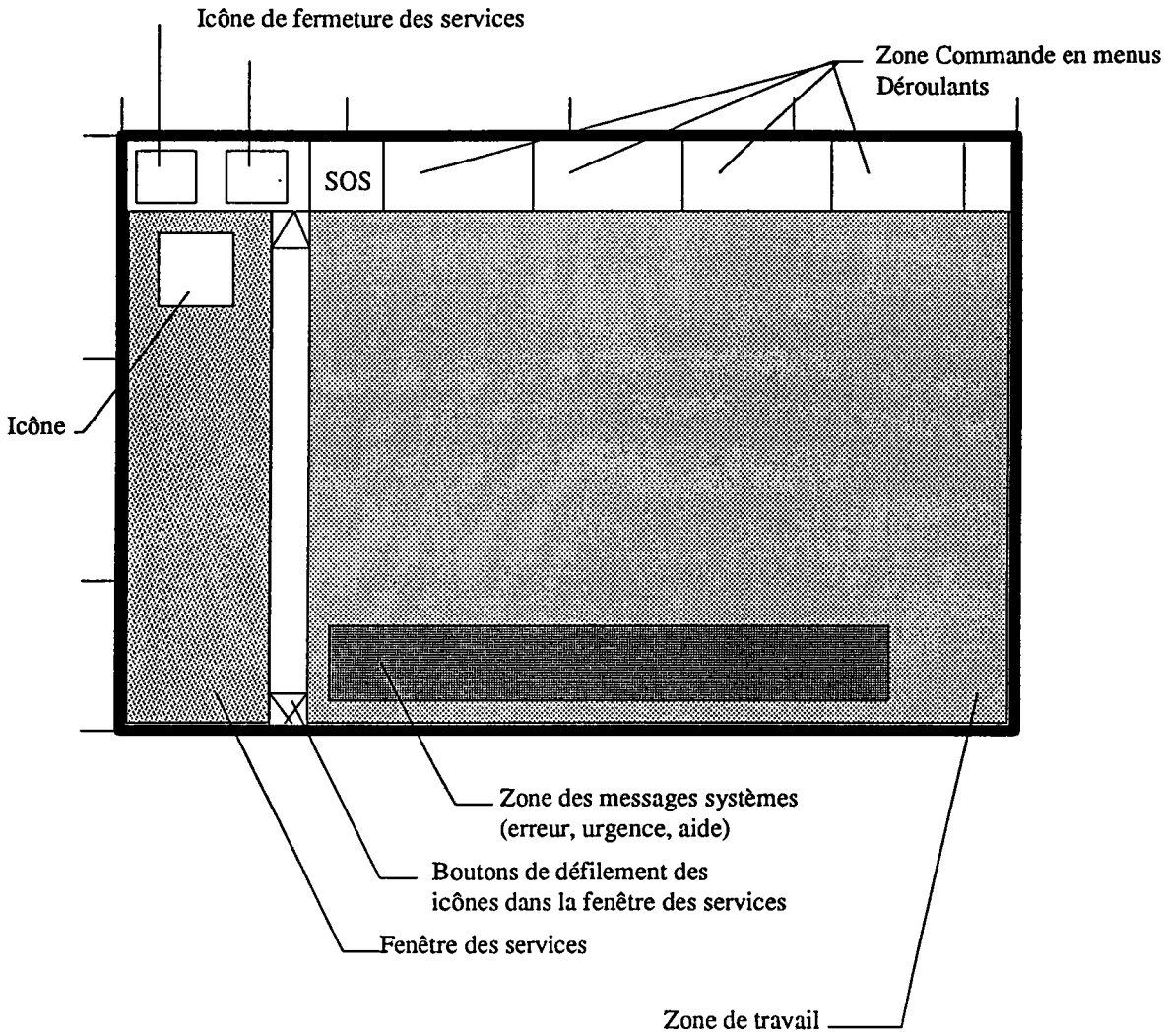


Fig n°: VII.1.: Structure de présentation graphique des services de l'interface visuelle

Bibliothèque d'icônes

La taille limité de la RAM privée du VDP ne permet de générer et stoker en mémoire qu'une centaine de caractères définis par le concepteur du logiciel, ce qui limite le nombre d'icônes disponibles selon qu'ils sont constitués de deux ou quatre caractères.

Chaque caractère est définie point par point à partir d'une matrice (10*8), codée sur dix octets, auxquels s'ajoute le numéro du caractère UDS (caractère redéfini par l'utilisateur).

Mécanisme de gestion des icônes

Le processeur vidéo EF 9345, renferme six jeux de caractères prédéfinis (mode texte) et un générateur de caractères. La matrice d'icône pour être utilisable doit être chargée, au préalable en mémoire privée, caractère par caractère.

Il est possible d'utiliser de façon consécutive différents jeux de caractères et notamment les jeux de caractère utilisateurs. A chaque commutation de caractère il est nécessaire de reconfigurer certains registres du processeur vidéo pour installer le jeux de caractère souhaité.

VII.2: Processus utilisateurs, applications

Les applications développées essaient de mettre en évidence les possibilités d'utilisation du terminal géré par un exécutif multitâche. Ces applications sont au nombre de quatre et constituent quatre types de service mis en oeuvre soit de par les ressources seules de l'équipement, soit en relation avec des appareillages avec lesquels le terminal peut échanger des informations quel que soit l'initiateur de l'échange.

La messagerie (fig n°VII.2)

Ce service d'intérêt général est incontournable et met en valeur l'ensemble des échanges nécessaires à une liaison concernant deux terminaux régis par un protocole défini. Le terminal utilise dans le cas de Médiabus, l'opportunité offerte par le gérant du réseau qui l'interroge pour signaler son désir d'émettre.
Ce service est bidirectionnel puisqu'il inclue l'émission et la réception des messages.

Au plan interne et logiciel du terminal ce service ne fait intervenir que la gestion du protocole de communication et la gestion de la mémoire centrale pour le stockage des informations reçues.

Ce service vu par l'utilisateur se présente par un icône dont la validation provoque l'ouverture d'une fenêtre comportant un certain nombre de paramètres à définir tel l'envoi ,et la consultation de messages. L'envoie d'un message suppose la génération d'un texte. Pour la démonstration on utilise un texte type du fait de l'impossibilité d'écrire un texte à partir des ressources existantes. Différentes options sont disponibles; la nature publique ou personnelle du message, et l'envoie immédiat ou différé du message (avec programmation de l'heure d'envoie).

La réception des messages n'implique que la consultation et éventuellement l'effacement des messages reçus.

Le programmateur (fig n°VII.4)

Ce service a été choisi car il met en oeuvre principalement les ressources du terminal et notamment le calendrier/horloge/alarme. Ce service utilise une file des alarmes classée

selon l'ordre horaires croissant. Le classement est opéré lors de la validation de l'alarme, par une fonction qui par tri, insère la nouvelle alarme dans la liste (fig n°VII.5). Chaque alarme est donc caractérisée par six valeurs correspondant aux secondes,minutes, heures, jour, mois année de l'alarme et un indicateur de nature de celle-ci . Cette alarme peut aussi bien correspondre à une alarme simple intégrée dans une fonction de mémo ou de pense-bête, mais aussi dans un mécanisme plus complexe de programmation tel que la mise en route et l'arrêt automatique d'un appareillage (four) ou même la programmation et la régulation temporelle du chauffage, et la simulation de présence.

L'horloge (fig n°VII.6)

Ce n'est pas à proprement parler un service incontournable, mais son avantage réside dans l'utilisation de l'aspect multitâche (qu'il prétend démontrer).

L'utilisateur désire ou ne désire pas connaître en permanence l'heure courante. Dans l'affirmative il sélectionne l'icône horloge et lance l'exécution du processus de lecture et d'affichage unique de l'heure disponible au niveau des registres internes du calendrier/horloge.

Ce processus est répétitif, et après son exécution reste programmé en exécution car il reste présent en file d'attente tant que l'utilisateur n'a pas dévalidé l'icône.

La périodicité du rafraîchissement de l'heure est fonction du nombre de processus en exécution et constitue un indicateur d'occupation du système..

VIII PROPOSITIONS, CONCLUSIONS

VIII.1: L'intendance de développement du logiciel

Les passage de paramètres

L'ensemble du logiciel est développé en deux langages, ayant chacun leurs spécificités et notamment des modes de passage de paramètres privilégiés.

->pour passer un ou plusieurs paramètres entre fonctions, en langage d'assemblage on procède:-soit par les registres internes: le nombre de paramètres est limité, et les transferts ainsi que les manipulations de données sont rapides.

-soit par variables globales: on utilise un intermédiaire en mémoire, ce qui augmente le temps de transmission mais offre une sécurité et une généralité plus importante.

-soit par la pile: en utilisant des instructions spécialisées, pour interfacer correctement les fonctions de haut niveau et des fonctions assembleur.

->le langage C effectue ses passage de paramètres;

-soit par variables globales: procédé réputé dangereux et au minimum contraignant.

-soit par la pile de façon systématique dans ce langage.

L'un des principaux problèmes rencontrés en exécution des programmes en langage C est le débordement de la pile, destructeur de données. Pour éviter cela il faut dans la mesure du possible ne pas dépasser un niveau d'imbrication des fonctions trop élevé, et dans chaque fonction ne pas multiplier le nombre de variables locales. Ceci impose de nouvelles contraintes au développeur d'application. A chaque appel de fonction, au minimum, la pile consomme quatre octets pour sauvegarder l'adresse de retour à la fonction appelante. A ceci s'ajoute la place réservée pour la transmission d'éventuels paramètres entre fonctions (chaque paramètre consomme quatre octets), et surtout les variables locales (de tailles variables) placées dans une pile secondaire (crée par l'instruction LINK) à l'intérieur de la pile principale du processus.

Dans le logiciel a été implanté un mécanisme de surveillance de l'évolution de la pile en cours d'utilisation. A partir d'un certain pourcentage d'occupation un nouvel espace mémoire est alloué automatiquement à la pile par exécution du processus d'extension d'une zone mémoire.

Utilisation adaptée des différents modes de passage de paramètres

Le passage de pramètres par la pile (langage C) est pratique car il offre la possibilité soit de travailler sur une ou plusieurs données et les modifier, soit de travailler sur une copie de la variable, qui reste inchangée après l'exécution de la fonction modifiante.

Pour un grand nombre d'opérations réalisables par l'exécutif j'ai voulu économiser l'utilisation de la pile par passage de paramètres et je me suis reporté sur l'utilisation privilégiée de variables globales d'implantation fixe. L'exécutif fonctionne sur l'utilisation de variables globales implantées dès l'installation du logiciel en mémoire, avec mémorisation permanente des informations fondamentales du système. La zone mémoire contenant ces variables globales système n'est pas la pile système mais une zone située à la base de la mémoire physique, et non allouable à un processus (fig n°V.12).

Implantation des variables globales en mémoires en phase de conception du logiciel

Une différence entre le langage C et le langage d'assemblage, et pouvant justifier l'emploi de ce dernier réside dans la possibilité de positionnement, distinction que le développeur peut opérer sur les différents segments de données, de code programme, et de pile et la manipulation trés fine des registres internes du microprocesseur. Cette souplesse d'implantation n'existe pas en langage de haut niveau.

L'implantation des variables globales du système s'opère en deux temps. En phase d'assemblage on implante en un endroit définie une table contenant les valeurs de l'ensemble des variables globales initialisées. Ces valeurs ne sont pas référencées symboliquement. C'est ultérieurement que sont créés les liens (phase de linkage) entre les valeurs constantes (contenus des variables ou tableaux) et les adresses symboliques. En phase de linkage la table des valeurs est implantée en ROM par une commande adaptée, alors que les adresses des variables pointent sur un espace mémoire RAM. En phase d'installation du système, une fonction spécialisée vient lire en ROM les valeurs de la table et les copie en RAM, aux adresses des symboles correspondants. Cette opération initialise les variables globales et les rend enfin utilisables.

Lors de la création des variables globales, les références et les valeurs de celles-ci doivent êtres placées et déclarées dans le même ordre en respectant la taille de ces différentes variables.

Cette procédure d'implantation permet de conserver en mémoire permanente une copie des variables initialisées, et utilisées pour le système. La fonction de chargement n'est utilisée que dans la phase d'initialisation du système ou en cas de détection d'un dysfonctionnement grave du système nécessitant sa réinstallation complète.
Ce mécanisme délicat permet d'inscrire dans le silicium les données de base du système puis par passage en RAM permet leur utilisation, modification en cours d'exécution des programmes, tout en préservant des liens d'adressage préétablis.

Utilisation de différents langages de programmation

Le logiciel a été développé en deux langages (langage C et langage d'assemblage) en réponse à certaines contraintes et notamment des contraintes de temps du fait de l'inexistence en matière scientifique d'un outil ou d'une solution universelle pour résoudre tous les problèmes du domaine.

Le langage C est un langage de haut niveau offrant une somme non négligeable d'avantages pour un développeur d'applications, par ses propriétés de portabilité (entre différents processeurs et éventuellement différentes familles de processeurs) et de standardisation (sous certaines conditions), de relogeabilité (par les adressages utilisés) et enfin de modularité. Ainsi l'interpréteur de commande, les programmes d'application, et les fonctions d'interfaçage généralistes sont développées en langage de haut niveau.

La portabilité d'un programme développé en langage de haut niveau n'est pas acquise pour toutes les machines et trouve sa limitation dans les fonctions spécifiques constituant les bibliothèques. En effet la norme ANSI couvre la strandardisation pour la structure générale et les instructions de base du langage C. Pour assurer la portabilité des applications certaines fonctions complexes et trop spécifiques ne peuvent être utilisées.

La relogeabilité est généralement assurée pour les langages de haut niveau par utilisation des modes d'adressage relatifs au PC, ou opérant par déplacement par rapport à la pile. Cette propriété originelle permet ainsi de placer les codes programmes générés à n'importe quel endroit de la mémoire aprés recalcul des adresses symboliques par le système d'exploitation.

La nature et le degré d'optimisation du code généré distingue les compilateurs entre eux. Elle porte sur le choix d'un code instruction par le compilateur pour minimiser le nombre d'instructions et le temps d'exécution de certaines opérations (transferts rapides, transferts multiples réalisés en une instruction). L'option d'optimisation peut être dévalidée, pour produire un code très portable. L'optimisation fait choisir par le compilateur des instructions spécifiques à un processeur (non utilisable par un autre processeur de la même famille).

Le langage d'assemblage par contre est difficile d'accès (rebutant) et très spécifique d'une famille de microprocesseurs. Cette spécificité donne au développeur l'accès et l'utilisation de toutes les instructions du microprocesseur (ce que ne peut faire le langage C qui est un langage plus généraliste) telles que les exceptions logicielles et les instructions privilégiées. Les éléments de code programme développés en langage d'assemblage l'ont été pour la rapidité d'exécution de ce code (en routine de traitement d'interruption constituant les processus externes) et pour les possibilités de manipulation qu'il offre (manipulation des registres et notamment de la pile lors des déroutement). L'aspect compacité du code assembleur généré par la chaîne de développement logiciel n'est plus un argument décisif pour son emploi. Un code optimisé, généré en langage C présente dans les bons cas un rapport d'extension de 1,3. L'utilisation de l'assembleur se justifie pour des programmes courts et pour réaliser très rapidement des opérations délicates. Ainsi pour des systèmes d'exploitation se rapprochant de la notion de temps réel une part de code développée (5 à 10%) l'est en langage d'assemblage (15 à 20 % pour cet exécutif). C'est naturellement le noyau du système qui renferme les segments de code assemblé car il est la partie qui utilise le plus les ressources du processeur. D'autres éléments logiciels tels que les pilotes de périphériques, pour des fonctions de base utilisent l'assembleur.

La relogeabilité et la translatabilité sont des propriétés accessibles en langage d'assemblage qui permet de développer aussi des codes absolus.

Le langage d'assemblage, contrairement au langages de haut niveau ne fournit pas de bibliothèques de fonctions intégrables, mais met à disposition l'ensemble des instructions microprogrammées du processeur et des directives de positionnement de mise en forme des codes et des données (directives inexistantes en langage C).

Pour leurs caractéristiques propres et en fonction du matériel et des contraintes, l'utilisation adaptée des deux langages se justifient.

Développement en langage C et utilisation de fonctions prédéfinies

Au cours du développement en langage de haut niveau, les fonctions des bibliothèques autres que les fonctions mathématique n'ont pas été utilisées. C'est particulièrement le cas des fonctions de gestion de la mémoire. Ces fonctions C sont utilisées dans les programmes d'application intervenant en complément des fonctions du système d'exploitation de gestion des espaces mémoire (allocation, désallocation extention, réduction d'une aire mémoire). Elles conviennent mal pour définir et structurer le mécanisme de gestion de la mémoire, puisqu'elles le complète. Il m'a fallu donc développer des fonctions spécifiques (en langage C ou assembleur) pour gérer la mémoire.

VIII.2: Les caractéristiques temporelles du logiciel

Les performances du logiciel vont se révéler lors de son exécution en conditions réelles sur site, ce qui malheureusement n'a pas encore été fait. L'élément logiciel ayant fait l'objet de tests a été l'exécutif avec un interpréteur de commande réduit, quatre processus en exécution pseudo-parallèle et l'ensemble des interfaces opérationnelles. Définir les performances du logiciel, nécessite de mesurer d'une part les temps de réaction à certains évènements et les temps de traitement de ceux-ci, et d'autre part la qualité, la souplesse de maniement des services proposés ce qui ne peut s'appréhender que sur un plan purement subjectif.

VIII.3: Evolutions matérielles et logicielles du terminal domotique

Les évolutions de ce terminal peuvent être envisagées aussi bien sur le plan matériel que logiciel mais cette évolution reste subordonnée au double aspect ergonomie et fonctionnalité qui feront de celui-ci un équipement grand public.

Quelles que soient les prétentions du concepteur de logiciels pour faire de cet équipement une réalisation ergonomique, rapide et souple de mise en oeuvre et d'utilisation (l'idéal en somme!), le matériel avec son architecture, les composants et leurs temps de réponse, définissent les limites de possibilité de fonctionnement.

Ainsi lors de la phase d'élaboration du cahier des charges matériel il est nécessaire de définir les fonctionnalités de base de l'équipement et des éléments le constituant en veillant à en faire un système modulable, évolutif à court terme pour les concepteurs (modifications, remplacements et ajouts de composants) et à moyen terme pour l'utilisateur (élargissement des possibilités et des performances).

Les axes d'évolution

Par rapport à la structure matérielle actuelle de l'équipement, un certain nombre de modifications matérielles ou de mises-à-niveau logicielles sont applicables à un nouveau prototype. Ces quelques suggestions concernent trois domaines:

-la sécurité de fonctionnement du matériel et des données qu'il renferme.

-l'élargissement des capacités mémoire.

-les performances et la structure de conception de l'équipement.

-les nouveaux périphériques.

Il est à préciser qu'un problème soit matériel soit logiciel sera le plus souvent résolu par combinaison de ces deux types de solutions; les deux étant indissociables.

La sécurité

La sécurité s'envisage aussi bien au plan du fonctionnement du matériel qu'au plan de la protection des données contenues dans l'équipement.

Protection des données:

Il est nécessaire pour le bon fonctionnement de tout système multitâche ou multi-utilisateur de séparer en deux espaces les codes et données du système de celles des

programmes d'application. Cette distinction de placement correspond aux deux modes de fonctionnement du microprocesseur.
Cette mise à niveau est indispensable. Les espaces RAM superviseur et utilisateur peuvent être de tailles différentes (16 Ko de RAM superviseur et 64 Ko de RAM utilisateur). La RAM superviseur est réservée pour contenir les filles d'attente, tableaux descripteurs, tableaux de correspondance icône-processus, tables de présence de périphériques, et enfin les variables globales et la pile du système.

En cas de coupure d'alimentation secteur la sauvegarde du contenu des RAM superviseur et utilisateur par une batterie permettrait de garder le terminal opérationnel en conservant toutes les données (et programmes d'applications importées) ainsi que les paramètres de configuration.

Sécurité de fonctionnement matériel.

Durant la phase d'initialisation des interfaces du terminal, il est nécessaire de stopper le battement naturel du calendrier/horloge/alarme (période d'une seconde) pour installer le signal d'horloge système qui régle le fonctionnement de l'exécutif. Pour éviter tout conflit si l'arrêt du battement n'est pas effectif durant cette phase délicate, on peut effectuer une connexion supplémentaire entre la ligne d'interruption I2C (niveau 5) et l'entrée C3 du temporisateur #3 (du PTM 6850 générateur de l'horloge système) programmé en détection d'un signal d'interruption I2C. La phase d'initialisation des interfaces touchant à sa fin, le registre d'état du PTM sera testé pour contrôler l'arrêt du battement. Alors sera effectuée la programmation du temporisateur #3 du PTM en horloge. En cas de poursuite du battement, un certain nombre d'essais de reprogrammation du PCD 8583 auront lieu avant génération d'un message de dysfonctionnement à destination de l'utilisateur.

Le logiciel de gestion du terminal devra comporter un ensemble de fonctions de tests des différents éléments matériels. Ces tests seront regroupés sous forme d'un processus d'exécution périodique. Ces tests de bon fonctionnement portent:
-sur l'interface série RS232C.
-sur les différents composants reliés au bus I2C.
-sur les différents espaces mémoire pour l'évaluation de leur taille et leur niveau d'occupation.
-sur l'horloge système.
et devront conduire à une correction adaptée des défauts constatés.

Pour résoudre un problème ponctuel de dysfonctionnement d'un périphérique (ne pouvant être résolu par le logiciel) il est nécessaire d'opérer uniquement sur celui-ci, en utilisant un des systèmes de réinitilisation matériel différencié pour chaque périphérique.

Pour résoudre les problèmes logiciels de l'exécutif (destruction totale ou partielle d'une table) il est possible de concevoir un mécanisme de réinitilisation logiciel adaptatif en fonction du niveau de gravité du dysfonctionnement. L'aspect adaptatif intervenant pour éviter de perdre le minimum de données applicatives.

Extension descapacités mémoire

Pour assurer le stockage d'un nombre important de données provenant du réseau, et du terminal lui-même il serait nécessaire de porter l'espace mémoire de 32 Ko (actuellement) à 256 Ko voir même 512 Ko.

Pour alléger le travail du microprocesseur, sur le plan de la gestion mémoire il serait intéressant d'implanter un composant contrôleur de gestion mémoire MMU (Memory Management Unit).

L'utilisation d'un support de masse offre (disquette de 720 Ko, au format 3p 1/4) la possibilité de décharger la mémoire embarquée du terminal, et une gestion de celle-ci selon un standart reconnu (format DOS reconnu par Atari et Macintosh ProDos) permet un portage des applications. Cet extension nécessiterait l'apport de modules logiciels pilotant ce périphérique et quelques modifications du gestionnaire de mémoire apportant une gestion de mémoire virtuelle.

Amélioration des performances par modification de la structure matérielle

C'est au niveau des performances et de l'intégration que s'effectue de façon décisive le choix entre microprocesseur et microcontrôleur à l'intérieur d'un même famille et que se dessine le choix de certaines solutions techniques.

Pour un version matérielle très intégrée (en deux cartes au format simple Europe pour le prototype de base), on utilisera un microcontrôleur 68070 (RTC Phillips Composants) intégrant dans un même boîtier (PLCC):

-un microprocesseur 68000.
-un interface série RS232C (à dédiée au réseau collectif domotique).
-une interface I2C (le reliant directement au récepteur de télécommande, au compteur d'évènements, au calendrier/horloge/alarme).
-un triple temporisateur programmable utilisable en horloge système.
-deux interfaces DMA (non utilisées).

Cette solution regroupe en une seule carte l'unité centrale, sa mémoire, et les interfaces de base du terminal et permet ainsi de supprimer deux composants d'interfaces indispensables.

Dans cette configuration le dispositif d'entrée tamponnée par le port série avec dispositif matériel de comptage, de reconnaissance d'adresse se révèle non applicable.

Choisir un microcontrôleur plutôt qu'un microprocesseur permet de réduire le nombre de composants et d'améliorer la fiabilité (des contacts). Pour échanger systématiquement des données avec les interfaces on passe par un réseau série plutôt que par un bus parallèle afin de réduire le câblage.

Employer un microprocesseur 16 bits tel le 68000 par rapport au prototype actuel (conçu autour d'un µp 8 bits; le 68008), se traduit par un élargissement des possibilités matérielles au niveau des interfaces (PTM 68230 ...) et aussi par une augmentation des performances d'exécution. Rappelons que pour augmenter les performances, on a recours à différentes solutions:
-augmenter la fréquence de fonctionnement du processeur. Il faut alors que les autres composants suivent (coût plus élevé).
-élargir la taille du bus d'adresse. C'est la capacité d'adresse mémoire qui est augmentée (1 Mo*4 pour un 68008 avec un bus d'adresse de vingt bits, 16Mo*4 pour un 68000 sur un bus d'adresse de 24 bits).

En utilisant un microprocesseur (composant généraliste) tel que le 68000, le prototype sera composé de trois cartes électroniques:

-une carte unité centrale portant le microprocesseur, ses quatre espaces mémoire extensibles par superposition ("piggy-back"), et l'horloge système.
-une carte de communication portant les interfaces RS232C (réseau domotique collectif) et I2C.
-une carte de visualisation graphique.

Pour conserver au système sa capacité à réagir de façon quasi-immédiate à un événement extérieur et à y répondre rapidement il est nécessaire que les temps de traitement des routines d'interruption soient minimum (processsus internes), et pour cela il faut éviter de connecter sur une même entrée un nombre trop important de sources d'interruption introduisant des test de reconnaissance programmés de celle-ci ("daisy-chaîn"). Deux ou trois sources par entrée semblent être le nombre maximal acceptable. Pour un composant d'interface interruptible particuliérement lent, une entrée sera monopolisée par celle-ci.

De façon spécifique pour alléger le travail du microprocesseur en réponse aux interruptions générées par le port série (période: 520 µs) peut être conçu un système de réception bufférisé des données avec mécanisme de génération d'interruptions différenciées

lorsqu'un taux de remplissage prédéfinie du tampon de réception est atteint. Le microprocesseur n'est alors interrompu que toutes les 3ms (au maximum) pour une vitesse de transmission de 19200 bauds . On peut y adjoindre une fonction de reconnaissance de l'adresse du terminal et de fin de message; fonctions réalisées par le matériel et ne faisant intervenir le processeur que pour traiter l'information dans son intégralité (protocole de communication). Le choix de traitement des informations provenant du port série influera sur le matériel (choix du microprocesseur et des interfaces) et sur le logiciel de gestion (conception).

La structure d'utilisation d'un double bus définissant deux ensembles matériels est une bonne approche, et devrait être complétée par utilisation d'un type d'interfaces "intelligentes" effectuant quelques traitements simples avant appel au processeur. Par sa capacité de traitement, le processeur devant être réservé aux opérations complexes.

Le terminal dans l'état actuel pose un double problème sur le plan électromagnétique, en étant sensible à ce type de perturbations et parce que lui même constitue une source de rayonnements. Les solutions à apporter portent sur sa conception (plan de masse des cartes et de l'ensemble, positionnement des composants...).

Des dysfonctionnement dûs à une élévation de la température ont été observés suggérant l'installation d'une ventilation forcée.

Les nouveaux périphériques

Le processeur de gestion d'écran (EF9345) utilisé présente pour avantage d'être un composant éprouvé (minitel) offrant des possibilités de choix d'affichage multiple en résolution, palettte de couleur, effets visuels, mais son affichage semi-graphique se prête difficilement aux manipulations graphiques actuelles (fenêtres multiples à zones actives incluses). Sa lenteur est un autre point négatif (bus adresses/données multiplexé).

Le contrôleur vidéo SCC66470 est un composant moderne s'interfaçant avec les microprocesseur de la famille 68xxx, offrant une gestion graphique de haute résolution (768*560) et capable de gérer un mémoire vidéo dynamique privée de 1.5 Mo.

Dans le but de multiplier les interfaces entre l'homme et la machine permettant ainsi élargir les possibilités offertes par ce terminal et donc le public intéressé (malvoyant), il serait possible de doter l'équipement d'une carte configurable par l'utilisateur de reconnaissance de la parole d'une part et de synthèse vocale d'autre part.

En application de la monétique le terminal pourrait être dôté d'un lecteur de carte magnétique permettant par ce terminal d'accéder aux services de télé-achat, ou à d'autres applications utilisant des cartes à microprocesseur (application médicale).

Selon les applications souhaitées il sera possible par une ou plusieures cartes de disposer d'interfaces de communication (une carte Médiabus, une carte Batibus par exemple) avec des automates, capteurs et actionneurs (domotique) ou avec différents serveurs (immotique).

Pour assurer la pleine évolutivité du matériel et du logiciel (sans refonte du logiciel embarqué) par rapport à l'installation de nouveaux périphériques ,il serait souhaitable de créer une table de référence de tous les périphériques présents. Les pilotes de périphériques pouvant être importés par le réseau Médiabus (téléchargés).

Un langage de programmation évolué

Les propositions d'évolutions logicielles et matérielles ont pour but d'améliorer le fonctionnement du terminal, par une réduction des temps de réponse, par une fiabilité et une intégration accrue.

A un autre niveau, il est concevable de modifier la couche logicielle d'interprétation et de gestion d'écran, du fait de la modularité et l'indépendance des trois ensembles de programmes. Cette modification consiste à développer encore l'aspect ergonomique en ajoutant un module langage de programmation graphique interactif. Essayons de définir plus précisément cette idée.

A partir des services existants et représentés par des icônes, l'utilisateur par sa télécommande, sous forme d'une ligne reliant deux icônes, initialiserait un lien de programmation dynamique et d'interaction entre ceux-ci. Ces liens étant prédéfinis aussi bien au plan des paramètres échangés, qu'au plan des services pouvant interagir. Ces deux services seraient exécutés conjointement, en interaction, permettant ainsi l'élargissement des possibilités du terminal. L'utilisateur aurait ainsi la possibilité de programmer ses applications sur mesure sans connaître un langage de programmation impliquant un ensemble de commandes et une syntaxe à respecter.

VIII.4: Applications possibles

Le but du logiciel d'exploitation, que j'ai développé est d'utiliser au mieux les ressources matérielles du terminal en vue d'en faire un équipement mettant à la disposition de l'utilisateur *lambda*, des services domotiques et immotiques. Comme les services à destination de l'usager ne sont pas clairement définis, il a fallu constituer un prototype matériel et logiciel très ouvert.

Le logiciel est structuré en trois couches, définissant trois unités programmées indépendantes: un moniteur multitâche, un interpréteur de commande et une interface graphique regroupés dans la même couche, et un ensemble de programmes d'application.

La constitution d'un exécutif multitâche provient du besoin de modularité issue de ces incertitudes liées au rôles d'unité de traitement et de visualisation des données provenant d'un nombre élevé et évolutif des sources d'information.

Un vaste champ d'applications

Le champ d'application de cet équipement est élevé, du domaine de la sécurité, au domaine de la communication interactive publique, en passant par ceux de la santé, de l'éducation, de l'automatisation du fonctionnement d'équipements ménagers.

Ce terminal est en fonctionnement permanent, tant pour l'écoute du réseau Médiabus, que pour toute sollicitation par la télécommande. Pour être complet, quelques équipements complémentaires devraient lui êtres adjoints (buzzer, commande d'allumage et extinction automatique de l'écran TV) pour lui permettre d'attirer l'attention de l'usager à la suite d'évènements prioritaires.

Ainsi sur le plan de la santé cet équipement permettrait d'une part de surveiller à son domicile et de façon permanente l'état d'un malade ou d'une personne âgée, le nécessitant. Une copie du dossier médical pourrait être consultée (avec une clé d'accès personnalisée) et enfin il serait possible au patient de noter d'éventuels problèmes de santé. Utiliser la carte pour contenir l'ordonnance prescrite par le médecin (domaine de l'EDI) et la diffusion des informations médicales à caractère préventif pour le grand public sont encore deux propositions d'application.

Au plan de la sécurité, trois axes de développement de services peuvent êtres envisagés: système de prévention et de détection d'intrusion, détection et signalisation d'incidents ménagers (fuites de gaz, inondation, court-circuit, incendie). Ce type d'applications sécuritaires fait du terminal un point de convergence de différents capteurs, faisant du boîtier un outils de contrôle et commande automatique.

Toutes ces applications nécessitent d'instaurer un échange d'informations permanent et systématique entre le terminal et différents serveurs (dont le rôle ne se limite pas

seulement à être une bases de données). Le domaine de la communication sur le plan des réponses apportées devra évoluer en s'ouvrant aux nouvelles techniques offertes par l'informatique, entre-autre (logiciels auto-adaptatifs).

Les éléments décrits dans différents ouvrages

Des nombreux ouvrages que j'ai consulté sur les systèmes d'exploitation, pour réaliser le logiciel traitent essentiellement des problèmes d'ordonnancement dont les mécanismes sont bien circonscrits. En revanche bien peu de livres étudient (et de façon superficielle) les mécanismes de gestion de la mémoire centrale, les interactions entre les différentes couches logicielles, et d'autres domaines similaires.
Seuls trois livres abordent de façon plus approfondie et complémentaire, l'organisation et les mécanismes de gestion de la mémoire centrale (non étendue), mais cette description (mémoire segmentée, paginée) est très liée à l'utilisation privilégiée d'une famille de microprocesseurs (Intel). J'ai donc développé une représentation et une méthode de gestion adaptée au prototype, décrite dans quelques ouvrages de façon superficielle (je l'ai découverte bien après avoir définie moi-même auparavant les règles de gestion nécessaires), et semble-t-il peu utilisée du fait de l'existence, actuellement, de grandes capacités mémoire primaires et secondaires et de méthodes paliatives aux besoins en mémoire ("swapping"). A un autre niveau, il n'existe pas non plus de description des interactions entre différentes couches logicielles. Le critère prédominant est la vitesse d'exécution des instructions par le microprocesseur au détriment de l'optimisation du fonctionnement des autres ressources matérielles.

Il en ressort que les systèmes d'exploitation gardent leur part d'opacité et de complexité seulement accessibles par des spécialistes. J'espère par ce mémoire apporter quelques éléments de compréhension supplémentaires.

La recherche dans ce domaine n'est pas une recherche phare. Elle se focalise dans des aspects de conception "sur-mesure", et dans la mise au point d'outils participant à la définitions des stratégies de gestion multiprogrammés en application industrielle ou prédomine la notion de temps réel. Il n'en reste pas moins que les systèmes d'exploitation doivent encore évoluer pour gérer les ressources à architecture et technologies différentes (transputer, SE orienté objet) et intégrer des fonctions supplémentaires (gestion réseaux, base de données).

Conclusions personnelles

Aucun service domotique ou même immotique n'est actuellement disponible pour le grand public. Le développement de cette activité qui est la somme de techniques ayant pour but de contrôler de façon décentralisée et interactive différents équipements domestiques est subordonnée à l'existence de véritables services utilisateurs. Ces services étant accessibles par un équipement banalisé; le terminal domotique présenté utilisant l'appareil de TV.

Dans un premier temps ce terminal (esclave) était destiné à dispenser des services fournis par un serveur et un réseau câblé de télédistribution adapté à cette tâche de transmission d'informations alphanumériques, dans un cadre limité (dans un hôtel par exemple). Cette premiére réalisation logicielle et matérielle a permis d'évaluer la pertinence des choix opérés sur les composants. La connexion au réseau de télédistribution (infrastructure matérielle et protocole de communication ayant été conçu au CERLOR) n'ayant pas eu lieu, il n'a pas été possible d'évaluer les possibilités du prototype en environnement réel.

Par la suite , la nécessité de rendre évolutif le logiciel s'est avérée nécessaire. La structuration du logiciel permettait de redéfinir le but, le rôle et les possibilités du matériel. Il m'a semblé que cette vision d'un terminal esclave défini par rapport à un réseau spécifique était étriquée et du point de vue du concepteur ne permettait pas d'utiliser pleinement le potentiel du matériel. Pour l'utilisateur, un autre rôle plus ambitieux pouvait lui être assigné et notamment celui de noeud de communication entre deux réseaux (deux bus) entre deux mondes, celui de l'immotique et celui de la domotique. Cette double appartenance est, la seule susceptible d'imposer ce terminal comme outil de communication et comme outil de contrôle d'évènements purement domotiques, ou/et immotiques. C'est en intervenant dans les deux espaces que le terminal joue pleinement son rôle, et peut présenter un intérêt sur le plan économique et utilitaire (services complémentaires). Une passerelle est alors concevable entre la domotique et l'immotique, impliquant différents automatismes de fonctionnement intéressant par exemple la sécurités des biens et des personnes.

C'est dans cet esprit que j'ai conçu la nouvelle structure logicielle en trois couches (interpréteur de commandes et interface graphique, programmes ou processus utilisateur, moniteur de gestion des ressources matérielles.

Le nombre élevé et la diversité des sources d'information prévisibles survenant aléatoirement ainsi que les limitations techniques (nombre d'entrées d'interruption hiérarchisées) ont été les principaux paramètres qui m'ont incité à opter pour un moniteur multitâche implanté en ROM (famille des systèmes d'exploitation à vocation industrielle). Les mécanismes de gestions particuliers de certaines ressources telle que la mémoire m'ont fait choisir la solution de développer un exécutif multitâche spécialisé pour ce terminal en premier lieu puis par la suite les éléments logiciels complémentaires pour proposer de façon

ergonomique des applications de services type (messagerie, programmateur, horloge, hôtellerie)

Ces trois entités logicielles sont conçues dans le soucis permanent d'assurer à long terme une modularité et une indépendance de celles-ci permettant de modifier d'une part les composants interface et donc les pilotes de périphérique correspondant et d'autre part étendre les possibilités de services offerts résidents (mise à jours de la ROM application) ou importés (en fonction des services proposés). Ces conditions imposent alors au niveau de la conception des programmes une bonnes lisibilité des programmes, une organisation des modules programmés en librairies, une implantation des modules en mémoire en fonctions de leur fonctionnalité et de leur évolutivité.

Une documentation multiforme accompagne chaque entité programmée, exposant pour chacun les mécanismes mis en jeu (gestion multiprogrammée, gestion mémoire, gestion graphique)et décrivant chaque fonction par ses caractéristiques, spécificités , et conditions d'utilisation, et enfin une description globale du logiciel et une notice d'utilisation. Ainsi, aussi bien le développeur d'applications, l'ingénieur informaticien que l'utilisateur pourrons trouver leur bonheur.

Bien que cet équipement puissent présenter certaines applications, il reste beaucoup à faire aussi bien sur le plan de la conception (méthodes de gestion différentes), écriture du logiciel (optimisation) permettant d'améliorer le résultat, de sécuriser son fonctionnement, à partir de l'existant, ou même d'évoluer pour apporter de nouveaux services.

Le développement d'un tel ensemble cohérent de programmes permet d'évaluer et de solutionner tous les problèmes de gestion de chacune des ressources matérielles et logicielles et de l'ensemble, compte tenu des contraintes de temps de traitement de certains évènements.

Le développement des réseaux câblés de télédistribution étant limité (200 000 abonnés pour la France) ce terminal en restant dépendant de cette configuration n'a qu'un avenir de démonstration. La possibilité de l'interfacer avec d'autres supports et moyens de communication ainsi que sa position intermédiaire entre les sphères domotiques et immotique pourrait utiliser le developpement de l'une d'entre elle pour assurer celui du terminal. Au niveau domotique, l'un des obstacles peut être, outre l'inexistence de services attractifs, le coût du terminal seul et des éventuelles extensions. L'évaluation du coût de cet équipement (1000 Frs) en dehors de toute réduction par économie d'échelle, permet de définir la valeur du logiciel embarqué. Le coût de l'équipement de base doit être raisonnable de l'ordre de 1000 à 2000 Frs pour en favoriser l'achat. Par contre les services devraient probablement être payants, facilité en cela par l'essor à moyen terme du télépaiement et de la monétique (l'appareil devrait être équipé d'un lecteur de carte à puce)..

ANNEXE I: Annexe bibliographique.

Cette liste recense l'ensemble des ouvrages ayant servis de base à mes travaux dans le cadre de la réalisation du logiciel d'exploitation de l'équipement de communication (E.C.A.D).

Les systèmes d'exploitation:

[KRA] KRAKOWIAK
Principe des systèmes d'exploitations des ordinateurs
Dunod Informatique, 1979.

[BEA] J BEAUQUIER, B RERARD
Systèmes d'exploitation; concepts et algoritmes
Mc GRAW-HILL, Janvier 1991

[LIS] A LISTER
Principes fondamentaux des systèmes d'exploitation
Eyrolles, Janvier 1987.

[TAN-1] A TANENBAUM
Les systèmes d'exploitation
Interédition, 1990.

[TAN-2] A TANENBAUM
Architecture de l'ordinateur
Interédition, Janvier 1987.

[POU] J-P POUJET
Ordonnancement en temps réel
Masson

[TSC] D TSCHIRHART
Commande en temps réel; conception et mise en oeuvre d'un exécutif multitâche
Dunod Informatique Industrielle, Janvier 1990.

[DOR] A DORSEUIL, P PILLOT
Le temps réel en milieu industriel; concepts, environnements, multitâche
Dunod, Janvier 1991.

[HAN] L HANNEBICQUE, P JAULENT
Le système OS9
Editest

Microprocesseurs:

[VIE] M-C VIELLEFONT
Le microprocesseur 68000
Sybex

[JAU-1] P JAULENT
Le microprocesseur 68000
Eyrolles, Avril 1987.

[GEO] B GEOFFRION
8086-8088, iAPX 186-188-286-80386
Programmation en langage assembleur
Edition Radio

[LIL] H LILEN
Le 80386; modes de fonctionnement (architecture, programmation, caractéristiques)
Edition Radio

Langages de programmation:

J-P MALENGE,S ALBERSTEN, P COLLARD, L ANDREANI
Programmation structurée en assembleur 68000
Masson, Juillet 1987.

Entrées-Sorties en C; PC-AT et compatibles
Masson

J YOUEDEOU
Techniques de programmation en C; les communications série
Sybex

Interfaces matérielles:

[LIL] H LILEN
Principes et application des interfaces pour micro-ordinateurs
-technique
-normes
-connexion des périphériques
Edition Radio, 1989.

Interfaces Hommes-Machines:

[BAR] M-F BARTHET
Logiciels interactifs et ergonomie; modèles et méthode de conception
Dunod Informatique

[COU] J COUTAZ
Interface Homme-Ordinateur; conception et réalisation
Dunod Informatique

[SIC] Y SICARD
Analyse et conception d'interface entre un système informatique et son utilisateur
Thèse de docteur ingénieur. 1983. Grenoble

Les réseaux de communication:

[PUJ] G PUJOLLE, D SERET, D DROMARD, E HORLAIT
Réseaux et télématique, Tome 1
Dunod, Mars 1989.

Méthodes de développement de logiciels

[CAL] J.P CALVEZ
Spécification et conception des systèmes, une méthodologie
Masson, Janvier 1990.

[AMG] A AMGHAR
Méthodes de développement d'un système à microprocesseurs
Masson, Févier 1987.

[BIG] T BIGGERSTAFF
Outils logiciels pour la programmation système
Masson, Février 1989.

Domotique et immotique

[NOZ] J NOZICK
La maison intelligente
Editions du moniteur, 1988.

[HUE] A HUET
Immeubles intelligents et téléports
Eyrolles, Juin 1990.

[COL] J COLOMES, P MERIEUX, J SCHOMOUKER
Domotique
Eyrolles, Janvier 1991.

Un certain nombre d'articles parues dans des revues spécialisées m'ont permis d'avoir une vue d'ensemble des systèmes d'exploitation, ou des exécutifs disponibles selon l'ordonateur hôte.

[VAN] J.J VANDEWALLE, P PARADINAS
Opérating Systèm pour Equipement Personnel Intelligent (EPI)
INRIA 1992, rapport de l'équipe architecture et systèmes (n°140).

[VER] J-P VERVAY
Systèmes d'exploitation temps réel; l'heure du choix
Mesures Juin 1990

[DUB] R DUBOIS, S LEPONT
Systèmes d'exploitation et temps réel
Minis et Micros n°297/Mars 1988

[DOR] A DORSEUIL
Exécutifs et temps réel
Minis et Micros n°327/18 1989

[JPV] Systèmes d'exploitation temps réel multitâche et multiuilisateur
Minis et Micros n°297/7 1988

[GRO] C GROSS
Des systèmes d'exploitation sur disque ou sur silicium qui gèrent le temps réel
Electronique industrielle n°104/15-03-1986.

ANNEXE II: Glossaire

J'ai regroupé ,ici les principaux termes anglais utilisés dans le domaines de l'informatique et notamment celui ,des systèmes d'exploitation.

time sharing:	temps partagé
time slice:	unité de temps d'exécution d'un processus
shell:	environnement de commande
current:	programme ou processus élu
ready:	programme ou processus éligible
waitting:	programme ou processus en attente
program counter:	compteur ordinal
processor status longword:	mot d'état du processeur
kernel:	noyau
load process context:	restauration du contexte processus
save process context:	sauvegarde du contexte processus
round robin:	principe du tourniquet
highest priority first:	principe de la plus haute priorité d'abord
feed-back queues:	principe de files d'attente rétroactive
cyclic round robin:	principe du tourniquet à cycle fixe
first fit:	la premiére zone libre
best fit:	le meilleur ajustement
worst fit:	le plus grand résidu
flag:	sémaphore
évents:	évènements
task:	une tâche
hand-shake:	procédure d'échange en poignée de main

daisy-chain:	interruptions chaînées
bootstrap:	initalisation du système
relocatable:	relogeable
context switching:	changement de contexte
process control bloc:	bloc de contrôle de la tâche
scheduler:	ordonnateur
dispatcher, job controller:	distributeur de tâche
driver:	pilotes (de périphériques)

ANNEXE III: Liste des fonctions constituant le logiciel d'exploitation du terminal

Ce descriptif est destiné au développeur d'applications pour le terminal domotique , et regroupe toutes les fonctions développées à ce jour pour constituer les trois couches logicielles du logiciel d'exploitation de l'E.C.A.D.
Les informations portent sur:

- -le langage d'écriture.
- -le rôle et l'utilisation de la fonction.
- -les particularitées d'écriture de la fonction.
- -les paramètres d'entrée/sortie et les mode de passage de ceux-ci.
- -les registres internes du microprocesseur utilisés, et leurs éventuelles modifications.

Fonctions de l'exécutif multitâche

Fonctions de gestion multitâche

PROG_NIV_INTERRUP:
Fonction système
Fonction de programmation du niveau d'interruption dans le registre SR (niveau évolué à placer en routine d'interruption car sauvegarde des drapeaux xnvz...)
Paramètre d'entrée: -le niveau d'interruption programmé
Passage de paramètre: par la variable globale NIV_INTERRUP
Paramètre de sortie: aucun
Protégée: -par masquage interruption

SUPERVISEUR:
Fonction système
Fonction de positionnement en mode SUPERVISEUR en registre SR
Parametre d'entrée: aucun
Paramètre de sortie: aucun

UTILISATEUR:
Fonction système
Fonction de positionnement en mode UTILISATEUR en registre SR
Parametre d'entrée: aucun
Paramètre de sortie: aucun

PROG_NIV7_INTERRUP:
PROG_NIV5_INTERRUP:
PROG_NIV2_INTERRUP:
PROG_NIV0_INTERRUP:
Fonctions système
Fonction de positionnement du masque d'interruption au niveau 7, 5, 2,0
Paramètre d'entrée: aucun
Paramètre de sortie: aucun

SYSTEME_TRAP:
Fonction système
Fonction de retourd à une couche de niveau inférieur, pour accés aux fonctions du superviseur.
Paramètre à passer:
Passage de paramètre:

LANCEUR:
Fonction système
Fonction de lancement du processus à exécuter

Il est nécessaire pour cela:
-de sauvegarde du pointeur courant de la pile système, et l'installation de la pile locale du processus. Cette opération doit impérativement être faite en premier car l'adresse de base de la premiére intruction sera écrite dans cette pile locale processus.
-de mettre en place l'accées indirecte (par A6) à la zone données non initialisées.
-de modifier l'adresse de retour du sous-programme en pile système, pour le branchement au programme processus à exécuter).
Paramètres d'entrée : D1-D2-D3
Passage des paramètres: par les registres internes
Paramétres de sortie: aucun
Protection: contre toute interruption de niveau 5 et 2
Registres internes: -utilisés: D1,D2,D3,D7,A6,A7
-modifiés: A7

STRUCTURE_PRO:
Fonction de création de la structure composée du: -n° de processus, & du -n° d'enregistrement chargée en file d'attente d'exécution.
Paramètre d'entrée:-le n° du processsus (NUM_PRO).
-le n°d'enregistrement (NUM_ENREGISTRE).
Passage de paramètre: par variable globale -NUM_PRO
-NUM_ENREGISTRE
Paramètre de sortie: aucun

DEROUTEMENT:
Fonction système
Fonction de retour dans la file appropiée aprés exécution d'un processus.
Paramètres d'entrée : aucun
Paramétres de sortie: aucun
Protection: cpntre toutes interruptionsde niveau 5 et 2
Registres internes: -utilisés:
-modifiés:

REST_REG_G0:
REST_REG_G1:
REST_REG_G2:
REST_REG_G3:
Fonctions de restauration des registres internes du processeur (Utlisées pour éviter des pertes d'informations lors de manipulations des registres internes entre les fonctions développées en différents langages).
Par soucis d'efficacité les registres sont divisés en quatre groupes
-> groupe 0:D0-D3 ; -> groupe 2: A0-A3
-> groupe 1:D4-D7 ; -> groupe 3: A4-A7
Paramètres d'entrée : aucun
Paramétres de sortie: aucun
Protection:
Registres internes: -utilisés:
-modifiés:

SAUV_REG_G0:
SAUV_REG_G1:
SAUV_REG_G2:
SAUV_REG_G3:
Fonctions de sauvegarde des registres internes du processeur (utlisées pour éviter des pertes d'informations lors de manipulations des registres internes entre les fonctions développées en différents langages).
Par soucis d'efficacité les registres sont divisés en quatre groupes
-> groupe 0:D0-D3 ; -> groupe 2: A0-A3
-> groupe 1:D4-D7 ; -> groupe 3: A4-A7

Paramètres d'entrée : aucun
Paramétres de sortie: aucun
Protection:
Registres internes: -utilisés:
-modifiés:

REST_CONT_SYS:
Fonction système
Fonction de restauration du contexte du processus systeme chargement en registres des pointeurs courants des files d'attente.
Paramètres d'entrée : le contexte système
Passage de paramètres: par les registres internes
Paramétres de sortie: aucun
Protection: contre tout interruption de niveau 5 et 2
Registres internes: -utilisés: A7
-modifiés: A0,A1,A2,A3,A4,A5,A6

SAUV_CONT_SYS:
Fonction système
Fonction de sauvegarde du contexte du processus systeme déchargement des registres utilisés pour les pointeurs courants de files d'attente, en mémoire.
Paramètres d'entrée : aucun
Paramétres de sortie: le contexte système
Passage de paramètres: par les registres internes
Protection: contre tout interruption de niveau 5 et 2
Registres internes: -utilisés:A0,A1,A2,A3,A4,A5,A6,A7
-modifiés:A7

REST_CONT_PRO:
Fonction système
Fonction de restauration du contexte d'un processus à partir des deux tableaux descripteur;
-de processus
-d'allocation mémoire
en registres dédiés: D0 pour l'info processus
D1 pour l'adresse d'accés au code
D2 pour l'adresse d'accés aux données
D3 pour l'adresse d'accés à la pile
Paramètres d'entrée : -le numéro du processus
-le numéro de l'enregistrement
Passage des paramètres: par variable globale: -NUM_PRO
-NBRE_ENREGISTRE
Paramétres de sortie: aucun
Protection: contre toute interruption de niveau 5 et 2
Registres internes: -utilisés: D4,D5,D7,A5
-modifiés: D1,D2,D3,D4,D5,D7,A5

SAUV_CONT_PRO:
Fonction système
Fonction de sauvegarde du contexte d'un processus dans les deux tableaux descripteur;
-de processus
-d'allocation mémoire
à partir des registres dédiés: -D0 info processus
-D1 pointeur d'accés zone code programme
-D2 pointeur d'accés zone données
-D3 pointeur d'accés zone pile locale
Paramètres d'entrée : -le numéro du processus
-le numéro de l'enregistrement

Passage des paramètres: par variable globale: -NUM_PRO
-NBRE_ENREGISTRE
Paramétres de sortie: aucun
Protection: contre toute interruption de niveau 5 et 2
Registres internes: -utilisés: D1,D2,D3,D4,D5,D7,A5
-modifiés: D4,D5,D7,A5

TRANSF_DAM_DCP:
Fonction système
Fonction de transfert des adresses de base d'accés, aux zones données et pile, du descripteur d'allocation mémoire au descripteur de contexte du processus. Cette fonction n'est utilisée que lors d'une premiére exécution d'un processus défini par -son n° de processus
-son n° d'enregistrement
Paramètres d'entrée : -le numéro du processus
-le numéro de l'enregistrement
Passage des paramètres: par variable globale
-NUM_PRO
-NBRE_ENREGISTRE
Paramétres de sortie: aucun
Protection:
Registres internes: -utilisés: D4,D5,D6,A5
-modifiés: D4,D5,D6,A5

TRANSF_DCP_DAM:
Fonction système
Fonction de transfert des adresses de base d'accés, aux zones données et pile, du descripteur de contexte du processus au descripteur d'allocation mémoire. Cette fonction n'est utilisée que lors d'une premiére exécution d'un processus défini par -son n° de processus
-son n° d'enregistrement
Paramètres d'entrée : -le numéro du processus
-le numéro de l'enregistrement
Passage des paramètres: par variable globale: -NUM_PRO
-NBRE_ENREGISTRE
Paramétres de sortie: aucun
Protection:
Registres internes: -utilisés: D4,D5,D7,A6
-modifiés: D4,D5,D7,A6

LIB_SEM_VDP:
Fonction système
Fonction de remise à zéro du sémaphore dédié VDP, puis retour à l'odonnanceur pour exécution d'un autre processus. En fait l'horloge interrompt l'exécution et voyant que la zone code est critique et est protégée par un sémaphore l'exécution est alors reprise jusqu'à la sortie de la zone critique. Cette fonction décide: -soit de poursuivre l'exécution du processus.
-soit d'interrompre l'exécution du processus et le système reprend le contrôle des opération et relance un nouveau processus.
Paramètre d'entrée: aucun
Paramètre de sortie: aucun
Protection:
Registres internes: -utilisés: D0,D1,D2,D3,A7
-modifiés: D0,D1,D2,D3,A7

LIB_SEM_RS232:
Fonction système
Fonction de remise à zéro du sémaphore dédié RS232, puis retour à l'odonnanceur pour exécution d'un autre processus. En fait l'horloge interrompt l'exécution et voyant que la zone code est critique et

est protégée par un sémaphore l'exécution est alors reprise jusqu'à la sortie de la zone critique. Cette fonction décide: -soit de poursuivre l'exécution du processus.
-soit d'interrompre l'exécution du processus et le système reprend le contrôle des opération et relance un nouveau processus.
Paramètre d'entrée: aucun
Paramètre de sortie: aucun
Protection:
Registres internes: -utilisés:
-modifiés:

LIB_SEM_RAM:
Fonction système
Fonction de remise à zéro du sémaphore dédié RS232, puis retour à l'odonnanceur pour exécution d'un autre processus. En fait l'horloge interrompt l'exécution et voyant que la zone code est critique et est protégée par un sémaphore l'exécution est alors reprise jusqu'à la sortie de la zone critique. Cette fonction décide: -soit de poursuivre l'exécution du processus.
-soit d'interrompre l'exécution du processus et le système reprend le contrôle des opération et relance un nouveau processus.
Paramètre d'entrée: aucun
Paramètre de sortie: aucun
Protection:
Registres internes: -utilisés:
-modifiés:

Fonction système
Fonction de test et de positionnement du bit n°7 d'état du sémaphore dédié VDP.
Cette fonction est placée au début d'une zone critique et selon l'état du sémaphore décide d'autoriser ou pas l'accés au processus de cette zone critique:
-bit n°7=0: entrée en zone critique pour l'exécution du processus.
-bit n°7=1: sauvegarde du contenu des registres du yp et du contexte du processus avant rechargement en fin file et déroutement vers l'ordonnanceur.
Paramètre d'entrée: aucun
Paramètre de sortie: aucun
Protection:
Registres internes: -utilisés: D0,D1,D2,D3,A7
-modifiés: D0,D1,D2,D3,A7

TEST_SEM_VDP:
Fonction système
Fonction de test et de positionnement du bit n°7 d'état du sémaphore dédié RAM. *
Cette fonction est placée au début d'une zone critique et selon l'état du sémaphore décide d'autoriser ou pas l'accés au processus de cette zone critique:

EXTRAC_ETAT_PRO:
Fonction système
Fonction d'extraction de l'information quantum de temps du processus (3bits)
Protection:
Registres internes: -utilisés: D0,D5
-modifiés: D5

EXTRAC_TEMPS_PRO:
Fonction système
Fonction d'extraction de l'information numéro du processus (7bits)
Protection:

Registres internes: -utilisés: D0,D5
-modifiés: D5

EXTRAC_NUM_PRO:
Fonction système
Fonction d'extraction de l'information type d'alarme incidente
Paramètre d'entrée: aucun
Paramètre de sortie: aucun
Protection:
Registres internes: -utilisés: D5,A6
-modifiés: D5

EXTRAC_TYPE_ALARME:
Fonction système
Fonction d'extraction de l'information portant sur le nombre d'enregistrements du processus (nombre total).
Paramètre d'entrée: aucun
Paramètre de sortie: le nombre d'enregistrement d'un processus particulier.
Protection:
Registres internes: -utilisés: D0,D5
-modifiés: D5

EXTRAC_NBRE_ENREGISTRE:
Fonction système
Fonction d'extraction du numéro d'enregistrement du processus dans le descripteur de mémoire allouée.
Paramètre d'entrée: aucun
Paramètre de sortie: le nombre d'enregistrement d'un processus particulier.
Protection:
Registres internes: -utilisés: D0,D5
-modifiés: D5

EXTRAC_NUM_ENREGISTRE:
Fonction système
Fonction de modification de l'état d'un processus dans le tableau descripteur de processus.
Les différents états possibles d'un processus et leur codage sont :

- -endormi : 000
- -en attente : 001
- -suspendu : 010
- -interrompu : 011
- -prêt : 100
- -actif : 101
- -terminé : 110

Cette fonction admet comme paramètre d'entrée le nouvel état du processus (technique: variable globale) .
Protection:
Registres internes: -utilisés: D0,D5,D6
-modifiés: D0,D5,D6

MODIF_ETAT_PRO:
Fonction système
Fonction de modification du nombre de quantums utilisé par un processus dans le tableau descripteur. Son but est d'incrémenter d'une unité cet indicateur à chaque nouvelle exécution du processus.
Protection:
Registres internes: -utilisés: D0,D5,D6
-modifiés: D5,D6

MODIF_INC_TEMPS:
Fonction système
Fonction de réallocation de deux quantums de temps aprés consommation de quatre unités. On modifie le mot long d'information du processus dans le descripteur de processus.

Protection:
Registres internes: -utilisés: D0,D6,D7
-modifiés: D6,D7

MODIF_TEMPS_PRO:
Fonction système
Fonction de modification du nombre d'enregistrements associés au processus en descripteur de contexte de processus: (modification de D0)
Paramètre d'entrée: D0
Paramètre de sortie: D0
Protection:
Registres internes: -utilisés: D0,D5
-modifiés: D0,D5

INC_NBRE_ENREGISTRE:
Fonction système
Fonction de modification du nombre d'enregistrements associés au processus en descripteur de contexte de processus: (modification de D0)
Paramètre d'entrée: D0
Paramètre de sortie: D0
Protection:
Registres internes: -utilisés: D0,D5
-modifiés: D0,D5

Fonctions de gestion de la mémoire

Pilotes de périphériques

CODE_VITESSE:
Fonction de gestion du port série.
Développement en langage C
Fonction de codage de la vitesse de transmission sur le bus série pour le sélecteur de fréquence.
Par rapport à la fréquence de référence (2,4576 Mhz) il est possible de sélectionner des vitesses multiples de deux de 19,2 Kbit/s à 150 bits/s.
Paramètres d'entrée: -la fréquence de transmission indiquée par l'utilisateur (VITESSE)
Passage de paramètre: par variable globale VITESSE
Paramètre de sortie: -le code de programmation du sélecteur (CELER).
Passage de paramètre: par variable globale : CELER

CODE_FORMAT:
Fonction de gestion du port série
Développée en langage C.
Fonction de codage du format d'émission et de réception sur la ligne série, pour programmation du registre de contôle du port série.
Paramètres d'entrée: -nombre de bits de données (DONNEES)
-parité (PARITE).
-nombre de bits de stop (STOP).
Passage de paramètres:-par variables globales
Paramètres de sortie: -le code de programmmation du format (FORMAT)
Passage de paramètres:-par variable globale

PRETRAITEMENT
Fonction de gestion du protocole de communication avec le réseau domotique.

Développée en langage C.
Fonction de traitement de la trame de communication reçue du serveur (trame ECAD 1ére version)
Le principe de ce traitement repose sur l'incrémentation et le test d'un compteur d'opérations de traitement.
Les test sur la trame de communication portent sur:

-la reconnaissance du drapeau de début de trame (ou de synchronisation).
-la reconnaissance de l'adresse du terminal.
-la reconnaissance du service
-la reconnaissance de la longueur du champ de données.

Ceux-ci étant effectués, les octets sont alors enregistrés.
-la reconnaissance de l'octet de fin de trame qui implique alors un nombre d'opérations: la génération d'un accusé de réception, la commutation de tampon, le chargement en file d'attente d'un processus, de gestion différencié des enregistrements.
Paramètres d'entrée:
Passage de paramètres:-par variables globales
Paramètres de sortie: -le code de programmmation du format (FORMAT)
Passage de paramètres:-par variable globale

Fonctions de l'interpréteur de commande

Processus Systèmes et Processus Utilisateurs

Processus de messagerie
Processus de télérelevé de consommation de fluide et d'énergie
Processus de télévote
Processus de multiprogrammation

ANNEXES IV: TABLE DES VARIABLES GLOBALES UTILISEES DANS LE LOGICIEL

Cette table recense toutes les variables globales utilisées dans le logiciel d'exploitation, et la taille de ces variables.

INSTALLATION_SYS:	-8 bits -variable globale indicateur d'installation complète et correcte du système multitâche.
ETAT:	-8 bits -indicateur d'état du processus (utilisé comme variable de programmation).
QUANTUM:	-8 bits -variable de programmation ou de indicateur du nombre de quantum de temps consommé par le processus.
NBRE_ENREGISTRE:	-8 bits -variable globale indicateur du nombre d'enregistrement du processus.
NUM_ENREGISTRE:	-8 bits -variable globale indicatrice du noméro d'enregistrement du processus.
NIV_INTERRUP:	-8 bits -indicateur de niveau d'interruption programmé.
ZONE:	-8 bits -indicateur de zone mémoire concerné soit par une opération de désallocation, soit d'extension, soit de réduction.
PILE_SYSTEME:	-8 bits -adresse de base de la zone de sauvegarde de la pile système.
SEM_VDP:	-8 bits -sémaphore de section code critique de programmation du VDP.
SEM_I2C:	-8 bits -sémaphore de section code critique de programmation des composants I2C.
SEM_REC_RS232:	-8 bits -sémaphore de section code critique de programmation du port série en réception.
SEM_EMIS_RS232:	-8 bits -sémaphore de section code critique de programmation du port série en émission.
SEM_MEM1:	-8 bits -sémaphore de section code critique d'accées à une zone mémoire RAM. programmation du VDP
COMPT_PRO_FILE:	-8 bits -compteur de processus en file d'attente.
RESIDENCE:	-8 bits -indicateur d'implantation mémoire du processus courant.

ALARME: -8 bits
-type de l'alarme programmée.

NBRE_ENREGISTRE: -8 bits
-nombre d'enregistrements du processus.

ENREGISTREMENT: -8 bits
-

ETAT_ANT: -8 bits
-état antérieur d'un processus.

PRIOR: -8 bits
-indicateur de priorité.

NUM_PRO: -8 bits
-numéro du processus courant.

CELER_RS232: -8bits
-vitesse de transmission série (encodage réalisé pur programmation de l'ACIA)

NB_ALARMES: -8 bits
-nombre d'alarme dans la file de programmation des alarmes.

CARACTERE: -8 bits
-code télécommande IR.

P_TAMPON_RS232: -4*8 bits
-pointeur du tampon de réception des données en provenance du port série.

TAMPON_RECEP_RS232: -8bits
-tampon de réception des données en provenance du port série.

TAMPON_EMIS_RS232: -8bits
-tampon d'émission des données sur le port série.

FORMAT: -8 bits
-code format de transmission.

VITESSE: -8 bits
-paramètre de programmation de la vitesse de transmission sur la ligne série.

CELER: -8bits
-code de programmation de la vitesse de transmission du port série.

PARITE: -8bits
-paramètre de programmation; choix de détection d'erreur de transmission.

STOP: -8bits
-paramètre de programmation d'un élément de format de transmission.

DONNEES: -8bits
-paramètre de programmation de taille des données transmises.

FORMAT: -8bits
-code de programmation du format de transmission des données série.

ADR_TERMINAL:	-8bits -adresse du terminal.
CENTIEMES:	-8bits -programmation des centièmes de seconde pour l'horloge.
SECONDES:	-8bits -programmation des secondes pour l'horloge.
MINUTES:	-8bits -programmation des minutes pour l'horloge.
HEURES:	-8bits -programmation des heures pour l'horloge.
DATE:	-8bits -programmation de la dâte pour l'horloge.
ANNEE:	-8bits -programmation de l'année pour l'horloge.
CENTIEMES_A:	-8bits -programmation des centièmes de seconde pour l'alarme.
SECONDES_A:	-8bits -programmation des secondes pour lalarme.
MINUTES_A:	-8bits -programmation des minutes pour l'alarme.
HEURES_A:	-8bits -programmation des heures pour l'alarme.
DATE_A:	-8bits -programmation de la dâte pour l'alarme.
ANNEE_A:	-8bits -programmation de l'année pour l'alarme.

ANNEXE V: Fiche signalétique du logiciel d'exploitation du terminal

Exécutif multitâche

Gestion multiprogrammée des tâches

-L'exécutif comporte deux mécanismes d'ordonnancement des processus.
-Les processus externes correspondent à des traitements impliquant un réponse immédiate.
-Les processus internes sont des programmes ou des éléments de programmes applicatifs de traitement adapté à la charge de travail du processeur.
-La gestion de l'exécution des processus internes est effectuée par priorité et selon le principe du tourniquet.
-Les deux files d'attente sont affectées chacune d'un niveau de priorité.
-Capacité (minimale) de chaque file d'attente: 20 structures processus (n°de processus + n° d'enregistrement).
-Les processus internes prioritaires grand consommateur de temps d'exécution sont pénalisés par un dispositif de repport d'exécution.
-Période d'horloge préemptive des processus: 30 à 35 ms.
-Sections critiques protégées par sémaphores simples.
-Sections critiques interruptibles mais exécutables entièrement.
-Modèle d'échanges producteur-consommateur des données entre processus.
-Différents types de processus (8 possibilités) qualifiés par trois paramètres (résident/importé, système/utilisateur, enregistrant/non-enregistrant).
-Accés aux fonctions du systèmes par exception.
-Tout échange entre les trois couches logicielles, fait intervenir par interruption ou exception le noyau de l'exécutif.
-Chaque processus est référencé par un élément descripteur renfermant les informations à caractère statique du processus (résident/importé...).
-Les processus externes sont ordonnés en exécution par le mécanisme microprogrammé des interruptions du µp.
-Les processus externes sont des processus de priorité supérieure à ceux des processus internes. Leur nombre est limité par celui des entrées d'interruption. Les processus externes sont des processus courts orientés communication et à traitement quasi immédiat.
-La répartition des interruptions est telle que le terminal domotique est en écoute permanente du réseau Médiabus (niveau 7). Les composant intervenant sur le bus I2C peuvent intervenir par une interruption commune de niveau inférieur, mais impliquant nécessairement un traitement..

Gestion mémoire

-Allocation et désallocation automatiques et adaptée des espaces mémoire nécessaires à l'exécution d'un processus selon ses caractéristiques (type, nature, fonction).
-Optimisation de la gestion mémoire par processus internes, systèmes non-préemptif;
-processus de compactage.
-processus d'extension.
-Gestion mémoire dynamique avec allocation compacte et contigüe des espaces mémoire.
-Représentation de la cartographie mémoire par un tableau descripteur d'allocation mémoire, constitués d'éléments en nombre égal au nombre d'enregistrements (d'espace de travail) présents en mémoire.
-Chaque élément descripteur d'allocation mémoire renferme les information dynamiques propre à l'exécution du processus lancé (état, N°d'enregistrement...).

ANNEXE VI: Description de quelques systèmes d'exploitation et exécutifs

OS-9/68000:

-système d'exploitation multitâche, multiutilisateur à structure parallèle.
-intégrable en ROM
-modulable:tout code objet exécutable est organisé sous forme d'un module mémoire, le rendant configurable et adaptable à toute configuration .
-portable : 90% du code est écrit en langage C (shell, ensemble des commandes...) le rendant réentrant, et le noyau ainsi que les modules de gestion sont écrits en assembleur (code compacte, et rapide d'exécution)..
-adapté à la réalisation d'application industrielle temps réel en environnement UNIX (les programmes source son compatibles en langage C du fait de la compatibilité des bibliotèques et d'une organisation similaire de OS9 et UNIX).
-conçu pour les microprocesseur de la famille 68000
-architecture générale:
    -le noyau exécutif:effectuant la gestion des E/S, du multitâche, de la mémoire, des liens dynamiques avec les autres modules système, des interruptions, des appels système. Le noyau est identique quel que soit la machine sur laquelle il tourne . Il est en position -code indépendant et romable.
    -deux modules système (spécifiques au matériel hôte):
        _horloge: spécifique au circuit horloge temps réel
        _ initialisation: gérant le tableau des périphériques, et précisant la taille des différentes tables uitlisées par le système.
    -de modules librairies constituées de fonctions élémentaires utilisées par plusieurs programmes (librairie de fonctions mathématiques et de fonctions d'entrées/sorties formatées).

Cette premiére partie (taille minimum=13 Ko) est complétée par une couche logicielle supérieure constituée de gestionnaires de fichiers, de circuits (pilotes de périphérique; disque dur et disquette, console, imprimante,moniteur graphique, pipe-line), de descripteurs de périphériques (12 Ko enviro et le shell interpréteur (15 KO).
-la gestion mémoire logique et physique est dynamique. La gestion physique concerne toute la mémoire vive détectée à la mise sous tension. Les techniques "d'overlay" et de "swapping" de gestion logique ne sont pas utilisées si la mémoire centrale est de taille suffisante.
-la mémoire allouée à chaque utilisateur est de 64 Ko .
-à tout processus est associé deux zones mémoire distinctes; une zone code objet et une zone données comportant la pile du processus.
-un signal d'horloge intervient toutes les 10 ms. A toute occurence d'un signal d'horloge, l'exécutif peut suspendre l'exécution d'un processus et reprende l'exécution d'un autre. Un processus actif est exécuté pendant une période de temps dépendante de son niveau de priorité relative par rapport aux autres processus actifs.
-pour des aplications temps réel critique, OS9 utilise le principe de la préemption. Un processus de niveau de priorité supérieur, prend le pas sur un processus de niveau de priorité plus basse dont l'exécution est interrompue et reportée ultérieurement, avec rechargement en fin de file.
-les appels système intégrés font le lien entre le noyau et l'utilisateur et concerne la gestion mémoire, la synchronisation entre processus (mode superviseur et utilisateur), la gestion du temps, la gestion des E/S.
-la gestion des fichiers est l'une des partie visible pour l'utilisateur. L'organisation des fichiers y est arborescente (regroupement hiérarchique des fichiers en catalogues avec un répertoire racine) ce qui protège les fichiers entre différents utilisateurs. Ce système comporte un dispositif de verrouillage d'accés simultannée à un même fichier par plusieures tâches, et une sécurité contre les accés non autorisés. OS9 sait gérer des unités de stockage sur disque, physique et logique. Chaque fichier est affecté d'indicateurs de propriété, de type, de classe, de droit d'accés aux fichiers, en lecture et/ou écriture, et en exécution, d'une dâte de création, d'un numéro utilisateur.

-le shell est un interpréteur de commande dont la fonction est de contrôler l'exécution des autres programmes d'OS9 à partir des données lues sur le port d'entrée standart (clavier). Ces commandes permettent le lancement et l'arrêt d'une tâche, le positionnement des priorités, la redirection des E/S, le traitement des fichiers de commande, l'utilisation du "pipi line"...

quelques détails...

-module de gestion mémoire:

Outre le fait qu'OS9 assure la répartition physique de la mémoire entre les différentes tâches, l'organisation logique est assurée par modules mémoire. Un module est constitué de trois parties:

->un en-tête qui renferme des informations descriptive du module et son utilisation (nom, taille,propriétaire, accés type, langage, n°d'édition). Il est défini durant la phase de développement des programmes (opération de linkage).

->le corps du module contient les données initialisées et le code programme du processus (écrit en code position indépendant et réentrant).

->un CRC vérifiant l'intégrité du module. Ce crc contient le nombre d'octet de longueur du module.

Tout programme, ou ensemble de données doit présenter cette structure pour être chargé en mémoire.

Les fichiers sont organisés en répertoire, et un fichier particulier; le fichier répertoire contient toutes les références des modules (les en-têtes), ainsi que l'adresse et le nombre de tâches que sollicite le module (nombre de liens). Un module ayant un nombre de liens égal à zéro fait l'objet d'une désallocation mémoire.

-entrées/sorties:

Quel que soit le langage de programmation utilisé, l'opération de lecture (ou d'écriture) sur un périphérique s'écrit ou se lit de la même façon qu'il s'agisse d'un clavier ou d'un fichier disque (ou d'un moniteur de visualisation). La seule différence porte sur la désignation du périphérique adressé. Le traitement des Entrées/Sorties interesse aussi bien le noyau, que le gestionnaire général des d'Entrées/Sorties et le gestionnaire de fichiers afin d'assurer les transferts de données. Les pilotes de périphériques sont considérés comme des modules programme. Les adresses des périphériques, les adresses des drivers correspondant, les niveaux d'interruption, les priorités d'interruption, et les paramètres d'utilisation du port (mode d'utilisation, options) sont référencés en descripteurs de périphériques.

-gestion des interruptions:

Les flux de caractère sont gérés lorsque nécessaire (périphériques SCF: ports série ou parallèle)par l'intermédiaire de deux tampons, l'un pour les entrées et l'autre pour les sorties. Pour économiser le temps CPU il souhaitable de traiter les échanges d'informations en entrées/sorties sous mode interruption. Différentes tables sont utilisées pour référencer et mettre en correspondance les niveaux d'interruption, les nom de périphériques ouverts, le numéro de vecteur, l'adresse de la routine de traitement de l'interruption, l'adresse du port périphérique, un pointeur sur le tampon de réception ou d'émission. Plusieurs périphériques peuvent êtres installés sur le même niveau ou sur le même vecteur d'interruption.

-gestionnaire de réseau:

Conformément à la philosophie modulaire d'OS9, le rendant configurable par rapport à la machine hôte, un module de gestion de réseaux a été développé, utilisant n'importe quel support de liaison. Plusieurs réseaux de même type peuvent être gérés. Le type (jeton, bus, étoile) et les caractéristiques du réseau sont indifférents pour OS9. Le noyau et le gestionnaire de réseau (file manager) correspondent aux couches 5 (session), 6 (présentation), 7 (application) du modèle ISO-OSI.

-gestion du multitâche:

Le noyau est chargé de la gestion des tâches en tenant compte de leur priorité.A chaque signal d'horloge, l'ordonnanceur ou répartiteur de tâches (Scheduler) intervient pour (dans l'ordre):

-examiner séquentiellement les priorités des processus à exécuter et placés queue des tâches actives. Le processus à exécuter sera celui de priorité la plus élevée. La priorité est évaluée par l'opération: priorité=priorité initiale+âge. Ainsi il est garanti qu'une tâche de priorité faible sera exécutée. Une tâche étant exécutée durant une tranche de temps voie sa priorité revenir à sa priorité initiale. La priorité initiale fixée pour une tâche est une variable évoluant exponentielement.

-si le signal d'interruption est survenu et l'interruption non masquée, sauvegarder le contexte de la tâche interrompue, puis démarer la tâche suivante de priorité maximale.

Un tâche est caractérisée par:

-un contexte d'exécution: comprenant les informations strictement nécessaires au processeur pour en assurer l'exécution.

-un contexte mémorisé: qui définie la zone mémoire prôpre à assurer l'exécution de chaque tâche.

Une tâche est dans un état déterminé:

-inexistant: la tâche ne possède pas de descripteur.

-existant: la tâche possède un descripteur.

-exécutable:la tâche possède un descripteur et peut s'exécuter.

-non exécutable: la tâche possède un descripteur mais ne peut pas démarer ou continuer son exécution.

-en service:la tâche est exécutable et elle commencée son exécution mais ne l'a pas terminée.

-hors service: la tâche est exécutable mais ne l'a pas démarrée, ou son exécution n'est pas terminée.

-actif: la tâche est en service et n'attend que le processeur pour s'exécuter.

-prêt: la tâche est active et attend que le processeur soit libre pour s'exécuter.

-en attente: la tâche est en service est attend qu'une condition explicite soit satisfaite pour continuer son son exécution.

-en sommeil: la tâche est hors service pour une période de temps donnée ou jusqu'à ce qu'un signal lui parvienne.

Le descripteur de tâche précise:

-le nom du processus.

-sa priorité.

-son état.

-ses allocations mémoire.

-ses canaux d'entrées/sortie.

-communication entre tâche:peut s'effectuer de plusieures façons différentes.

-par modules de données: ce sont des blocs de mémoire partageables entre plusieurs tâches. La communication est rapide et souple, mais doit etre arbitrée.

-par signaux: utilisés pour transmettre rapidement un nombre limité d'informations. Ce sont des interruptions logicielles utilisées pour synchroniser le déroulement de plusieurs processus entre-eux ou avec des évènements externes.

-par pipelines: ce sont des tampon FIFO, pour stocker et transmettre des informations entre deux processus (liens).

-par disque RAM: on simule une unité de disquettte en RAM, ce qui rend les accées de lecture, écriture des données trés rapide.

-par disque magnétique: non utilisés pour des applications temps réels, ils permettent un accés aléatoire aux données et une grande capacité de stockage.

-l'aspect temps réel: est représenté par deux principes:

-de préemption:

-de gestion des évènements: les ressources partageables doivent être protégées d'un accés simultanné de deux processus. La synchronisation des accés est réalisée au moyen de sémaphores simples, indiquant au processeur si la ressource est disponible ou non. Le sémaphore encadre une zone critique (zone de code non partageable). Le sémaphore est un mécanisme d'exclusion mutuelle solutionnant en partie le partage des ressources.

La notion d'évènement, plus puissante que celle de sémaphore, .

REALTIME CRAFT:

-moniteur temps réel multitâche
-implantable en ROM
-utilisable sur microprocesseur 16bits d'INTEL, et microprocesseur 16 et 32 bits de MOTOROLA et NATIONAL SEMICONDUCTOR.
-standardisé à la norme SCEPTRE (Strandardisation du Coeur des Exeécutifs des Produits Temps Réel Européens.
-compacte ,il nécessite 3 Ko en mémoire.
-la période de fonctionnement de l'horloge système est choisie par l'utilisateur, entre 10 et 100 µs.
-un coprocesseur numérique est supporté par le moniteur.
-indépendant du système de développement et des langage de programmation utilisés pour réaliser les programmes d'application.
-constitue un support de développement d'applications.
-utilise un système de préemption des tâches.
-utilise un système de gestion des processus basé sur les priorité des tâches prêtes. Les performances du système dépendent du nombre de niveaux de priorité des tâches à exécuter.
-le moniteur est constitué de:
    -du noyau effectuant les opérations de gestion multitâche, de protection, de communication et de synchronisation des tâches, ainsi que le partage des ressources, la gestion des évènements, la prise en compte des temporisations et la liaison avec les interruptions.
    -de bibliothèques d'interfaces langage complètent les composants système portant le moniteur.
-extensions du moniteur:
    -un superviseur d'Entrées/Sorties.
    -un système de gestion de fichiers en accés séquentiel ou direct.
-le noyau peut assurer les fonction de gestion dynamique de la mémoire, par division de l'ensemble de la mémoire en pools statiques de mémoire. A chaque pool est associé une boîte aux lettre, et chaque pool est découpé en segments mémoire de taille fixe.
-les tâches disposent d'un service de temporisation dont il est possible d'indiquer le temps maximum d'attente.
-l'interface entre une application et son environnement matériel est réalisée par les routines d'interruption, ce qui limite la portabilité du moniteur. Ces routines d'interruption doivent se limiter à envoyer un message à la tâche de traitement de l'évènement source de l'interuption, à réinitialiser la source externe de l'interruption,. à démasquer les niveaux d'interruption ou à effectuer tout autre traitement réalisé dans un délai de 100µs.
-ce moniteur permet de concevoir une application temps réel comme un ensemble de modules autonomes structurés en tâches, et coopérant par l'entremise des primitives du moniteur.
-dans la phase d'initalisation du système, l'utilisateur doit définir la configuration de fonctionnement du moniteur (nombre de tâches et de niveaux de priorités, nombre de sémaphores et de bôite aux lettres...). L'ensemble de ces informations est vu sous forme d'un objet (propriété d'encapsulation).

Quelques précisions...

-protection entre tâches et partage des ressources:

Par sémaphore.

Les sections critiques sont encadrées par des sémaphores compteurs. Quand une tâche demande d'entrer en section critique, elle inhibe le macanisme de l'ordonnanceur des tâches. La tâche restera dans l'état courant tant qu'elle n'aura pas demandée l'opération de sortie en région critique, et elle disposera de façon exclusive l'accés à toutes les ressources libres du système temps réel. L'arrêt d'une tâche ne se produit qu'aprés sa sortie de région critique. Une tâche en région critique n'est pas autorisée à utiliser des opérations de temporisation ou d'attente. Les interruptions ne sont pas masquées en régions critiques.

Sur évènement.

Les opérations sur évènements sont préférées lorsu'une tâche a besoin de se synchroniser sur plusieurs évènements. Lorsqu'une tâche doit se synchroniser sur un évènement unique on peut, soit utiliser les opérations sur sémaphores à compteur, soit les opérations sur ènements. 16 évènements sont disponibles.

Communication entre tâches.
Pour s'échanger des données les tâches disposent de de primitives de gestion de boîtes aux lettres (5aux maximum).

-gestion mémoire:

-

-une tâche se trouve dans l'un des quatre états:

-non opérationnelle.
-en attente de réalisation d'une condition d'exécution.
-prête (la tâche n'est pas une priorité suffisante pour être exécutée).
-en cours d'exécution.

-les tâches utilisateur peuvent se répartir en plusieurs catégories (applications complexes) dont les tâches de gestion des périphériques (par type) affectées d'un niveau de priorité en rapport avec la vitesse de l'appareil géré, puis les programmes d'application proprement dits (prorité moindre), et enfin une tâche de fond, utilisée pour lancer des tests de fonctinnement,ou de statistisques de fonctionnement de l'appareil.

OS/2:

-système d'exploitation multitâche, monoutilisateur généraliste.
-implanté sur disque.
-utilisé pour les micro-ordinateurs IBM AT et compatibles, IBM-PS.
-conçu pour s'exécuter à partie des microprocesseurs de la famille INTEL 80x86.
-accompagé d'une interface utilisateur graphique: Présentation Manager.
-communication d'informations interprocessus.
-assure la gestion et l'exécution des programmes en mode réels (monotâche)et en mode protégé (multitâche) en adaptant la gestion mémoire à ces modes.
-assure la compatibilité avec les applications DOS (mode réel).
-technique de gestion mémoire virtuelle avec indirection.
-espace d'adressage de 16 Mo.
-nécessite 1,5 Mo de mémoire centrale.
-offre en mode protégé la possibilité d'utiliser une mémoire virtuelle.
-OS2 joue le rôle d'un superviseur, et pour cela exploite le mode protégé (lié à l'aspect multitâche) des processeurs 80x86 et l'ensemble des instructions réservées. Il contrôle le fonctionnement et l'accés aux périphériques, ainsi des commandes non autorisées ne peuvent êtres exécutées, et un programme désirant utiliser un périphérique devra attendre au préalable qu'un programme précédent ait terminé d'occuper celui-ci.
-pour assurer la compatibilité des programmes écrits sous DOS 3.3, le système peut fonctionner en mode réel (non multitâche). Pour cette raison on retrouve toutes les fonctions DOS dans ce mode.
-les priorités dans le CPUse répartissent en quatre couches en mode protégé.

>niveau 0:pour le noyau d'OS2
>niveau 1:pour les fonctions du système tournées vers l'utilisateur.
>niveau 2:pour les sous-systèmes.
>niveau 3:destinée aux programme d'application.

-la mémoire ne peut être adressée directement en mode écriture par un programme utilisateur. Elle doit l'être par l'intermédiaire d'un sélecteur qui en contrôle l'accés. Chaque programme se voit alloué un segment mémoire auquel il a seul accés. Les programme utilisateurs ne peuvent connaître les adresses physiques de stockage de leur données.
-en cas d'insuffisance d'espace mémoire principale, la méthode d'échange de segments mémoire ("swapping") est utilisée. C'est le segment mémoire qui n'a pas été utilisé pendant l'intervalle le plus long qui est placé sur le disque. Par ce mécanisme le le système peut gérer jusqu'à 1 Mo.
-la communication entre processus peut être réalisée en autre par des secteurs mémoire mis à la disposition de plusieurs programmes (mémoire partagée).
-le CPU avec l'aide de l'horloge décide du temps pendant lequel le CPU sera alloué à tel ou tel programme. L'utilisateur a la possibilité de choisir les durées minimale et maximale des tranches de temps d'exécution de chaque tâche.
-un "task manager" c'est à dire un gestionnaire de programme prend en compte les périphériques connectés et le, la tâche, et les diverses priorités.

-certaines applications sont plus gourmands en temps de CPU, ou peuvent revendiquer une priorité d'exécution du fait de l'urgence ou de la répétitivité de certains traitements. Aussi OS2 définit-il des niveaux de priorités affectés aux processus à exécuter. Le "dispatcher" préemptif choisit d'exécuter à chaque signal d'horloge le processus de rang de priorité le plus élevé. Si différentes tâches ont même niveau de priorité, c'est la tâche qui est restée la plus longtemps inactive qui est choisie. Trois priorités primaires différentes ont été définies:

>pour les processus critiques à traiter en urgencce.

>pour les processus standarts.

>pour les processus lents.

et à l'intérieur de cette hiérarchie on définie un niveau de priorité secondaire comprenant 32 échelons.les applications de rang le plus élevée passeront en premier pour les processus s'incrivant dans une hiérachie primaire. Lorsque toutes les applications urgentes auront été traitées

PSOS:

-exécutif temps réel.
-implantable en ROM.
-environnement de développement réduit.
-existence d'une passerelle entre UNIX et pSOS. Toutes les tâches temps réelles sont traitées par pSOS, et les autres sont traitées sous UNIX. Cette passerelle permet la synchronisation et le passage d'informations entre les deux systèmes.
-pSOS assure le la gestion des tâches, de la mémoire, du temps et des Entrées/Sorties.
-compact il occupe 4 Ko en mémoire.
-implanté sur des unités centrales à base de microprocesseur de la famille Motorola 680x0.
-accompagné d'un logiciel de mise au point Probe, lui auusi romable et complètement translatable.

THEOS:

MS DOS:

-système d'exploitation monotâche, monoutilisateur généraliste.
-implanté sur disque.
-utilisé dans les micro-ordinateurs IBM PC et compatibles (différentes versions pour microprocesseurs 8, 16, ou 32 bits), pour le développement de logiciels d'application, ou la conduite de petits processus industriels.
-contrôle le déroulement des processus internes de la machine.
-système de gestion de fichiers arborescent simple.
-limitation initiale de la taille mémoire centrale utilisable par le DOS à 640 Ko (mémoire basse) bien que les processeurs 8086, 80286 puissent gérer 1 Mo (mémoire haute réservée à la mémoire d'écran et aux périphériques).
-n'utilise que le mode réel de cette famille de processeur.
-pas d'accés prévus à un réseau d'échange d'informations.
-technique d'affichage à l'écran en E/S mémoire (un pixel=une adresse mémoire).
-le noyau est réduit à la gestion des Entrées/Sorties de base, et à la gestion des fichiers.
-l'utilisation des périphériques n'est pas contrôlé. Ainsi il n'y a aucun moyen d'examiner si l'imprimante connectée est ou non utilisée. MS-DOS n'intervient que sur requète de l'utilisateur ou d'un programme application.
-les interruptions sont reconfigurables.
-la fréquence de l'horloge interne est de 18.2 Hz.
-il est possible par adjonction d'une couche logicielle supplémentaire de doter le DOS de fonctions multitâches et éventuellement d'en faire un système temps réelle. Les fonctions alors introduites complètes les fonctions propres du DOS. Mais attention le code n'est pas" réentant".

RTE:

UNIX:

-système d'exploitation multitâche et multiutilisateur.
-système modulable conçu autor d'un noyau minimum (core).
-structuré en quatre couches:
-le noyau central; compact et non modifiable constitué des fonctions de base de la gestion des processus en temps partagé en fonction de priorités.
-les conducteurs d'Entrées/Sorties: Ces deux couches constituent le noyau complet.
-la troisième couche est constuée par l'interpréteur de commandes.
-la quatrième couche: constituée des applications et des utilitaires pour l'utilisateur.
-deux types d'environnement processus disponibles (donc deux niveaux d'actions possible).
->pour un processus utilisateur: en cas de besoins les fonctions système sont utilisables et il y a changement d'environnement.
->pour un processus système on fait intervenir d'autres fonctions du coeur du système en élargissant les possibilités, accessibles sous contrôle, offerte à l'application. L'ordonnancement est le type même de processus système.

Quelques détails...

-le mécanisme d'ordonnancement fait intervenir:
-une table des processus, résidente en RAM avec une entrée par processus (entrée allouée à la création du processus) et accessible uniquement par le noyau du système.
-l'ordonnanceur qui utilise le temps partagé et des niveau de priorité variables pour gérer l'exécution des processus. Le niveau de priorité d'un processus est inversement proportionnel au temps déjà passé en exécution. Toutes les priorités sont révues, réaffectées à la fin de chaque intervalle de temps. La file d'attente fonctionne selon le principe du tourniquet. La priorité est donnée aux processus utilisant des Entrées/Sorties. Un processus en attente (d'une donnée d'Entrées/Sorties par exemple) va obtenir l'unité centrale (le processeur) par préemption sur un processus de priorité plus faible.
-Unix est pourvu d'un système de fichiers hiérarchisé selon une structure arborecente comportant trois types de fichiers:
--ordinaires:
--catalogues: constituant les noeuds hiérarchiques et contenant la liste des noms de fichiers qui sont rattachés à ce catalogue. Chaque fichier est référencé par un numéro de fichier unique.
--spéciaux: utilisés par le Entrées/Sorties et catalogués en E/S structurées (ou E/S blocs) et en E/S non structurées (ou E/S caractères). Un périphérique de type bloc est est vu comme constué de n blocs de 512 octets accessibles de façon aléatoire.
-

MAC-OS

-système d'exploitation monoutilisateur, monotâche du Macintosh.
-pas de version multitâche à l'heure actuelle.
-marqué par une ergonomie trés poussée dés sa conception.
-structuré en systèmes de gestion trés modulaires hiérarchisées:
-le noyau du système comportant les gestionnaires d'évènements système, d'erreurs, de position du curseur pour déplacement de la souris, et des services système.
-la gestion de la mémoire
-la gestion des Entrées/Sorties contenant les gestionnaires de périphériques et les pilotes d'interfaces.
-la gestion des fichiers.
-la boîte à outils utilisateur: collection de 500 routines et fonctions écrites en assembleur, stockées en ROM et accessibles diretement pae l'unité centrale. L'accés à ces routines depuis un programme se fait par l'exécution d'une Trappe (exeption), caractérisé par un numéro utilisé comme index dans une table d'adresse. La boîte à outil est divisée en une douzaine de sous-ensembles gestionnaires .
-de dialogue -de texte élémentaire

-de contôle
-de la boîte à outils
-du bureau
-des ressources
-de paquets de caractères.
-des fenètres
-des évènements de la boîte à outils
-du presse-papier
-des jeux de caractère et dessin

ANNEXE VII: Schémas électroniques.

Comparaison du bus de fond de panier définit pour l'E.C.A.D et du bus VME

Date:24/01/90 Tableau de branchement des connecteurs

Disposition du connecteur N°3

N°	Rangée A	N°	Rangée C
A_1	D00	C_1	RTS
A_2	D01	C_2	CTS
A_3	D02	C_3	RXD
A_4	D03	C_4	TXD
A_5	D04	C_5	DCD
A_6	D05	C_6	NC
A_7	D06	C_7	Gnd
A_8	D07	C_8	Gnd
A_9	Gnd	C_9	Gnd
A_10	SYS-CLK	C_10	E
A_11	Gnd	C_11	Gnd
A_12	VPA	C_12	RESET/
A_13	DS/	C_13	IPL0/2
A_14	R/W	C_14	IPL1
A_15	Gnd	C_15	Gnd
A_16	DTACK/	C_16	SCL
A_17	Gnd	C_17	SDA
A_18	AS/	C_18	Gnd
A_19	Gnd	C_19	A19
A_20	IACK/	C_20	A18
A_21	IACK IN/	C_21	A17
A_22	IACK OUT/	C_22	A16
A_23	A07	C_23	A15
A_24	A06	C_24	A14
A_25	A05	C_25	A13
A_26	A04	C_26	A12
A_27	A03	C_27	A11
A_28	A02	C_28	A10
A_29	A01	C_29	A09
A_30	A00	C_30	A08
A_31	-12V	C_31	+12V
A_32	+5V	C_32	+5V

Noms des broches	Rangée A	Rangée B	Rangée C
1	D00	$\overline{BBSY}$	D08
2	D01	$\overline{BCLR}$	D09
3	D02	$\overline{ACFAIL}$	D10
4	D03	$\overline{BG0IN}$	D11
5	D04	$\overline{BG0OUT}$	D12
6	D05	$\overline{BG1IN}$	D13
7	D06	$\overline{BG1OUT}$	D14
8	D07	$\overline{BG2IN}$	D15
9	GND	$\overline{BG2OUT}$	GND
10	SYSCLK	$\overline{BG3IN}$	$\overline{SYSFAIL}$
11	GND	$\overline{BG3OUT}$	$\overline{BERR}$
12	$\overline{DS1}$	$\overline{BR0}$	$\overline{SYSRESET}$
13	$\overline{DS0}$	$\overline{BR1}$	$\overline{LWORD}$
14	$\overline{WRITE}$	$\overline{BR2}$	AM5
15	GND	$\overline{BR3}$	A23
16	$\overline{DTACK}$	AM0	A22
17	GND	AM1	A21
18	$\overline{AS}$	AM2	A20
19	GND	AM3	A19
20	$\overline{IACK}$	GND	A18
21	$\overline{IACKIN}$	SERCLK (1)	A17
22	$\overline{IACKOUT}$	SERDAT (1)	A16
23	AM4	GND	A15
24	A07	$\overline{IRQ7}$	A14
25	A06	$\overline{IRQ6}$	A13
26	A05	$\overline{IRQ5}$	A12
27	A04	$\overline{IRQ4}$	A11
28	A03	$\overline{IRQ3}$	A10
29	A02	$\overline{IRQ2}$	A09
30	A01	$\overline{IRQ1}$	A08
31	— 12 V	+ 5 V STDBY	+ 12 V
32	+ 5 V	+ 5 V	+ 5 V

Note : SERCLK et SERDAT sont prévues pour l'implantation d'un bus de communication série.

RTC

Note technique

CI 52

Les atouts du BUS I²C pour les applications de grande série

Beaucoup d'appareils électroniques actuels, destinés aux productions de grande série contiennent au moins un contrôleur – en général, un micro-ordinateur – et un certain nombre de circuits intégrés standard et spécialisés pour le stockage et l'affichage de données, et pour l'exécution de fonctions numériques et analogiques. Certes, les moyens utilisables pour interconnecter les circuits internes et connecter les circuits intégrés périphériques au(x) micro-ordinateur(s) sont nombreux, mais les concepteurs et fabricants d'appareils ont avantage à opter pour des interfaces de commande simples, standardisées, facilitant l'extension des matériels ou leur construction modulaire. C'est pourquoi, nous avons mis au point le bus I²C (Inter Integrated Circuits) et avons inclus l'interface matérielle de ce bus dans une large gamme de circuits destinés à toutes sortes d'applications : appareils TV et vidéo, systèmes audio et radio, postes téléphoniques, systèmes électroniques automobiles, appareils électroménagers, etc.

Le bus I²C est conçu avec un souci de souplesse, d'économie et d'efficacité. Comme le débit nécessaire aux fonctions de commande est faible, le bus transfère les données de commande en série, dans les deux directions, à une vitesse de 100 kbits/seconde maximum. Ceci ne nécessite que deux fils : un pour les données, l'autre pour l'horloge système. Par conséquent, seules deux broches par circuit suffisent et le câblage des cartes à circuit imprimé et des connecteurs de carte est simplifié. Cet arrangement ne limite pas le niveau d'intégration par boîtier et réduit le coût et l'encombrement des cartes. En outre, le bus I²C est un véritable bus multi-maître, permettant à plusieurs circuits connectés au bus d'en prendre le contrôle. Pour éviter la perte ou la dégradation des informations, une adresse unique est affectée à chaque circuit et une procédure d'arbitrage est prévue pour décider de la maîtrise des commandes. Lorsque des circuits intégrés à horloges rapides communiquent avec d'autres circuits à horloges lentes, le protocole synchronise l'horloge système en définissant la source de l'horloge.

Le bus I²C supporte une vaste gamme de micro-ordinateurs et de circuits intégrés périphériques réalisés dans différentes technologies de semi-conducteur. Il s'est imposé rapidement dans l'industrie comme un outil puissant pour la conception et le développement de systèmes.

CONDITIONS A REMPLIR PAR LE BUS

Transfert série des informations

Le transfert des données numériques à l'intérieur de l'appareil s'effectue sur un bus série bifilaire, comprenant une ligne de données et une ligne d'horloge. Ceci réduit les coûts d'interconnexion du fait de la simplification des cartes imprimées, et deux fils d'interconnexion suffisent lorsque les fonctions sont éloignées comme dans un système modulaire. Par exemple, lorsque la fonction de l'accord dans un poste de TV est implantée sur la carte principale, et la fonction de commande et d'affichage est commandée à partir du panneau avant. Les frais de connecteurs pour les liaisons entre cartes sont également réduits.

Un autre avantage du bus série bifilaire est qu'il ne requiert que deux broches sur le circuit intégré. Il reste donc davantage de broches disponibles pour réaliser une meilleure intégration au niveau de chaque circuit avec une économie sur le coût des boîtiers.

Transfert bidirectionnel des informations

Le bus série permet le transfert bidirectionnel des informations. Ceci découle directement des exigences fonctionnelles du système. Par exemple, dans un récepteur TV, une mémoire non volatile qui contient les numéros des canaux ou les réglages analogiques préférés doit pouvoir être lue et écrite via le bus.

Adressage binaire des circuits intégrés

Les modules ou circuits doivent avoir une adresse binaire unique, puisqu'il n'y a pas de ligne de validation dans un système bifilaire. Ceci laisse au contrôleur davantage d'Entrées/Sorties disponibles pour d'autres besoins.

Accusé de réception (ou Acquittement)

Le système de bus doit permettre au récepteur d'accuser réception d'un message. Des contrôles peuvent ainsi être effectués sur le transfert des données, et la configuration du système peut être déterminée par un contrôleur maître.

Fonctionnement multi-maître

Il est indispensable qu'un bus fonctionne en mode multi-maître. Un maître est un dispositif qui peut prendre l'initiative d'un transfert de données. Le mode multi-maître permet à plusieurs dispositifs de prendre le contrôle du bus pour démarrer un transfert. Ceci nécessite la présence d'un système d'arbitrage pour éviter la perte ou la détérioration des informations lorsque plusieurs maîtres essaient de prendre le contrôle du bus simultanément.

En mode multi-maître, il n'est pas nécessaire de prévoir un maître central (généralement un micro-ordinateur) pour gérer l'ensemble des fonctions. Cela signifie que l'extension des fonctions d'un système ou sa modularisation n'affectent pas le système originel. Les nouvelles fonctions et le logiciel supplémentaire nécessaire sont placés dans un contrôleur maître supplémentaire, simplement rajouté au bus.

Alors que dans un système à un seul maître, le contrôleur central doit transmettre constamment des données, dans un système multi-maître, les contrôleurs ne doivent transférer des données que si elles ont une importance mutuelle. Ceci réduit le volume des données transportées par le bus et permet d'utiliser le bus de façon plus efficace. Dans un système multi-maître, qui possède une intelligence répartie inhérente, les interruptions temps réel peuvent être traitées localement et n'affectent donc pas l'ensemble du système.

Le fonctionnement multi-maître simplifie aussi les procédures de diagnostic pendant lesquelles un maître supplémentaire peut temporairement se connecter au bus pour exécuter les programmes de contrôle ou de réglage.

Possibilité d'une interface esclave économique

Un autre point important est que le bus doit permettre la mise en œuvre d'une interface matérielle esclave économique. Un esclave est une unité commandée par le maître contrôlant la transmission en cours.

Le coût de l'interface matérielle du bus revêt une importance primordiale puisqu'il concerne chaque CI connecté au bus. Par exemple, l'interface ne doit pas nécessiter la présence d'un oscillateur interne dans l'esclave, car celui-ci serait inutile pour les autres fonctions remplies par ce circuit.

Protocole standardisé

Le protocole du bus doit être standardisé pour permettre une réalisation modulaire du logiciel.

LE BUS I²C

Définition de la terminologie du bus I²C

Emetteur : – Unité qui envoie les données sur le bus

Récepteur : – Unité qui reçoit les données du bus

Maître : – Unité qui démarre un transfert, génère des signaux d'horloge et met fin au transfert

Esclave : – Unité adressée par un maître

Multi-maître : – Possibilité pour plusieurs maîtres de tenter de prendre le contrôle du bus en même temps, sans altérer le message

Arbitrage : – Procédure permettant de résoudre les conflits d'accès des maîtres au bus et d'éviter l'altération du message. Le contrôle du bus n'est accordé qu'à un maître à la fois.

Le bus I²C remplit toutes les conditions décrites plus haut :

- C'est un bus série bifilaire, comprenant une ligne de données et une ligne d'horloge.
- Il permet le transfert bidirectionnel des données.
- C'est un véritable bus multi-maître, ce qui signifie qu'il est possible de connecter au bus plusieurs unités capables d'en prendre le contrôle. Chaque maître génère sa propre horloge.
- Chaque circuit intégré a une adresse unique sur 7 bits et peut fonctionner comme émetteur ou comme récepteur.
- Chaque circuit peut être considéré comme maître ou comme esclave selon le cas lors d'un transfert de données.
- Une procédure d'arbitrage évite la perte ou l'altération des données lorsque plusieurs maîtres se disputent le contrôle du bus.
- Le premier octet d'un transfert contient l'adresse de l'esclave sur 7 bits. Le bit de poids le plus faible de cet octet est un bit de direction.
- Chaque octet transféré est acquitté par le récepteur.
- La mise en œuvre de l'interface esclave est simple et économique.
- Le protocole est standardisé.

Une autre caractéristique du bus I²C est que chaque maître peut utiliser le bus à sa propre vitesse tant qu'elle ne dépasse pas 100 kbits/sec. Par conséquent, les transferts de données sont asynchrones et le maître ne génère son horloge que tant qu'il a le contrôle du bus. Si plusieurs maîtres tentent d'accéder simultanément au bus, un signal d'horloge système résulte des horloges des maîtres actifs.

Les niveaux d'entrée du bus I²C ont été conçus de façon à pouvoir protéger les circuits contre les phénomènes transitoires sur la ligne. Par exemple, dans un récepteur de TV, des résistances série de 300 Ω maximum peuvent être utilisées pour protéger les C.I. contre les pointes de surtension sur les lignes de données et d'horloge, provoquées par exemple par un amorçage dans le tube image.

Outre la limite due à l'adressage sur 7 bits, le nombre maximal d'unités pouvant être commandées par le bus n'est limité que par la charge capacitive maximale du bus, soit 400 pF. La longueur totale du bus peut atteindre 3 à 4 mètres.

Caractéristiques générales

Les lignes de données série (SDA) et d'horloge série (SCL) sont toutes deux bidirectionnelles. Elles sont connectées chacune à une tension d'alimentation positive par une résistance de rappel (voir Fig. 1). Quand le bus est libre, les deux lignes sont au niveau HAUT. Les étages de sortie des circuits connectés au bus doivent avoir un drain ou un collecteur ouvert pour remplir la fonction « ET-câblé ». Le nombre de circuits pouvant être connectés au bus, et la longueur du bus, sont limités par la charge capacitive maximale du bus : 400 pF.

En raison de la diversité des technologies des dispositifs (CMOS, NMOS, bipolaires) qui peuvent être connectés au bus I²C, les niveaux du '0' (BAS) et du '1' (HAUT) logiques ne sont pas fixes, mais dépendent de la tension d'alimentation des C.I. (voir SPECIFICATIONS ELECTRIQUES DES ENTREES ET DES SORTIES DES DISPOSITIFS I²C). Une impulsion d'horloge est générée pour chaque bit de données transféré. La vitesse maximale de transmission de données est de 100 kbits/sec.

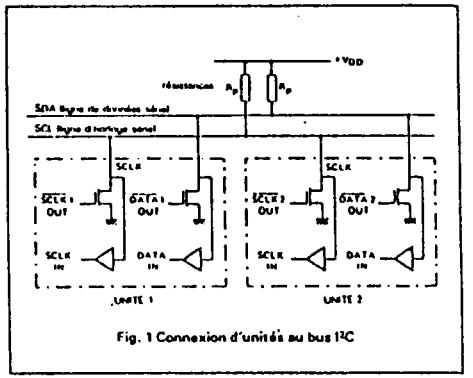


Fig. 1 Connexion d'unités au bus I²C

Validité des données

Comme l'indique la Fig. 2, les données transmises sur la ligne SDA doivent être stables pendant que l'horloge est au niveau HAUT. Les changements d'état de la ligne de données ne sont autorisés que lorsque le signal d'horloge sur la ligne SCL est au niveau BAS (Fig. 2).

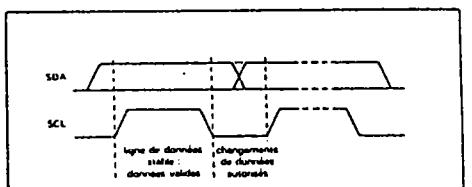


Fig. 2 Validité des données sur le bus I²C

Conditions de départ et d'arrêt

Comme le montre la Fig. 3, un front descendant sur la ligne SDA quand SCL est au niveau HAUT indique une condition de départ. Un front montant sur la ligne SDA quand SCL est au niveau HAUT indique une condition d'arrêt. Les conditions de départ et d'arrêt sont toujours générées par le maître. Le bus est

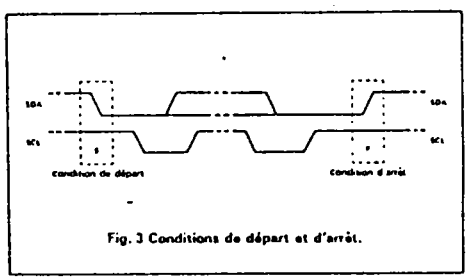


Fig. 3 Conditions de départ et d'arrêt.

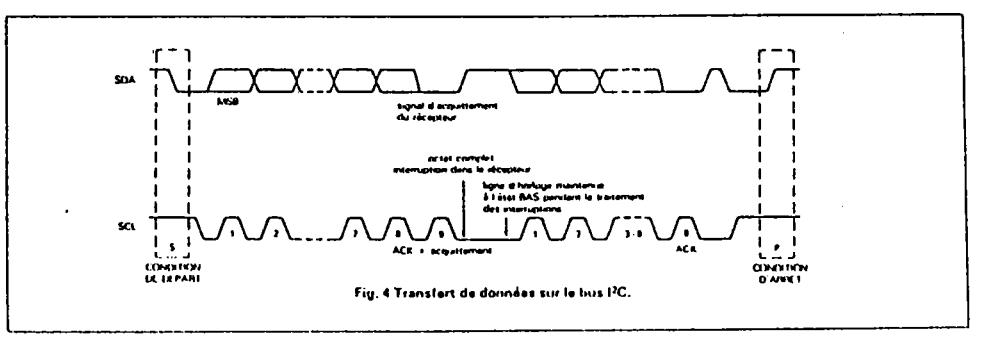


Fig. 4 Transfert de données sur le bus I²C.

considéré occupé après l'apparition de la condition de départ. Il est considéré libéré au bout d'un certain laps de temps après l'apparition de la condition d'arrêt. Ce point sera repris plus loin en détail.

Les unités connectées au bus peuvent détecter facilement les conditions de départ et d'arrêt si elles possèdent l'interface nécessaire. Cependant, les micro-ordinateurs, qui ne possèdent pas ce type d'interface, doivent échantillonner la ligne SDA au moins deux fois par période d'horloge pour détecter les changements d'état.

Transfert de données

Format des mots. Chaque mot transféré sur la ligne SDA doit contenir 8 bits. Le nombre d'octets qui peuvent être transmis par transfert est illimité. Chaque octet doit être suivi d'un bit d'acquittement. Comme le montre la Fig. 4, le bit de plus fort poids (MSB) est transmis en premier. Si le récepteur ne peut pas recevoir un autre octet de données complet tant qu'il n'a pas rempli une certaine fonction – par exemple, le traitement d'une interruption interne –, il peut maintenir la ligne d'horloge SCL à l'état BAS pour mettre l'émetteur en attente. Le transfert reprend dès que le récepteur est prêt à recevoir un autre octet et libère la ligne d'horloge SCL.

Acquittement. L'impulsion d'horloge correspondant à l'acquittement est générée par le maître qui a le contrôle du bus. Pendant cette impulsion, l'émetteur libère la ligne SDA (niveau HAUT), tandis que le récepteur remet la ligne SDA au niveau BAS où elle reste stable pendant la durée de l'impulsion (Fig. 5). Bien sûr, les temps d'établissement et de maintien doivent être pris en compte. Un récepteur qui a été adressé est normalement obligé de générer un acquittement après chaque octet reçu. Lorsqu'un récepteur esclave n'accuse pas réception de son adresse, par exemple, lorsqu'il ne peut pas recevoir parce qu'il est occupé à exécuter une fonction temps réel, il doit maintenir la ligne de données au niveau HAUT. Le maître peut alors générer une condition d'arrêt pour abandonner le transfert.

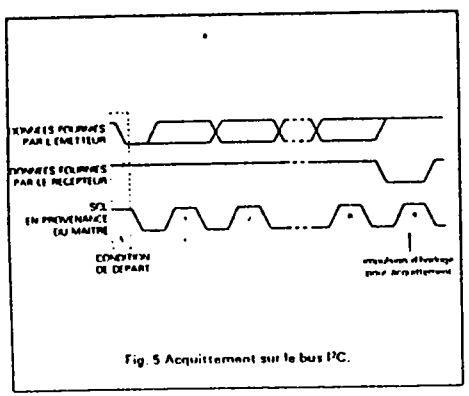


Fig. 5 Acquittement sur le bus I²C.

Si un récepteur esclave accuse réception de son adresse d'esclave, mais qu'ultérieurement pendant le transfert il ne peut plus recevoir d'autres octets de données, le maître doit à nouveau arrêter le transfert. Dans ce cas, l'esclave n'ayant pas accusé réception d'un octet de données, la ligne de données reste au niveau HAUT et le maître génère une condition d'arrêt.

Si un récepteur maître reçoit des données, il doit signaler une fin de données à l'émetteur esclave en n'acquittant pas le dernier octet de l'émetteur esclave. Ce dernier doit alors libérer la ligne de données pour permettre au récepteur maître de générer la condition d'arrêt.

Génération d'horloge et arbitrage

Synchronisation. Tous les maîtres génèrent leur propre horloge sur la ligne SCL pendant le transfert des données. Les données ne sont valides que pendant que l'horloge est au niveau HAUT. Il est donc nécessaire de définir une horloge pour permettre l'exécution d'une procédure d'arbitrage bit par bit. La synchronisation de l'horloge est réalisée par la connexion « ET-câblé » des dispositifs à la ligne SCL. Comme le montre la Fig. 6, lorsqu'un front descendant est présent sur la ligne SCL, tous les dispositifs candidats décomptent leurs niveaux BAS et la ligne SCL est maintenue au niveau BAS jusqu'à ce que le dispositif ayant le niveau BAS le plus long ait fini de compter. Ceux qui ont des niveaux BAS plus courts restent dans un état d'attente HAUT. Lorsque tous les dispositifs concernés ont fini de décompter leurs niveaux BAS, la ligne d'horloge est libérée et passe au niveau HAUT. Il n'y a alors aucune différence entre l'état des horloges des dispositifs et celui de la ligne SCL, et tous les dispositifs se mettent à compter la durée de leurs niveaux HAUT. Le premier dispositif qui ait fini de compter, met la ligne SCL au niveau BAS. C'est ainsi qu'est générée une horloge SCL synchronisée, dont la durée du niveau BAS est déterminée par le dispositif ayant la période d'horloge niveau BAS la plus longue, et le niveau HAUT par le dispositif ayant la période d'horloge niveau HAUT la plus courte.

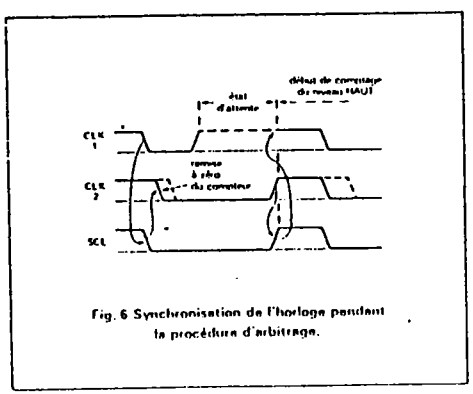


Fig. 6 Synchronisation de l'horloge pendant la procédure d'arbitrage.

Arbitrage. Le système d'arbitrage permet à un maître qui transmet un niveau HAUT sur la ligne SDA tandis qu'un autre maître transmet un niveau BAS, de mettre son étage de sortie de données hors service car le niveau présent sur le bus ne correspond pas à son propre niveau. L'arbitrage peut s'étendre sur plusieurs bits. La première phase de l'arbitrage consiste à comparer les bits d'adresse. Si les maîtres adressent la même unité, les bits de données sont comparés. Comme ce sont les informations d'adresse et de données qui servent à l'arbitrage, ceci ne retarde pas le transfert des informations.

Un maître qui perd l'arbitrage peut générer des impulsions d'horloge jusqu'à la fin de l'octet dans lequel il a perdu l'arbitrage. Si un maître perd l'arbitrage pendant la procédure d'adressage, il est possible que ce soit lui que le maître gagnant tente d'adresser. Le perdant doit donc immédiatement passer en mode récepteur esclave.

La Fig. 7 montre la procédure d'arbitrage pour deux maîtres. En fait, le nombre de maîtres en compétition peut être quelconque. Dès que le maître générant DATA1 génère un niveau HAUT interne pendant que les données sur la ligne SDA sont au niveau BAS, il met sa sortie hors service de sorte que le maître générant DATA2 gagne et sa sortie continue à être transférée sur la ligne SDA. Comme l'arbitrage est strictement basé sur les adresses et les données transmises par les maîtres concurrents, il n'y a pas de maître central, ni aucun ordre de priorité sur le bus.

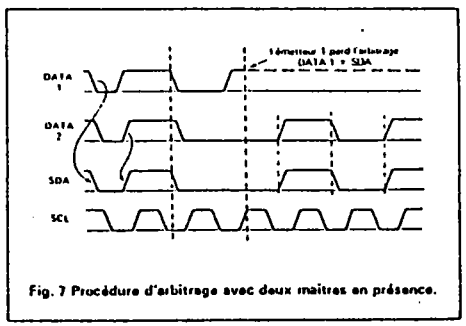


Fig. 7 Procédure d'arbitrage avec deux maîtres en présence.

Utilisation du mécanisme de synchronisation de l'horloge comme protocole d'établissement d'une liaison

En dehors de la procédure d'arbitrage, le mécanisme de synchronisation de l'horloge permet aux récepteurs de recevoir des transmissions de données rapides, soit au niveau de l'octet, soit au niveau du bit.

Au niveau de l'octet, une unité peut être capable de recevoir des octets de données à une cadence accélérée, mais demande plus de temps pour stocker l'octet reçu ou préparer la transmission d'un autre octet. Les unités esclaves peuvent alors maintenir la ligne SCL au niveau BAS, après réception et acquittement d'un octet, pour forcer le maître à se mettre en attente jusqu'à ce qu'elles soient prêtes pour la transmission de l'octet suivant dans une procédure comparable à celle du protocole d'établissement d'une liaison.

Au niveau du bit, une unité telle qu'un micro-ordinateur n'ayant pas d'interface I²C matérielle implantée sur la puce peut ralentir l'horloge du bus en prolongeant le niveau BAS de chaque période d'horloge. La vitesse du maître est donc adaptée automatiquement à la vitesse interne de fonctionnement de cette unité.

Formats

Le format des transferts de données est représenté à la Fig. 8. Après la condition de départ, l'adresse (7 bits) de l'esclave est transmise, suivie d'un bit de direction des données (L/È) ; un '0' indique ECRITURE, un '1' LECTURE. Un transfert de données se termine toujours par une condition d'arrêt, générée par le maître. Cependant, si un maître désire continuer à communiquer sur le bus, il peut générer une autre condition de départ et adresser un autre esclave sans générer au préalable une condition d'arrêt. Différentes combinaisons de formats lecture/écriture sont alors possibles à l'intérieur de ce transfert. Les différents formats de transfert de données sont :

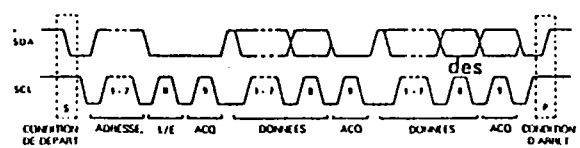
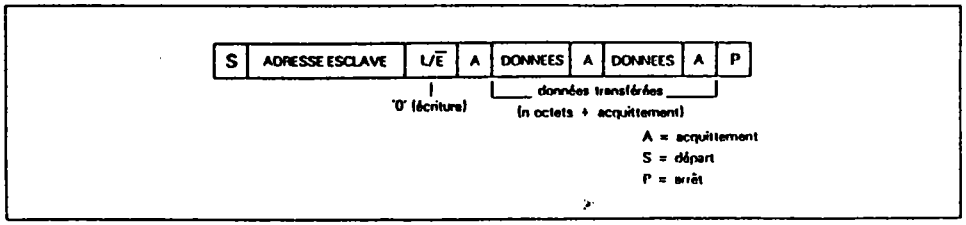
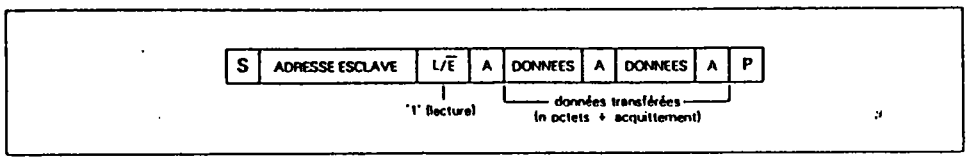


Fig. 8 Transfert de données complet.

(a) L'émetteur maître transmet des données à un récepteur esclave. La direction n'est pas modifiée.

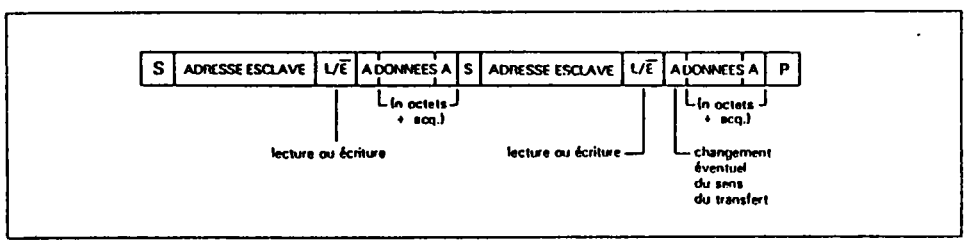


(b) Le maître lit l'esclave aussitôt après le premier octet.



Au moment du premier acquittement, l'émetteur-maître devient récepteur-maître et le récepteur-esclave devient émetteur-esclave. Cet acquittement continue à être généré par l'esclave. La condition d'arrêt est générée par le maître.

(c) Combinaison de formats.



Lors d'un changement de sens à l'intérieur d'un transfert, la condition de départ et l'adresse esclave sont répétées, mais avec le bit L/Ê inversé.

Notes :

1) Des combinaisons de formats peuvent être utilisées, par exemple, pour commander une mémoire série. Pendant le premier octet de données, l'adresse mémoire interne doit être écrite. Après la répétition de la condition de départ, les données peuvent être transmises.
2) Toutes les décisions concernant l'incrémentation ou la décrémentation automatique des emplacements mémoire préalablement adressés sont prises par le concepteur du dispositif.
3) Chaque octet est suivi d'un acquittement (A) comme l'indique le schéma de la séquence.
4) Les dispositifs I²C doivent remettre la logique du bus à zéro à la réception d'une condition de départ en prévision de l'émission d'une adresse d'esclave.

Adressage

Dans la procédure d'adressage du bus I²C, le premier octet après la génération de la condition de départ détermine le choix de l'esclave par le maître, sauf pour l'adresse « d'appel général » qui peut adresser toutes les unités. Lorsque cette adresse est utilisée, toutes les unités doivent en principe répondre par un acquittement (bien que certaines puissent ignorer cette adresse). Le second octet de l'appel général définit l'action à effectuer.

Définition bits du premier octet. Les sept premiers bits de cet octet constituent l'adresse de l'esclave (Fig. 9). Le huitième bit (LSB ou bit de poids le plus faible) détermine la direction du message. Un '0' signifie que le maître écrit des informations dans un esclave sélectionné ; un '1' signifie que le maître lit des informations d'un esclave.

Au moment de la transmission de l'adresse, chaque unité connectée au bus compare les 7 premiers bits après la condition de départ à sa propre adresse. S'il y a concordance, l'unité se considère adressée par le maître comme récepteur ou émetteur esclave, en fonction du contenu du bit L/Ê.

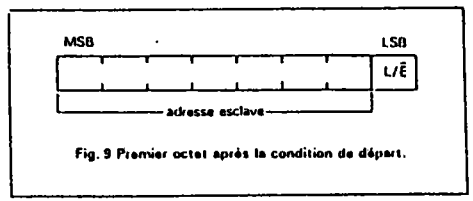


Fig. 9 Premier octet après la condition de départ.

Comme plusieurs circuits identiques peuvent être utilisés dans un système, l'adresse de l'esclave peut être composée d'une partie fixe et d'une partie programmable. Le nombre de bits d'adresse programmables dépend du nombre de broches disponibles. Par exemple, si un dispositif a 4 bits d'adresse fixes et 3 bits programmables, il est possible de connecter jusqu'à 8 unités identiques sur le même bus.

Adresse d'appel général. L'adresse d'appel général permet d'adresser toutes les unités connectées au bus I²C. Cependant, si une unité n'a pas besoin des données fournies dans la structure de l'appel général, elle peut ignorer cette adresse. Si, par contre, elle a besoin de ces données, elle se comporte comme un récepteur esclave. Le deuxième octet et les octets suivants sont acquittés par chaque récepteur esclave capable de traiter ces données. Un esclave qui n'est pas capable de traiter un de ces octets doit l'ignorer en ne l'acquittant pas. La signification de l'appel général est toujours indiquée par le deuxième octet (Fig. 10).

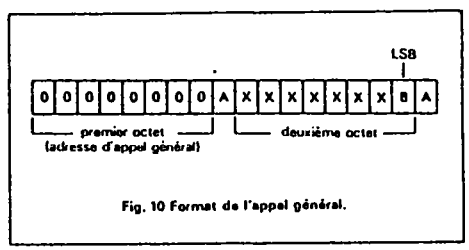


Fig. 10 Format de l'appel général.

SPECIFICATIONS ELECTRIQUES DES ENTREES ET DES SORTIES DES DISPOSITIFS I²C

Le bus I²C permet la communication entre dispositifs réalisés dans différentes technologies et utilisant éventuellement des tensions d'alimentation différentes. Pour les dispositifs fonctionnant sur une tension d'alimentation de 5 V ± 10%, les niveaux d'entrée sont :

V_{ILmax} = 1,5 V (tension d'entrée maxi. niveau BAS)

V_{IHmin} = 3,0 V (tension d'entrée mini. niveau HAUT)

Les dispositifs fonctionnant sur une tension d'alimentation différente de 5 V (par ex. bipolaire) doivent également avoir ces niveaux d'entrée.

Pour les dispositifs fonctionnant sur une large plage de tension d'alimentation (par ex., CMOS), les niveaux d'entrée sont :

V_{ILmax} = 0,3 V_{DD} (tension d'entrée maxi. niveau BAS)

V_{IHmin} = 0,7 V_{DD} (tension d'entrée mini. niveau HAUT)

Pour les deux groupes de dispositifs, la tension de sortie maximale, niveau BAS est :

V_{OLmax} = 0,4 V (tension de sortie maximale niveau BAS pour un courant absorbé de 3 mA)

Les dispositifs ayant des niveaux d'entrée indépendants de l'alimentation peuvent avoir leur propre alimentation 5 V ± 10%. Les résistances de rappel peuvent être connectées à n'importe quelle alimentation, comme indiqué à la Fig. 11. Cependant, les dispositifs ayant des niveaux d'entrée référencés à V_{DD} doivent avoir une ligne d'alimentation commune à laquelle sont également connectées les résistances de rappel (Fig. 12).

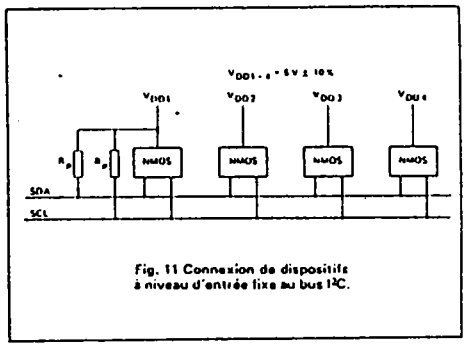


Fig. 11 Connexion de dispositifs à niveau d'entrée fixe au bus I²C.

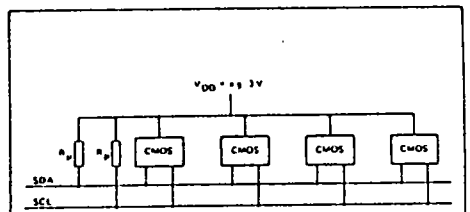


Fig. 12 Connexion de dispositifs ayant une large plage de tension d'alimentation au bus I²C.

Lorsque des dispositifs à niveaux d'entrée fixes sont mélangés à des dispositifs à niveaux référencés à V_{DD}, ces derniers doivent être connectés à une ligne d'alimentation commune de 5 V ± 10% à laquelle sont connectées les résistances de rappel (Fig. 13).

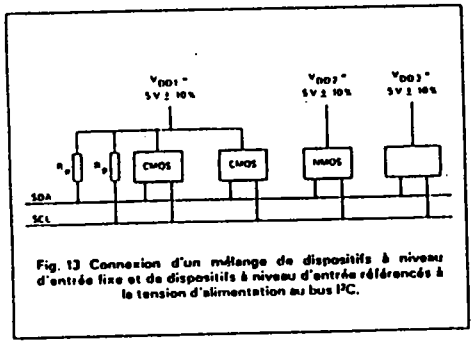


Fig. 13 Connexion d'un mélange de dispositifs à niveau d'entrée fixe et de dispositifs à niveau d'entrée référencés à la tension d'alimentation au bus I²C.

Les niveaux d'entrée sont définis de telle manière que :

- la marge de bruit au niveau BAS est égale à 0,1 V_{DD}
- la marge de bruit au niveau HAUT est égale à 0,2 V_{DD}
- la protection contre les surtensions sur les lignes SDA et SCL (par exemple par suite d'un amorçage du tube image de TV) peut être assurée par des résistances série (R_S) de 300 Ω maximum (voir Fig. 14).

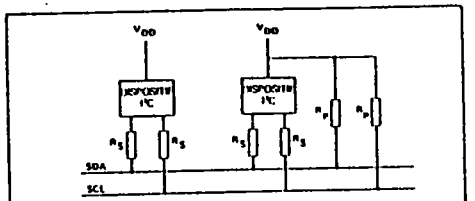


Fig. 14 Résistances série protégeant les circuits intégrés contre les surtensions transitoires.

La charge capacitive maximale du bus est de 400 pF par ligne. Ceci inclut la capacité du fil lui-même et celle des broches qui y sont connectées.

BASE DE TEMPS

L'horloge du bus I²C a une période minimale de 4,7 µs au niveau BAS et de 4 µs au niveau HAUT. Les maîtres peuvent générer une horloge bus à une fréquence de 100 kHz maximum.

Tous les dispositifs connectés au bus doivent être capables de suivre des transferts jusqu'à une fréquence de 100 kHz, soit en émettant ou en recevant directement des données à cette fréquence, soit en appliquant la procédure de synchronisation de l'horloge qui met le maître en attente et étend les périodes de l'horloge au niveau BAS. Bien-entendu, dans ce dernier cas, la fréquence est réduite.

130, AVENUE LEDRU-ROLLIN - 75540 PARIS CEDEX 11 - TÉL. (1) 43.38.80.00 - TÉLEX : 680.495 F

RTC-COMPELEC - S.A. AU CAPITAL DE 327 156 700 FRANCS - R.C.S. PARIS B 672 042 470 - SIÈGE SOCIAL : 130, AVENUE LEDRU-ROLLIN, 75540 PARIS CEDEX 11 - ADRESSE TÉLÉGRAPHIQUE : TUBELEC-PARIS 012 - SIRET 672 042 470 00084 - APE 2916 - C.C.P. PARIS 11 773-32 - AUTRES ÉTABLISSEMENTS A DRIVE-LA-GAILLARDE - CAEN - DREUX - EVREUX - FONTENAY-AUX-ROSES - LIMEIL-BRÉVANNES - SURESNES

UNIVERSAL LCD DRIVER FOR LOW MULTIPLEX RATES

GENERAL DESCRIPTION

The PCF8566 is a peripheral device which interfaces to almost any liquid crystal display (LCD) having low multiplex rates. It generates the drive signals for any static or multiplexed LCD containing up to four backplanes and up to 24 segments and can easily be cascaded for larger LCD applications. The PCF8566 is compatible with most microprocessors/microcontrollers and communicates via a two-line bidirectional bus (I²C). Communication overheads are minimized by a display RAM with auto-incremented addressing, by hardware subaddressing and by display memory switching (static and duplex drive modes).

Features

- Single-chip LCD controller/driver
- Selectable backplane drive configuration: static or 2/3/4 backplane multiplexing
- Selectable display bias configuration: static, 1/2 or 1/3
- Internal LCD bias generation with voltage-follower buffers
- 24 segment drives: up to twelve 8-segment numeric characters; up to six 15-segment alphanumeric characters; or any graphics of up to 96 elements
- 24 x 4-bit RAM for display data storage
- Auto-incremented display data loading across device subaddress boundaries
- Display memory bank switching in static and duplex drive modes
- Versatile blinking modes
- LCD and logic supplies may be separated
- 3 V to 6 V power supply range
- Low power consumption
- Power-saving mode for extremely low power consumption in battery-operated and telephone applications
- I²C bus interface
- TTL/CMOS compatible
- Compatible with any 4-bit, 8-bit or 16-bit microprocessors/microcontrollers
- May be cascaded for large LCD applications (up to 1536 segments possible)
- Cascadable with the 40 segment LCD driver PFC8576
- Optimized pinning for single plane wiring in both single and multiple PCF8566 applications
- Space-saving 40-lead plastic mini-pack (VSO-40; SOT-158A)
- No external components required (even in multiple device applications)
- Manufactured in silicon gate CMOS process

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specification defined by Philips.

PACKAGE OUTLINES

PCF8566T: 40-lead mini-pack; (VSO-40; SOT-158A).
PCF8566P: 40-lead DIL; plastic (SOT-129).

BOITIER TELECOMMANDE

Chain div. int I2C bus

Chain div. int RS232

JUMPER 1

JUMPER 2

RESEAU 10K

RESEAU 4K7

I2C BUS

Memory Mapping:

Total card: 64 from 22000 to 23FFF Hex

RS232: 22000 to 221FF Hex
BALD: 22200 to 223FF Hex
free: 22400 to 227FF Hex
I2C: 22800 to 229FF Hex
free: 22A00 to 23FFF Hex

Title	CARTE03
	ECAO PROJET
Size	Document Number
C	3
Date: February 7, 1995	Sheet

Memory Mappings:

Eprom System	00000 to 07FFF hex	CE1\
Ram System	08000 to 0FFFF hex	CE2\
Eprom User	10000 to 17FFF hex	CS1\ to CS4\
Ram User	18000 to 1FFFF hex	CE4\

EEPROM

VERS 68008(ADRESSES+DONNEES)

Generation du DTACK\

RAM 32K

EPROM 32K

Projet:	ECAD - CERLOR
Titre:	Schéma éléctrique Carte01
Date:	9 Février 1990