TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

Analyse et Conception de Circuits Electroniques Assistées par Ordinateur (ACCEAO) est un logiciel de simulation de circuits électroniques conçu sous l'environnement MS-DOS. C'est un logiciel pédagogique, destiné à être utilisé par des étudiants des lycées techniques ou universitaires suivant des cours d'*Électronique Analogique*.

En effet, par son utilisation, l'étudiant peut concrétiser les acquis de sa formation sur ordinateur. Le but du logiciel n'est pas de remplacer l'étudiant dans son travail mais de lui servir d'outil d'apprentissage et de recherche.

Devant l'évolution de l'informatique actuellement, aussi bien dans le domaine matériel mais surtout au niveau logiciel, ACCEAO nécessite une maintenance adaptive vers le système d'exploitation Windows. Ce changement d'environnement permet entre autres une amélioration de l'interface et une meilleure maîtrise des événements de la souris ainsi que du clavier, dans un but de rendre l'utilisation du logiciel plus facile.

Les étapes de la migration et de la maintenance sont effectuées en utilisant l'outil de développement C++ Builder version 6.0. Le choix de cet outil s'explique par la nécessité de portabilité du programme ainsi créé dans toutes les versions de Windows : Windows 95 à Windows NT et le récent Windows Vista.

Ce présent mémoire s'intéresse particulièrement sur le sujet de la migration. Aussi sont évoquées les actions de maintenances perfectives. Dans cette étude, nous allons découvrir : dans le premier chapitre, quelques notions sur la programmation sous Windows. Dans le second chapitre, la description et l'historique du logiciel ACCEAO. La mise en place d'une nouvelle interface utilisateur ainsi que l'introduction de nouveaux composants sont détaillées dans le troisième chapitre. Un guide d'utilisation du logiciel ACCEAO version 5.0 est détaillé dans le dernier chapitre.

Chapitre 1 PROGRAMMATION SOUS WINDOWS

1.1. Les systèmes d'exploitation MS-DOS et Windows

L'apparition au grand public des micro-ordinateurs en 1978 avec l'Apple et l'IBM-PC, a débuté les différentes phases d'évolution des générations d'ordinateurs.

Les OS[1] des micro-ordinateurs ont suivi la même démarche avec, au début, les systèmes de monoprogrammation comme MS-DOS et MacOs pour évoluer vers les systèmes multitâches avec Windows et Linux.

MS-DOS est le premier système d'exploitation de PC[2] conçu par la société Microsoft au début des années 1980. Ce système fonctionne sur l'architecture 16 bits. Les limitations de MS-DOS portent sur plusieurs aspects. Tout d'abord, la taille maximale de la mémoire centrale qui peut être gérée est de 1 Mo. Le système est mono-utilisateur et mono-tâche (un seul programme peut s'exécuter en un instant donné). Il n'y a pas de protection mémoire ; une erreur d'exécution dans un programme peut donc bloquer le système. Les noms de fichiers sont limités à huit caractères, plus trois pour l'extension (caractérisation du type de fichier).

C'est au début des années 1990 qu'est apparue la première version de Windows. Contrairement au MS-DOS, ce nouveau système repose sur les plates-formes basées sur les principaux microprocesseurs équipant les PC du marché, aussi bien sur l'architecture 32 bits que sur l'architecture 64 bits toute récente.

Ainsi, Windows est parti d'un OS primitif et spécifique à un PC pour intégrer au cours du temps les fonctionnalités d'un OS de multiprogrammation (avec processus, mémoire virtuelle, multitâches, préemptif, etc. …). L'interface de communication est intégrée dans le cœur même du système. Le mode console (interface en ligne de commande genre MS-DOS ou ligne de commande Linux) est présent mais est très peu utilisé, les fonctionnalités de base du système étant assurées par des processus fenêtrés.

[1] Operating System : système d'exploitation
[2] Personnal Computer : micro-ordinateur

De ces changements, les développeurs de logiciels sont passés de l'écriture de programmes en mode console de MS-DOS à l'écriture de programmes pouvant interagir avec le système d'exploitation Windows.

1.2. Les langages de programmation

On attend d'un programme informatique :

- l'exactitude (réponse aux spécifications)
- l'extensibilité (aptitude à l'évolution)
- la réutilisabilité (utilisation de modules)
- la portabilité (support d'une autre implémentation)
- l'efficience (performance en termes de vitesse d'exécution et de consommation mémoire).

Pour répondre à ces exigences, la programmation algorithmique classique telle qu'on peut la connaître à travers des langages de programmation comme Pascal, C,… a cédé sa place à la programmation orientée objet.

a) Les langages structurés

Ces langages nécessitent la définition détaillée du code constituant du programme. Ils ont l'avantage d'être facile à manipuler car leur utilisation ne nécessite pas l'apprentissage de l'outil de programmation. Ces langages reposent sur le principe de la programmation structurée (algorithmes+structures de données). Parmi les plus connus, on distingue le langage C, FORTRAN.

b) Les langages orientés objets

Les langages orientés objets sont des langages adaptés à la programmation orientée objet, type de programmation où chaque programme est considéré comme un ensemble d'objets distincts, ces objets constituant eux-mêmes des ensembles de structures de données et de procédures intégrées [1]. Dans de tels langages, chaque objet appartient à une classe qui définit les structures de données et les procédures associées à cet objet.

Les langages orientés objets ont été développés pour faciliter l'écriture et améliorer la qualité des logiciels en termes de modularité. Un langage orienté objet sera livré avec

une bibliothèque de classes. Le développeur utilise ces classes pour mettre au point ses logiciels.

(i). **Illustration : le langage C++**

Le C++, langage orienté objet le plus utilisé a deux grands ancêtres.
D'abord, Simula, dont la première version (Simula I) a été conçue en 1967 par Dahl, Mayrhaug et Nygaard à Oslo. C'est le premier langage qui introduit les principaux concepts de la programmation objet.

Probablement parce qu'il était en avance sur son temps, il n'a pas connu à l'époque le succès qu'il aurait mérité, mais il a eu cependant une influence considérable sur l'évolution de la programmation objet.

Conçu d'abord à des fins de modélisation de systèmes physiques, en recherche nucléaire notamment, Simula I est devenu un langage spécialisé pour traiter des problèmes de simulation. Son nom fut changé en Simula en 1986. Comme son prédécesseur Simula I, Simula permet de traiter les problèmes de simulation. En particulier, un objet est considéré comme un programme actif autonome, pouvant communiquer et se synchroniser avec d'autres objets. C'est aussi un langage de programmation général, reprenant les constructions de la programmation modulaire introduites par Algol[3]. Il y ajoute les notions de *classe*, d'*héritage* et autorise le *masquage* des méthodes, ce qui en fait un véritable langage à objets.

Et évidemment, le langage C qui a été conçu en 1972 aux laboratoires *Bell Labs*. Il est un langage structuré et modulaire, dans la philosophie générale de la famille Algol. Mais c'est aussi un langage proche du système, qui a notamment permis l'écriture et le portage du système Unix. Par conséquent, la programmation orientée système s'effectue de manière aisée en C, et on peut en particulier accéder directement aux fonctionnalités du noyau Unix.

Le langage C possède un jeu très riche d'opérateurs, ce qui permet l'accès à la quasi-totalité des ressources de la machine [2]. On peut par exemple faire de l'adressage indirect ou utiliser des opérateurs d'incrémentation ou de décalage. Il est aussi possible d'implanter une variable dans un registre. En conséquence, on peut écrire des programmes

[3] ALGOrithmic Language : Langage algorithmique.

presque aussi efficaces qu'en langage d'assemblage, tout en programmant de manière structurée.

Le concepteur de C++, Bjarne Stroustrup, qui travaillait également aux *Bell Labs*, désirait ajouter au langage C les classes de Simula. Après plusieurs versions préliminaires, le langage a trouvé une première forme stable en 1983, et a très rapidement connu un vif succès dans le monde industriel. C++ peut être considéré comme un successeur de C. Tout en gardant les points forts de ce langage, il corrige certains points faibles et permet l'abstraction de données. De plus, il permet la programmation objet [1].

(ii). **Les outils de développement pour C++**

L'utilisation du C++ dans le développement d'application est actuellement facilitée par l'existence des logiciels spécifiés [3]. Parmi eux, on distingue les produits Borland tels que le Borland C++ et le C++ Builder. La différence principale entre les deux est que C++ Builder permet la conception d'une application Windows (interface fenêtrée). Or Borland C++ (au dessous de la version 4.0) n'offre qu'une application en mode console. Nous avons donc choisi C++ Builder comme logiciel de développement.

1.3. Introduction à C++ Builder 6

C++ Builder est un environnement de développement basé sur C++ proposé par Borland [3]. Fort du succès de Delphi, Borland a repris la philosophie, l'interface et la bibliothèque de composants visuels de ce dernier pour l'adapter depuis le langage Pascal Orienté Objet vers C++ ; répondant ainsi à une large faction de programmeurs peu enclins à l'utilisation du Pascal qu'ils jugent quelque peu dépassé.

Tout d'abord C++ est un outil RAD[4], c'est à dire tourné vers le développement rapide d'applications sous Windows. En un mot, C++ Builder permet de réaliser de façon très simple l'interface des applications et de relier aisément le code utilisateur aux événements Windows, quelle que soit leur origine (souris, clavier, événement système, etc). Cependant, il faut savoir que la technologie RAD ne s'applique qu'au squelette ou à l'interface d'une application. Bien entendu, toute la partie spécifique au projet reste à la charge du développeur.

Pour ce faire, C++ Builder repose sur un ensemble très complet de *composants*

[4] Rapid Application Development

visuels prêts à l'emploi. La quasi-totalité des contrôles de Windows (boutons, boîtes de saisies, listes déroulantes, menus et autres barres d'outils) y est représentée. Leurs caractéristiques sont éditables directement dans une fenêtre spéciale intitulée *éditeur d'objets*. L'autre volet de cette même fenêtre permet d'associer du code au contrôle sélectionné (Fig.1.1).

Il est possible d'ajouter à l'environnement de base, des composants fournis par des sociétés tierces et même d'en créer soi-même.

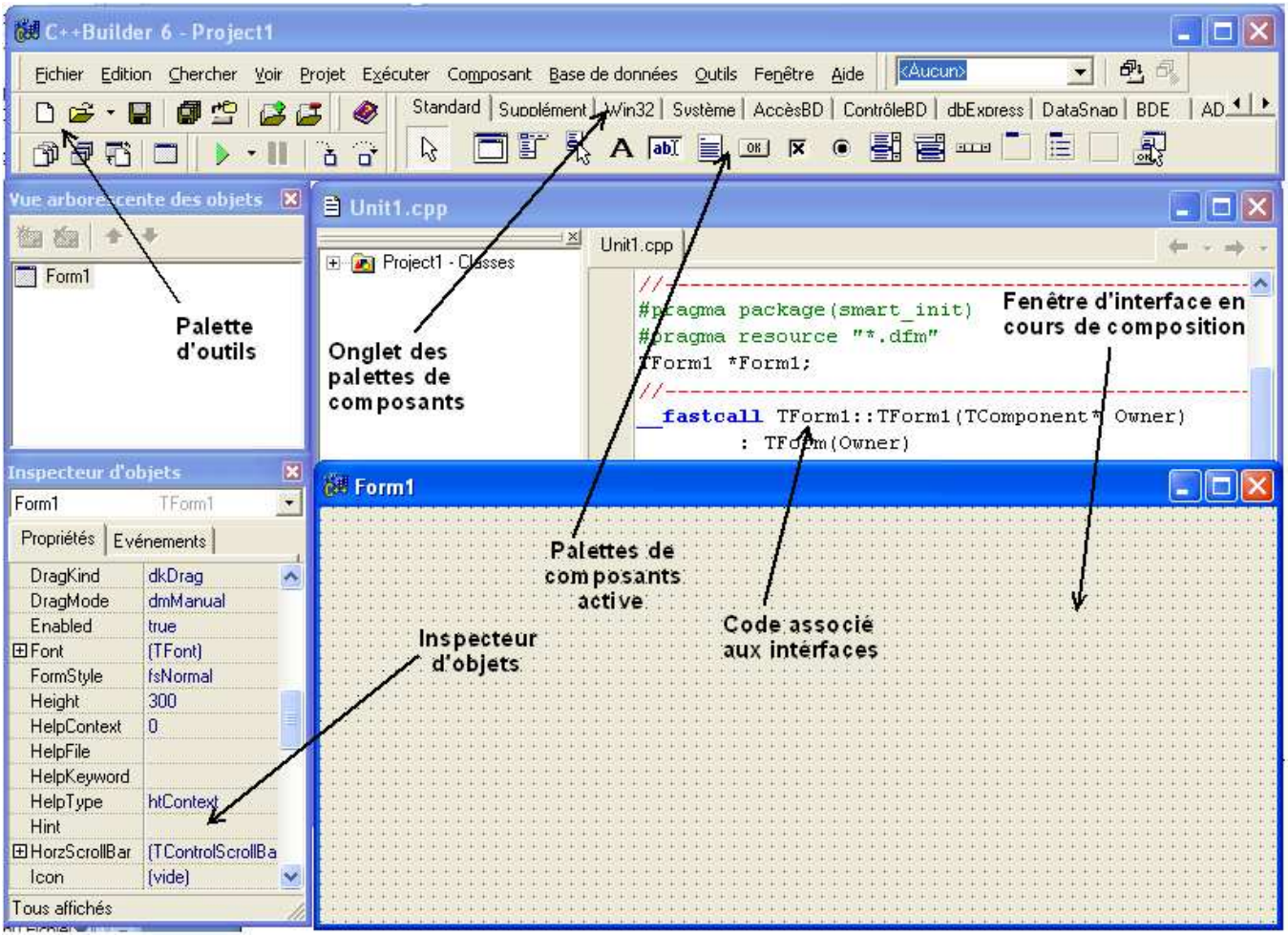


Figure 1. 1 : Interface de C++ Builder 6

Création d'une application simple sur C++ Builder :

C++ Builder permet de créer différents types de module très simplement en se laissant guider par des experts. Toutefois, il est possible de demander à créer une application simple en activant l'option **Nouvelle application** du menu **Fichier**.

Les éléments automatiquement créés sont les suivants : une fiche nommée **Form1** ainsi que les fichiers associés **Unit1.cpp** et **Unit1.h**. Notons au passage que la terminologie **Unit** est directement calquée sur celle chère à Delphi et que les fonctionnalités ainsi créées sont toujours ultérieurement, possibles à renommer.

Notons aussi, la nécessité de sauvegarder le projet juste après sa création : on évite ainsi la création des fichiers de compilation dans les répertoires par défaut. Cette opération est réalisée avec la commande **Sauvegarder le projet sous…** du menu **Fichier**. Le projet en lui même (fichier **.bpr**) est sauvegardé après les différents fichiers **.cpp** et **.h**.

1.4. Enregistrement des données sous Windows

Au cours de la programmation, un grand nombre de données est manipulé.

Avec l'outil C++ Builder, pour stocker les données, on peut utiliser principalement les fichiers et un tableau particulier dénommé : **vector**.

a) Les fichiers

Un fichier est une collection de données de même nature. Il est enregistré sur une entité mémoire de type mémoire de masse comme les disques durs, les disquettes,…

(i). Format d'un fichier

Le format représente la façon dont les données contenues dans le fichier sont stockées. Le format peut être de type texte (ASCII) ou binaire.

Dans un fichier texte, chaque caractère est représenté par un octet, tandis que dans un fichier binaire un entier est représenté par quatre octets.

On enregistre généralement les fichiers spécifiques d'une application sous forme binaire.

(ii). **Nomenclature**

Le nom d'un fichier est composé de deux parties séparées par un point. La première partie sert à accueillir le nom propre du fichier ; celle-ci est limitée à huit caractères sous MS-DOS. La seconde partie est l'extension qui est généralement en trois caractères.

b) Les vecteurs

Pour pallier les défauts inhérents à la rigidité des tableaux de taille fixe, la librairie (générique) standard de C++ fournit un type de donnée dénommée **vector** (*vecteur*), offrant au programmeur un moyen très efficace pour construire des structures de données permettant de représenter des **tableaux de tailles variables** (ou *tableaux dynamiques*). La taille de ces «tableaux» n'est pas obligatoirement prédéfinie, et peut donc varier en cours d'utilisation.

Pour pouvoir utiliser ces *vecteurs* dans un programme, il faut, comme dans le cas des entrées-sorties, importer les prototypes et définitions contenus dans la librairie, au moyen de la directive d'inclusion :

```
#include <vector>.
```

En opposition avec les fichiers, **vector** se situe dans la mémoire vive ou RAM[5].

Exemples de commandes sur vector :

- **Déclaration** : **vector**<*type de données*> *nom_du_vecteur* ;
- **Effacement** : *nom_du_vecteur* **.Clear**() ;
- **Nombre d'élements** : *nom_du_vecteur* **.size** () ;
- **Detection d'un élement sur le rang *i*** : *nom_du_vecteur* **.at**(i) ;
- **Insertion d'un element à la fin**:

nom_du_vecteur **.insert** (*nom_du_vecteur* **.end** (), *donnée_à_inserer*) ;

[5] Random Acces Memory

Donc pour enregistrer des données permanentes, on doit utiliser les fichiers. Mais pour les données temporaires, afin d'avoir un gain en mémoire et en vitesse, on emploie un vecteur au lieu d'un fichier.

Chapitre 2 LE LOGICIEL ACCEAO

Analyse et Conception de Circuits Electroniques Assistées par Ordinateur (ACCEAO) est un logiciel pédagogique conçu et développé à l'Ecole Supérieure Polytechnique d'Antananarivo, dans le Département Electronique. Il n'y a donc pas de version professionnelle de ACCEAO. C'est par son utilisation que les étudiants peuvent vérifier ce qu'ils ont acquis dans leurs cours d'Electronique, vérifier les résultats des exercices qu'ils ont traités, et essayer les circuits qu'ils ont conçus, avant même de passer aux essais sur breadboard.

ACCEAO dispose d'une interface utilisateur conviviale pour la saisie d'un schéma électronique. En effet, il suffit à l'utilisateur de dessiner le schéma à l'aide des outils qui lui sont procurés, et le logiciel se charge de le traduire en un langage qui lui est intelligible. Cette traduction se présente sous forme d'un fichier, donc il est facile pour le programme d'effectuer des sauvegardes sur disque et de charger des données enregistrées. Les références [4] et [5] sont des exemples d'anciennes versions de ACCEAO.

2.1. Historique

Pour satisfaire graduellement les critères de qualité mentionnés plus haut, le logiciel *ACCEAO* a subi des améliorations et des extensions.

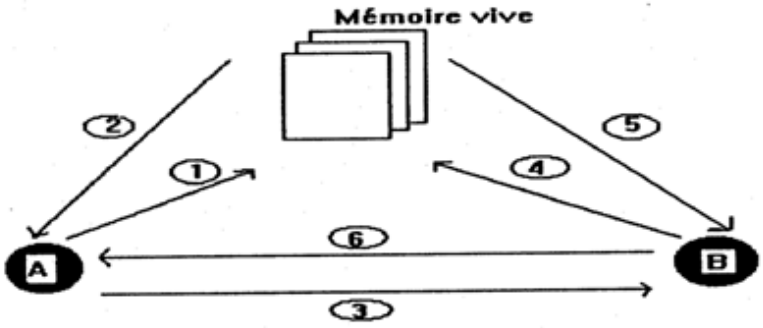
La première version de *ACCEAO* à être dotée d'une interface utilisateur graphique a été la **version 3.0**, dont la conception a été basée essentiellement sur le concept de la *classe-relation*. Ce concept a permis une conception orientée objet ainsi qu'une bonne qualité concernant la réutilisabilité et l'extensibilité. Mais un problème de gestion de mémoire est apparu.

L'utilisation de la *classe* accélère considérablement la vitesse de traitement du fait que le partage d'un même *champ de données* par plusieurs *modules* évite le transfert de données. Comme l'utilisation de *classe*s se fait par des allocations statiques des objets se référant à ces *classe*s, la taille exacte de chaque structure de données ne peut être connue que juste avant le traitement. De ce fait, il se peut que le nombre d'*objets* réservés soit insuffisant ou soit trop grand. Dans un cas comme dans l'autre, le résultat est un problème

de mémoire (insuffisance ou gaspillage). Et comme la réalisation était faite essentiellement avec des *pointeurs de classe,* la conséquence est la difficulté réelle de gestion de mémoire.

La solution trouvée a été de décomposer le système en deux sous-ensembles différents : la *manipulation des données (A*) et le *traitement des données (B*). C'est une alternative de l'utilisation des *classe*s où les fonctions membres partagent les *données* membres. Le principal avantage de cette décomposition est que *(A)* et *(B)* n'occupent pas simultanément la mémoire. Le mécanisme d'occupation de mémoire peut se résumer par le schéma de la Fig.2.1. La solution choisie pour établir l'interface entre ces deux parties est l'utilisation des fichiers.

- *Avantage de l'utilisation de fichiers* : l'emploi de fichiers comme stockage de données présente beaucoup d'avantages vis à vis des pointeurs. Car un fichier n'est chargé en mémoire qu'après une requête et il est immédiatement rangé en mémoire de masse dès que l'on ne l'utilise plus. La mémoire vive n'est donc utilisée que temporairement pour la remise à jour des fichiers contenant les résultats du traitement.



(1) A se charge en mémoire et en est le maître absolu
(2) A se décharge
(3) A transmet un message à B de la fin de son exécution
(4) B se charge en mémoire
(5) B se décharge de la mémoire
(6) B redonne la main à A

Figure 2. 1 : Mécanisme d'occupation de mémoire entre A et B

- *Inconvénient de l'utilisation de fichier*s : la réponse de la mémoire de masse est beaucoup plus lente que celle de la mémoire vive et ceci a une conséquence sur la vitesse de traitement.

Cette méthode est donc plus adaptée à des systèmes dont la quantité de mémoires importe plus que le temps d'exécution. Comme dans le cas de ACCEAO, étant donné qu'il est conçu pour la pédagogie (où il est important de trouver une solution optimum avec un minimum de moyen), et est destiné à être utilisé sur des machines juste au dessus de bas de gamme.

Fichier de configuration :

Pour assurer la *portabilité* du logiciel, *ACCEAO* est muni d'un fichier de configuration *ACCEAO.INI*, à partir de la **version 4.0**. En effet, un logiciel écrit en C++ et travaillant en mode graphique nécessite des fichiers externes pour pouvoir tourner correctement. Le nom du répertoire contenant ces fichiers externes doit être connu par le programme. Et ceci se réalise en utilisant le fichier *ACCEAO.INI.*

Base de données des composants :

ACCEAO dispose d'une base de données des composants actifs (diodes, BJT, FET) que l'on peut enrichir au cours des temps et mettre à jour. Après chaque ajout *ACCEAO* trie automatiquement les composants suivant leurs noms, par ordre alphabétique. Cette base de données renferme les paramètres des composants les plus fréquemment utilisés, et il suffit de le consulter lorsqu'on utilise un composant. Avec un clic gauche de la souris, la saisie des paramètres est dirigée vers le fichier contenant les données, et on n'a qu'à choisir un composant parmi d'autres.

Rubrique ***Help :***

Cette rubrique assiste l'utilisateur pendant la session. L'aide peut être *appelée en ligne.* Les textes explicatifs ne sont pas écrits au sein des codes sources du programme, ils sont enregistrés dans un fichier externe *ACCEAO.HLP*. Un fichier d'index *ACCEAO.IDX*

est prévu pour faciliter l'accès à une rubrique. L'affichage se fait comme suit : à chaque rubrique est associé un titre (unique), et si le programme est sollicité pour afficher une rubrique, il consulte le fichier *ACCEAO.IDX* pour déterminer la position du texte dans *ACCEAO.HLP*, ainsi que la taille de l'information correspondante, en octets (Fig.2.2).

L'intérêt de cette séparation des informations est la possibilité de mise à jour sans avoir à recompiler le programme. Ce qui a l'avantage de ne pas "grossir" *ACCEAO.EXE*, étant donné que pour tourner sous DOS, chaque programme exécutable ne doit pas dépasser environ 570 ko.

Pour anticiper un basculement vers le système d'exploitation Windows, une version de ***Help*** a été réalisée dans la version 4.0 à l'aide du programme *Microsoft ® Help Compiler Version 3.10.445*. Ses avantages sont l'exploitation de l'interface graphique de Windows (couleurs et polices de caractères) et surtout les liens hypertextes qui permettent de "naviguer" d'une rubrique à une autre sans devoir passer par la liste des rubriques. Mais il est important de souligner ici que *ACCEAO* ne pourra l'utiliser en ligne qu'avec une version sous Windows.

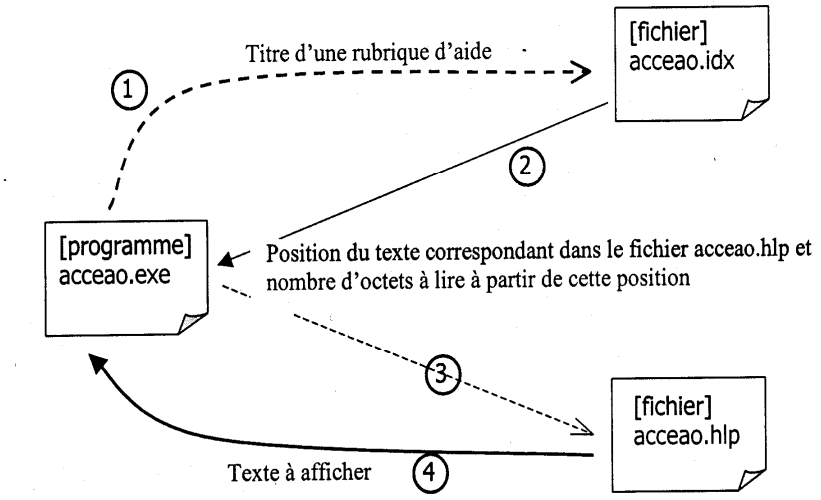


Figure 2. 2 : Structure de la rubrique ***Help***

Capture d'écran :

Comme *ACCEAO* ne dispose pas d'une commande qui permettrait d'imprimer le schéma d'un circuit analysé ou les résultats obtenus sur du support papier, on peut utiliser des programmes comme *MSPAINT.EXE* (pour les ordinateurs dotés du système d'exploitation *WINDOWS*) pour le faire. Pour la capture d'écran, on appuie (en deux reprises) sur les touches *Alt* Gr et *$*. *ACCEAO* demandera alors à l'utilisateur l'endroit où stocker le fichier, qui est un fichier *BITMAP* (***.bmp**), et effectue l'opération.

Implantation de la souris :

Une des améliorations les plus importantes est l'implantation de la souris dans *ACCEAO,* réalisée avec succès dans la **version 4.3**. L'introduction du fichier **souris.cpp** et la correction de certaines erreurs ont conduit à cette fin.

Le trait commun entre ces versions de ACCEAO réside sur leur environnement technique, le système d'exploitation MS-DOS.

2.2. Architecture de ACCEAO sous DOS

Le logiciel *ACCEAO* a sa propre architecture, pas très différente des architectures des programmes normalisés. Il doit être compatible à plusieurs systèmes d'exploitation comme Windows, Unix, Linux.

ACCEAO est constitué d'une fonction *main ( )* qui est la fonction principale. C'est la *fonction* que le compilateur appelle en premier lieu quand on exécute le programme. Pour *ACCEAO* (Fig.2.3), la fonction *main ( )* est subdivisée en deux grandes parties. Il y a tout d'abord, une partie du programme qui crée une *interface utilisateur* et puis une autre partie pour la simulation de cette interface nommée *boucle de message*. Ces deux programmes utilisent des *fonctions* qui se trouvent dans un autre *fichier*.

Pour la création d'interface utilisateur, des *fonctions définies* dans les autres fichiers (***.cpp**) et des *fonctions prédéfinies* dans le logiciel de développement sont utilisées. Le principe est basé sur la manipulation des *fonctions graphiques* pour créer des *objets graphiques*. Et c'est l'assemblage de ces objets graphiques qui forme l'interface utilisateur.

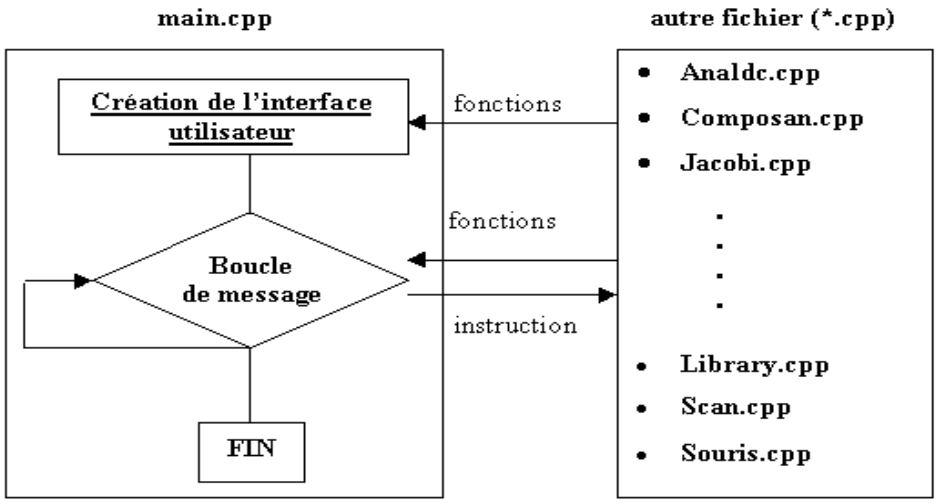


Figure 2. 3 : Architecture de *ACCEAO*

Ce dernier comprend les *menus*, les *sous-menus*, les *fenêtres*, les *boîtes de dialogue*, etc. Tous ces objets doivent être bien représentés pour rendre le logiciel convivial.

2.3. Vue générale de l'interface de la dernière version sous DOS

L'ACCEAO version 4.3 présente cinq menus (Fig.2.4) dont :

- le menu **File** englobant les sous-menus : **New**, **Load**, **Save**, et **Quit**. Chaque sous-menu permet respectivement d'obtenir une nouvelle fenêtre de traçage, le chargement d'un circuit existant, l'enregistrement d'un circuit sous un nom choisi par l'utilisateur et la fermeture du logiciel.
- le menu **Tools** contient :
  - **Devices** : permet de charger les divers composants pour constituer un circuit**,**
  - **Edit model** : permet de modifier les valeurs du composant, ou de le supprimer**,**
  - **Library :** contient des modèles de transistors et de diodes**,**
  - **Pspice :** bouton nécessaire pour interfacer ACCEAO avec le logiciel PSPICE**,**
  - **Source** : contient les alimentations pour le circuit : générateurs de signaux, masse.

- Le menu **Analysis** effectue tous les calculs et donne les résultats de l'analyse d'un circuit. On distingue **Node**, **DC**, **AC**, **MF**, **Transient**.
- Le menu **Design** est utile pour la conception d'un circuit à transistor à travers son sous-menu : **Qp triangle**. On effectue, avec ce dernier, le choix du type de transistor, ensuite le type de polarisation, et la saisie des paramètres du circuit sur une boîte d'invite afin d'avoir le circuit voulu.

Design, par son sous-menu, permet le traçage du diagramme de Bode.

Enfin, **Help** est le menu d'aide qui contient les renseignements concernant le logiciel ACCEAO et son utilisation.

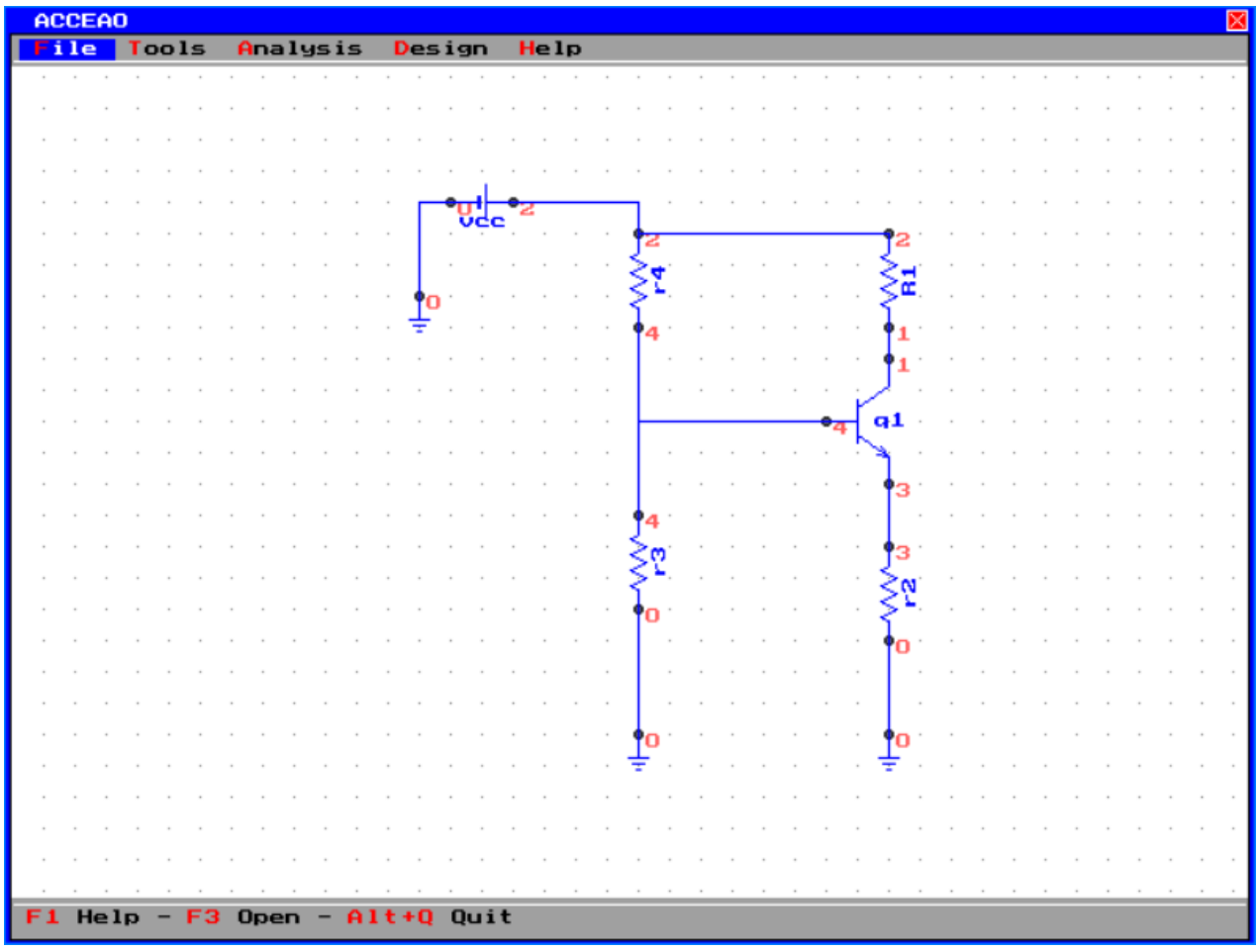


Figure 2. 4 : Vue générale de l'interface de ACCEAO 4.3

2.4. Le logiciel ACCEAO face à l'évolution

C'est surtout au niveau de l'interface qu'ACCEAO a rencontré un retard par rapport à l'évolution de l'informatique.

En effet, l'apparition du système d'exploitation Windows a fait apparaître une nouvelle donne dans le monde informatique concernant les logiciels.

Une maintenance adaptive est donc requise pour l'ACCEAO sans pour autant diminuer sa capacité d'analyse. Ainsi, l'ACCEAO version 5.0, objet de ce mémoire est une version sous Windows. Les étapes de sa migration sont évoquées dans le prochain chapitre.

Chapitre 3 MISE EN PLACE D'UNE NOUVELLE INTERFACE UTILISATEUR POUR ACCEAO

La structure graphique de ACCEAO a été totalement révisée en faveur d'une meilleure qualité visuelle et d'une facilité d'utilisation du logiciel. ACCEAO comporte trois fenêtres : une fenêtre principale servant pour la schématisation de circuits, une fenêtre pour l'affichage des résultats (des courbes et des valeurs) et une pour la saisie des caractéristiques d'un composant.

3.1. Construction de la fenêtre principale

La fenêtre en question : **ACCEAO_frm** est générée automatiquement avec le projet **ACCEAO.bpr**. Ses caractéristiques comme la dimension et le positionnement sont spécifiées par la suite. Chaque élément de la fenêtre est un composant de la bibliothèque visuelle de C++ Builder ou **VCL**[6].

En haut de la fenêtre, une barre est mise en place pour accueillir les menus déroulants. Cette **barre de menus** comporte le menu **Fichier**, le menu **Analyse** et le menu **Aide** ayant chacun ses propres sous-menus.

Une barre d'état est par contre placée en bas de la fenêtre pour afficher des messages afin de guider l'utilisateur pendant la manipulation. L'écriture de ces textes est possible grâce à la propriété **Text** de la barre.

Sous la barre de menus est située une barre d'outils servant essentiellement pour interfacer des accessoires utilisés lors des analyses. Elle accueille aussi des boutons de raccourcis des sous-menus : **Sauvegarder**, **Ouvrir** et **Nouveau**.

Une boîte de composants (composant de type **TTreeView**) placée au dessous d'un bouton « **Placer** » sert d'interfacer le choix d'un composant. En effet, **TTreeView** a la propriété d'accueillir une liste arborescente de noms. A chaque nom est associé un indice correspondant à un composant électronique et permet l'accès à toutes ses caractéristiques.

[6] Visual Component Library

Le champ de traçage des circuits est un cadre de type **TLabel**. Sa dimension en pixels est définie comme suit : sa longueur est de 3072 et sa largeur 1024. La surface de sa zone cliente permet au logiciel d'accueillir un circuit comportant environ 50 composants. Deux barres de défilement, verticale et horizontale, permettent de visualiser toutes les zones du champ.

Un point est tracé sur cette zone tous les 8 pixels pour servir de repère à l'utilisateur pour tracer son circuit. La fonction **Initdesktop** assure l'affichage de ces points. La figure 3.1 présente cette fenêtre avec tout son contenu.

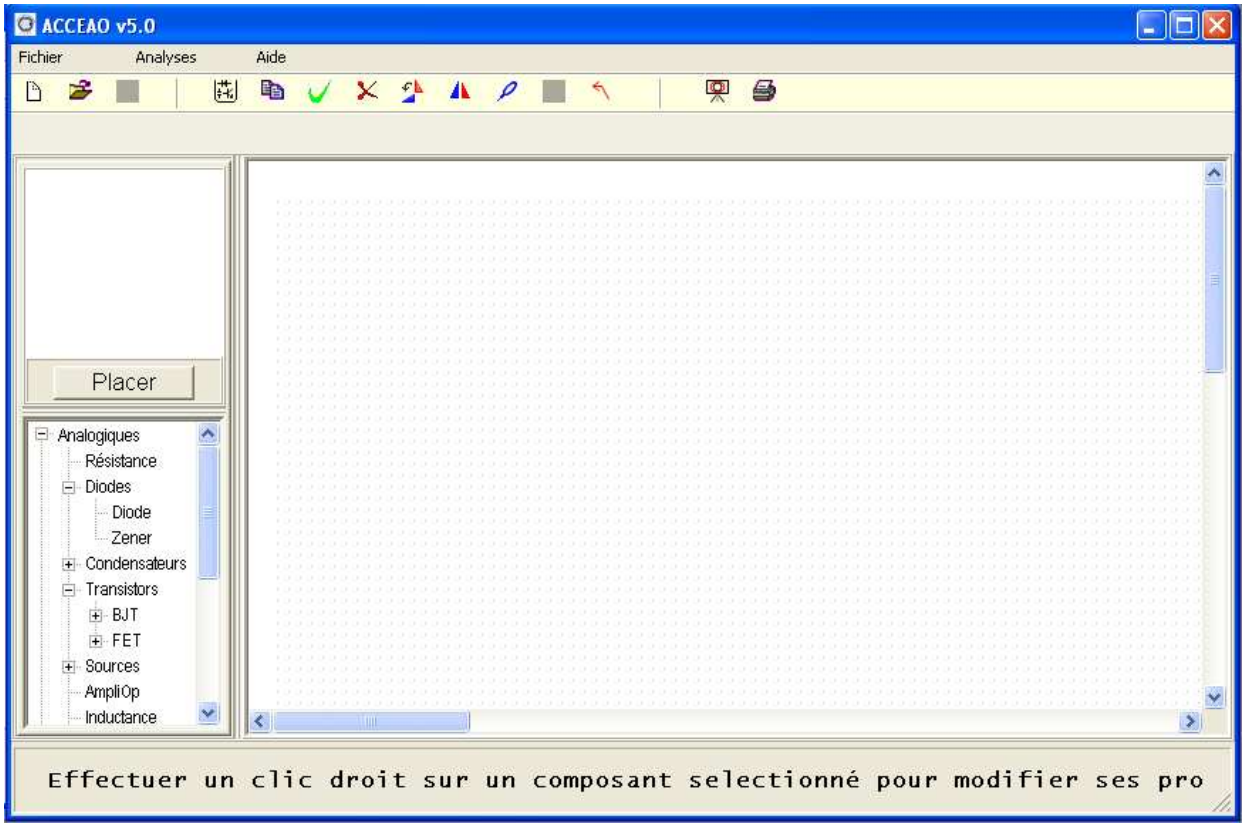


Figure 3. 1 : La fenêtre principale de ACCEAO

Affichage permanent de la fenêtre :

D'abord sous Windows, il est nécessaire de redessiner le contenu d'une fenêtre après chaque modification de son état.

Toute action de l'utilisateur sur la fenêtre, en outre sa minimisation, provoque donc son changement d'état. Dès que ce cas se présente pour **ACCEAO_frm**, tout le contenu du champ de traçage des circuits est effacé.

Pour permettre sa visibilité permanente, il est nécessaire de boucler l'image de cette zone de la fenêtre.

Comme l'utilisation d'une boucle infinie de type : for (; ;) est impossible sous Windows, une fonction ayant la possibilité de remplacer cette boucle (**MyIdleHandler**) a été créée. En effet, cette fonction permet d'écrire un gestionnaire d'événement pour effectuer des actions particulières quand une application est inoccupée.

Toute action de dessin s'effectue donc sur une image de format BITMAP, de même dimension que le champ. Puis **MyIdleHandler** se charge de l'affichage de cette image.

3.2. La fenêtre d'affichage des résultats

Le résultat obtenu dépend de l'analyse spécifiée par l'utilisateur. Pour une analyse transitoire, les courbes sont affichées. Pour l'analyse en régime permanent, le résultat est l'ensemble des valeurs de toutes les tensions aux nœuds du circuit.

a) La fenêtre servant à l'affichage des courbes

Cette fenêtre dénommée **ANALYSE_frm** est appelée pendant l'analyse transitoire pour afficher l'allure des courbes désirées. Comme dans la fenêtre principale du logiciel, un bitmap est toujours utilisé pour tracer les courbes. Ce dernier est réaffiché toutes les deux cents millisecondes grâce à l'utilisation d'un composant non visuel de C++ Builder : **TTimer**.

Une barre de défilement verticale est présente sur la fenêtre pour visualiser les courbes éventuelles qui pourraient exister en plus de la capacité de la zone cliente. La couleur de la zone cliente de **ANALYSE_frm** est fixée en bleu nuit pour permettre une bonne visibilité des courbes et aussi pour permettre l'utilisation de différentes couleurs de tracé (Fig.3.2).

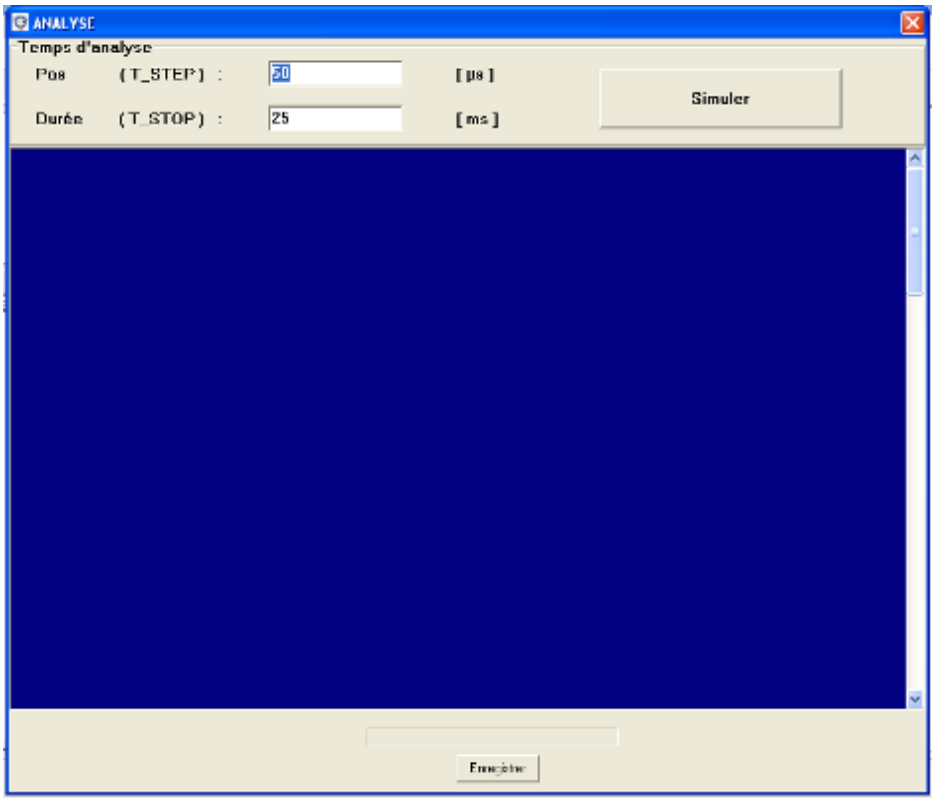


Figure 3. 2 : Fenêtre de la visualisation des courbes

b) Affichage de résultats des analyses

En dehors de l'analyse transitoire, l'affichage des résultats s'effectue sur la fenêtre principale avec le circuit. Pour l'analyse des nœuds, les numéros sont placés sur les bouts des composants du circuit. Une fonction **retrace** est appelée à chaque modification du circuit, effectuant ainsi l'effacement de tous ces numéros. En effet, une modification de la structure du circuit provoque un changement d'ordre des composants dans le vecteur **vcomp**, nécessitant ainsi une nouvelle analyse des nœuds.

Dans le cas de l'analyse en régime permanent, chaque tension est affichée à une distance de seize pixels du nœud correspondant.

Enfin, dans l'analyse en moyenne fréquence, un cadre de type **MessageBox** a servi pour l'affichage du résultat.

3.3. La fenêtre de saisie des caractéristiques d'un composant

Cette fenêtre nommée **SCAN_frm** est appelée pour la saisie des caractéristiques d'un composant. **SCAN_frm**, comporte des champs de saisie (composants **TLabel**) dont le nombre varie selon le type du composant. Elle se présente alors généralement sous deux formes :

- pour les composants actifs, elle comporte une bibliothèque dans sa partie gauche. Cette bibliothèque s'appuie sur un fichier **BIBLIO.FIL** qui lui sert de base de données. Trois boutons sont mis en places pour sa manipulation : **Ajouter**, **Editer** et **Effacer** (Fig.3.3).

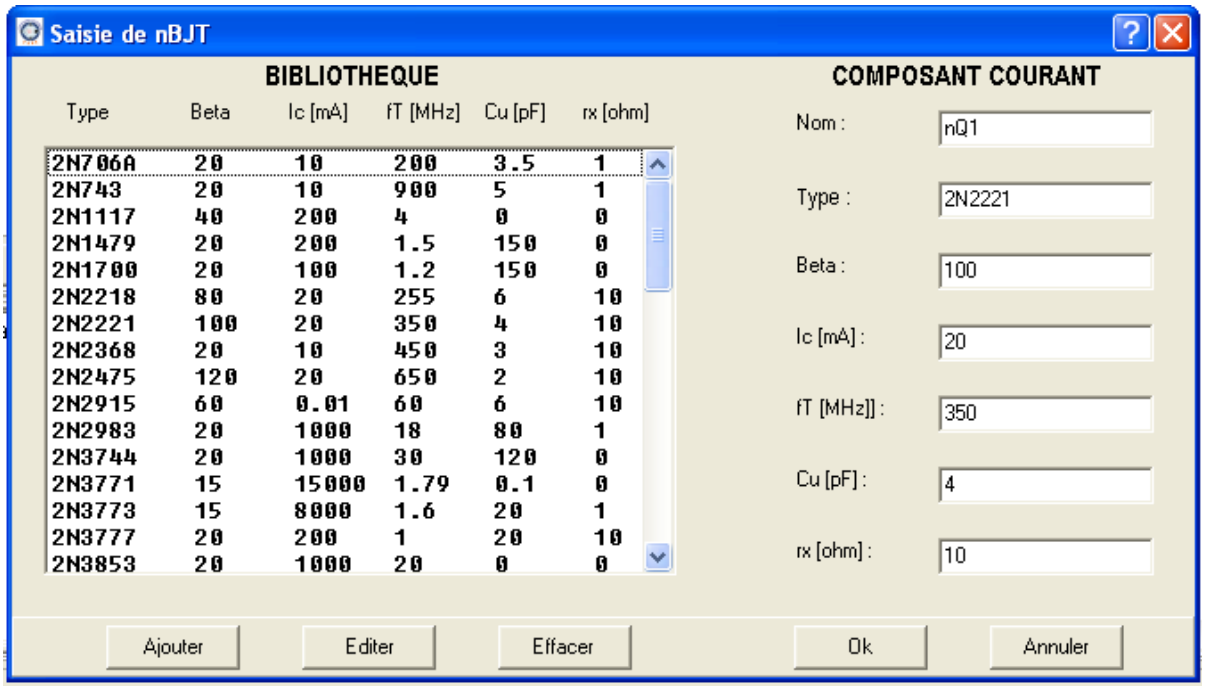


Figure 3. 3 : Fenêtre de saisie pour le BJT de type NPN

- Pour les composants passifs, elle ne comporte que les champs de saisie avec les boutons : **Annuler** et **OK** (Fig.3.4).

Figure 3. 4 : Fenêtre de saisie pour la résistance

3.4. Introduction de nouvelles fonctions pour l'interface

Dessiner un circuit dans la version sous DOS a demandé un temps considérable à l'utilisateur. Pour remédier a ce problème, quelques fonctions ont été introduites.

a) Une fonction pour générer le nom d'un composant

Globalement, chaque type de composant est identifié et son indice est incrémenté automatiquement si un composant de même type existe déjà sur le champ de conception.

Pour ce faire, la fonction **genere_nom** retourne un nom numéroté de composant (suivant le type) et qui sera affiché avec le composant (Fig.3.5). La fonction utilise un vecteur **vnc** dont chaque élément est formé par la **class ncomp**. Cette classe possède deux variables (type et numéro du composant) et deux fonctions membres pour accéder au numéro du composant (void set_num(int) et int get_num( )).

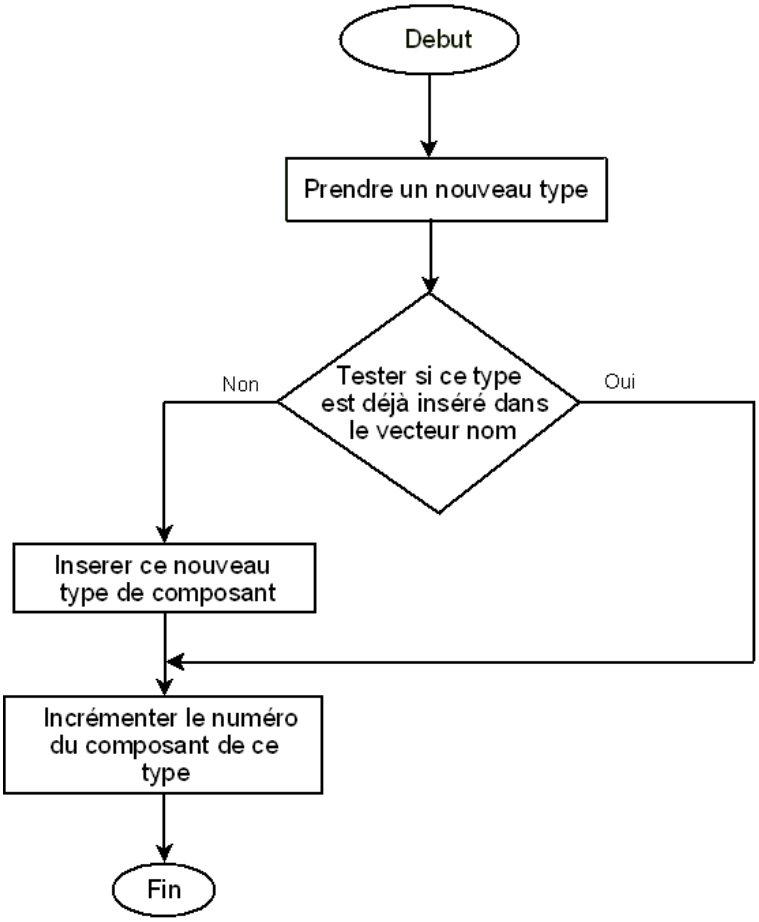


Figure 3. 5 : Organigramme pour le générateur de nom de composants

b) Une nouvelle fonction pour le traçage du fil de connexion

Ayant déjà existé dans la version sous MS-DOS, la fonction **line,** responsable de la définition de la forme du fil a été améliorée. Elle produit désormais huit styles de formes pour le tracé des fils entre deux pins de composants différents. En effet, un fil peut être formé soit par une, deux ou trois droites selon les distances qui les séparent : la distance verticale et la distance horizontale entre les deux composants à relier.

La détection de chaque bout de composant est témoignée par son clignotement en rouge, grâce à la fonction membre de la classe composant : bool **wires** (int &, int &). Cette fonction retourne une valeur booléenne vraie lorsque la pointe de la souris passe sur le bout en question.

3.5. Introduction de nouveaux composants

L'introduction de nouveaux composants a augmenté la performance du logiciel. Ces nouveaux composants sont donnés à la Fig.3.6. La partie conception aussi bien que la partie analyse a bénéficié de cette action.

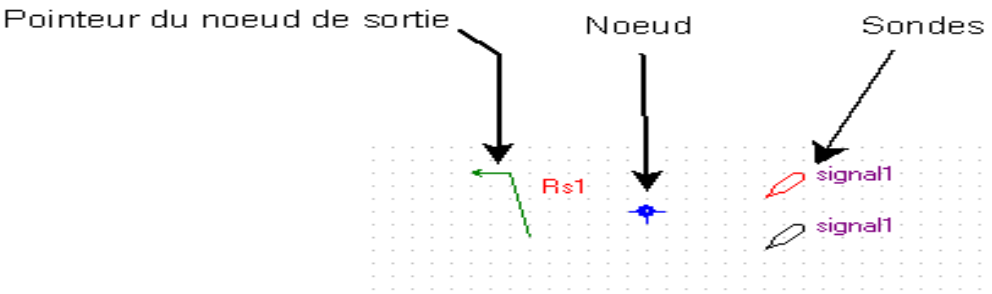


Figure 3. 6 : Les nouveaux composants

a) Le composant nœud

C'est un nouveau type de composant dont la fonction est d'être un point de liaison de deux fils. Un nœud a la propriété de se créer automatiquement sur un fil lors de la liaison. Cependant, il ne possède pas de valeurs caractéristiques comme les autres composants ; il ne sera donc pas pris en compte dans chaque analyse.

b) Des outils nécessaires pour les analyses

Ce sont des composants placés sur la barre d'outils dont chacun a une fonction spécifique dans une analyse où il va servir. Leur ressemblance se situe dans le fait que leurs actions s'effectuent sur les fils. La détection du numéro de nœud du fil du circuit en question par ces composants est possible grâce à la fonction membre de la classe **composant** : int **isonlin** (int ,int). Cette fonction retourne une valeur entière différente de zéro lorsque le composant a détecté la présence d'un fil.

Particulièrement, pour les sondes, comme celles d'un oscilloscope, sa fonction est de prendre le numéro de nœud sur le fil de connexion choisi par l'utilisateur.

Les composants pointeurs **Ren** et **Rs**, après détection du numéro de nœud du fil choisi, affichent respectivement la valeur de la résistance d'entrée et la résistance de sortie du circuit (exprimées en ohms), vue entre ce nœud et le potentiel 0.

c) Impression d’un schéma

Pour imprimer le circuit tracé sur un papier de format spécifié, un bouton comportant l’icone d’une imprimante sur la barre d’outils prend en charge cette fonctionnalité (Fig.3.7). L’utilisation d’un composant visuel de C++ Builder (**Printdialog**) permet d’interfacer le logiciel et l’imprimante à utiliser. Une boîte de dialogue d'impression, s’affichera pour aider l’utilisateur après la pression du bouton (Fig.3.8 et Fig.3.9).

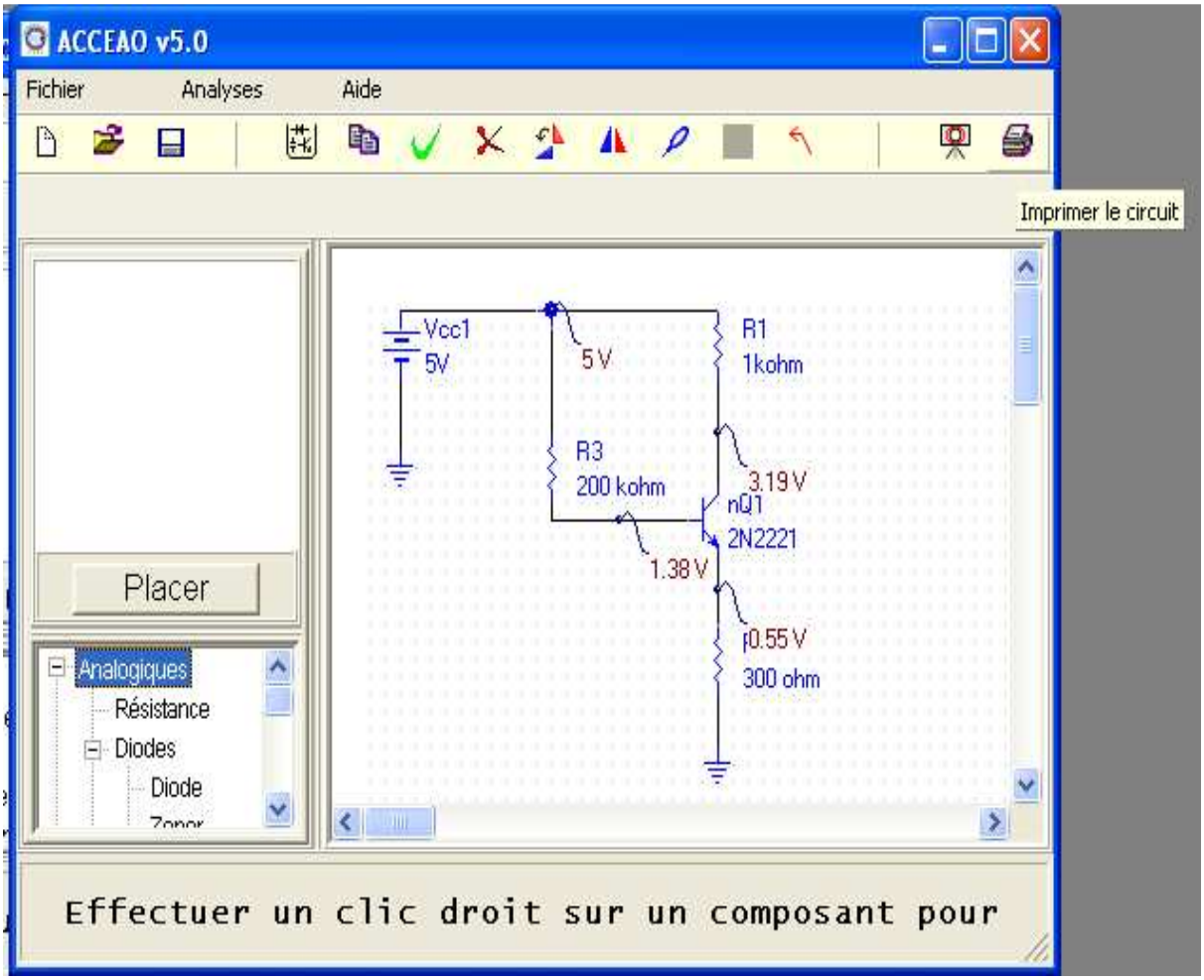


Figure 3. 7 : Exemple d’un schéma à imprimer

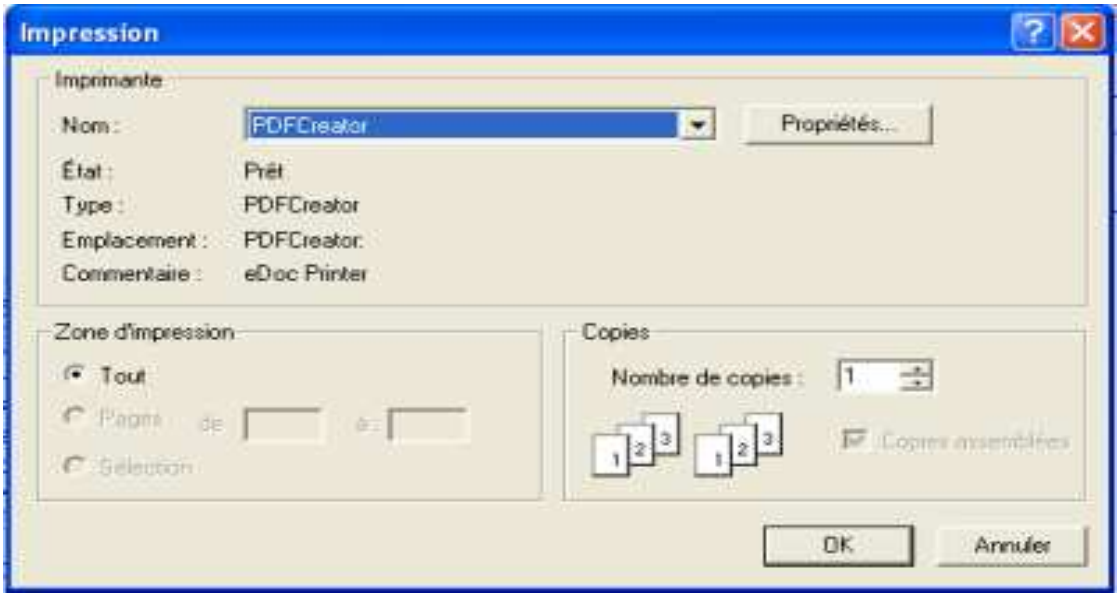


Figure 3. 8 : Boîte de dialogue d'impression

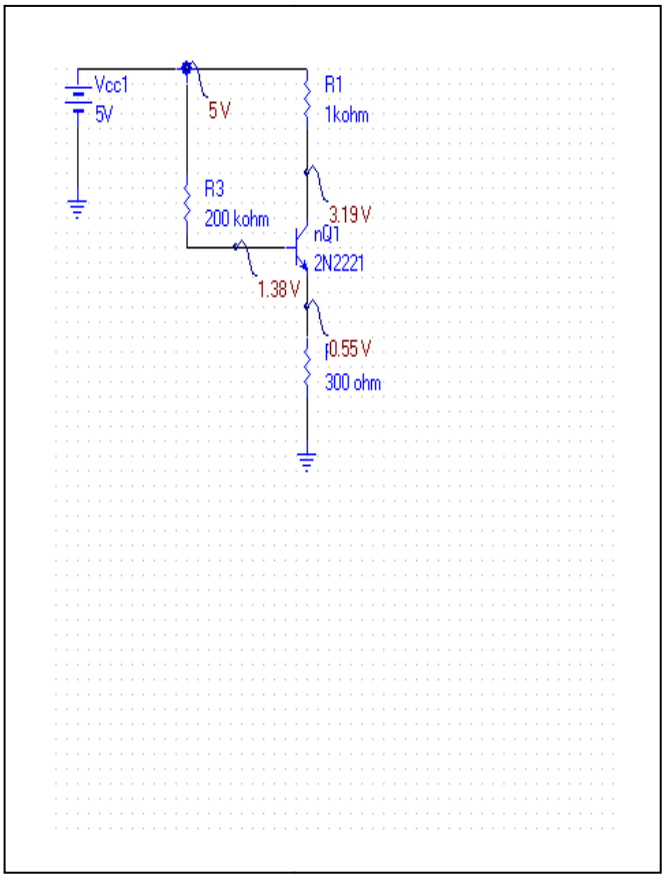


Figure 3. 9 : Schéma imprimé au format A4

Chapitre 4 UTILISATION DU LOGICIEL ACCEAO 5.0

Ce dernier chapitre sert de guide d'utilisation pour cette version 5.0 du logiciel ACCEAO. Outre l'utilisation de la souris et du clavier, la manipulation des nouveaux composants introduits est détaillée.

4.1. Lancement du programme

Après le lancement du programme **acceao.exe,** la fenêtre principale apparaît. Par défaut, le logiciel propose un fichier vierge. Deux possibilités s'offrent à l'utilisateur : utiliser ce fichier ou charger un fichier existant dans le dossier **exemples**.

a) Les menus

Le programme contient trois menus actifs dont : **Fichier**, **Analyse** et **Aide**. Chaque menu est accessible par la souris ou par des touches de raccourcis du clavier. De même, la plupart des sous-menus autorisent des touches de raccourcis.

b) Le menu **Fichier**

C'est un menu déroulant à quatre sous-menus.

(i). Le sous-menu **Nouveau**

Pour avoir une fenêtre de saisie vierge, on effectue un clic gauche sur le menu **Fichier** puis sur son sous-menu **Nouveau** (ou appuyer sur C**trl+N**). L'utilisateur peut alors saisir le schéma d'un circuit électronique (Fig.4.1).

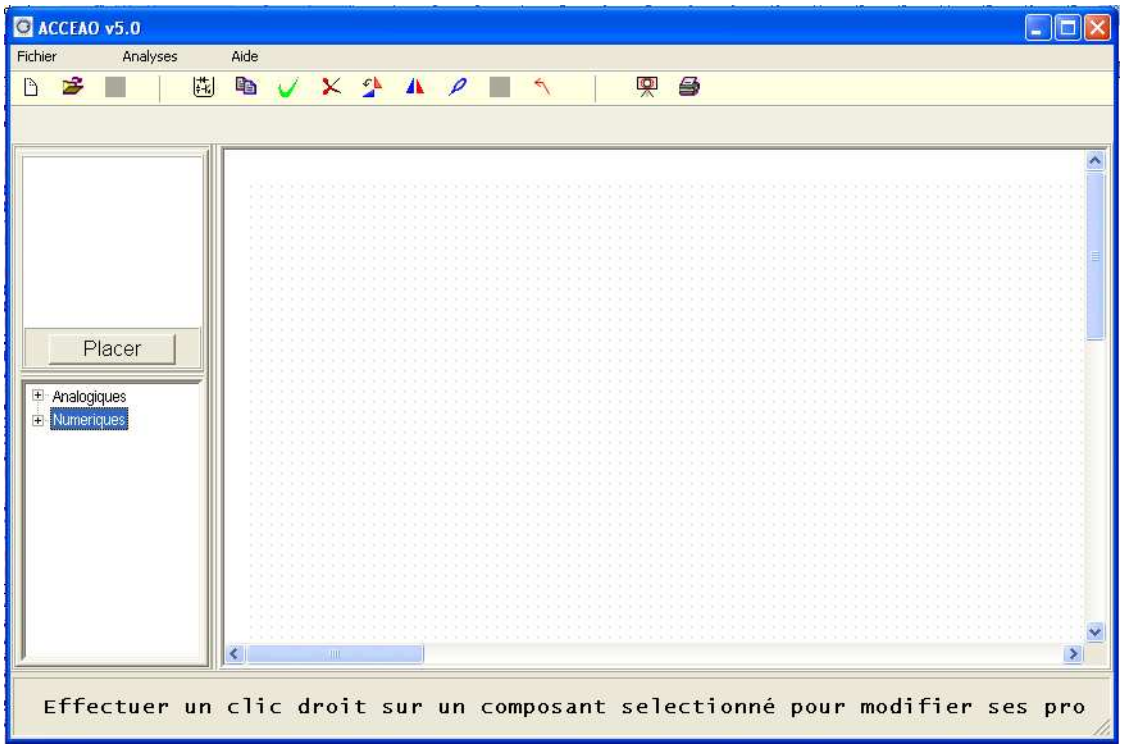


Figure 4. 1 : Obtention d'une nouvelle fenêtre de saisie

(ii). Le sous-menu **Ouvrir**

Pour charger un fichier ***.cao**, il suffit de cliquer sur **Ouvrir** (ou directement, appuyer sur **Ctrl+O** au clavier).

Une boite de dialogue contenant le nom des circuits dans le dossier **exemples** de l'application apparaît (Fig.4.2). Pour choisir un fichier particulier, double cliquer sur le fichier ou sélectionner le par un clic gauche du souris, puis appuyer sur le bouton **Ouvrir** de la boîte de dialogue.

Aussi, l'utilisateur peut à travers cette boîte, s'introduire dans n'importe quel répertoire pour une recherche éventuelle d'un fichier circuit. Notons qu'un fichier ***.cao** est facilement reconnaissable par son icône qui est le même que celui du logiciel.

Si un circuit est encore présent sur la fenêtre de saisie, un message d'avertissement s'affiche, demandant à l'utilisateur s'il veut sauvegarder le circuit actuel ou non avant d'ouvrir un autre.

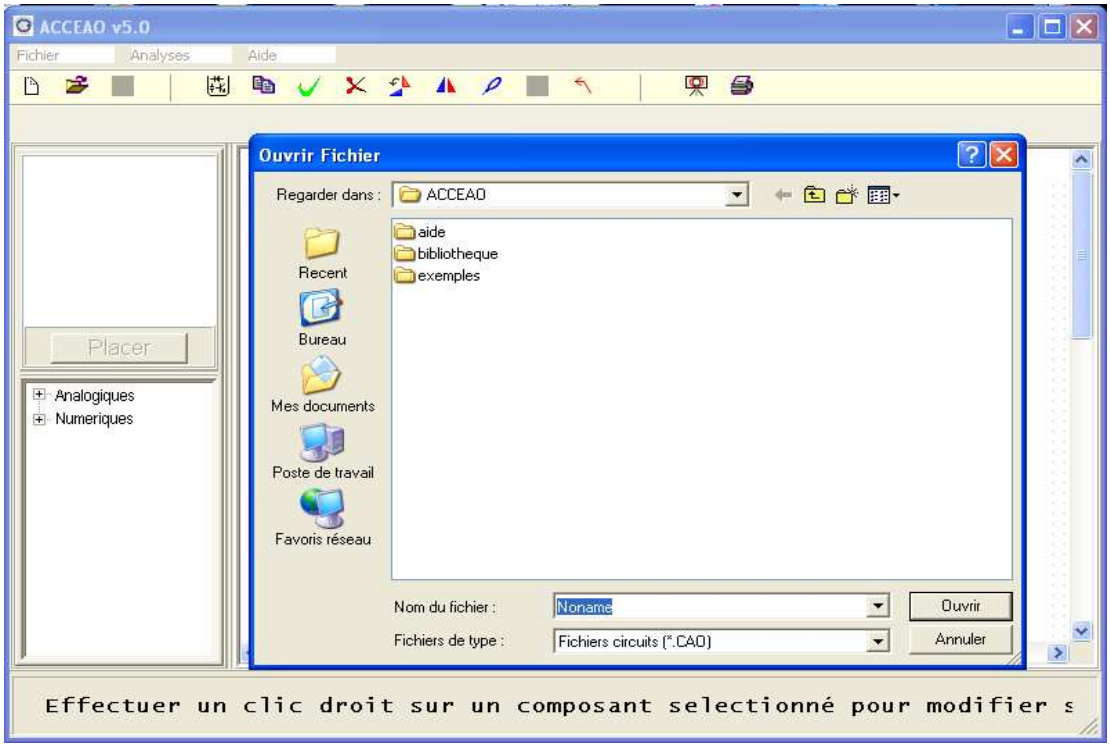


Figure 4. 2 : Chargement d'un circuit

(iii). Le sous-menu **Sauvegarder**

La sauvegarde d'un circuit se fait à travers ce sous-menu ou par la combinaison de touches **Ctrl+S**. Une boîte de dialogue pour la sauvegarde apparaît sur l'écran avec un cadre invitant l'utilisateur à saisir le nom du fichier. Après saisie du nom, appuyer sur le bouton **Enregistrer** pour confirmer l'action.

(iv). Le sous-menu **Quitter**

C'est le sous-menu servant à la fermeture du programme.

Dans le menu **Fichier**, choisir **Quitter** ou appuyer sur **Ctrl+Q**.

c) Le menu **Analyse**

C'est le menu contenant toutes les analyses possibles pour un circuit donné. Chacun de ses sous-menus présente un type d'analyse (Fig.4.3).

Les analyses disponibles sont :

- Analyse des nœuds

A travers cette analyse, on obtient la numérotation de chaque nœud de tout composant du circuit étudié. Les numéros sont affichés aux bouts du composant correspondant.

- Analyse en régime permanent (**Analyse DC**)

Cette analyse calcule la tension à chaque nœud du circuit et affiche le résultat près du nœud correspondant.

- Analyse en régime transitoire (**Analyse Transitoire**)

Si on la choisit, la tension sur un nœud déterminé par l'utilisateur est affichée sous forme d'une courbe dans une nouvelle fenêtre.

- Analyse en moyenne fréquence (**Analyse MF**)

Elle effectue le calcul de la résistance d'entrée, de la résistance de sortie et du gain en tension du circuit. Le résultat est affiché dans une fenêtre au milieu de l'écran.

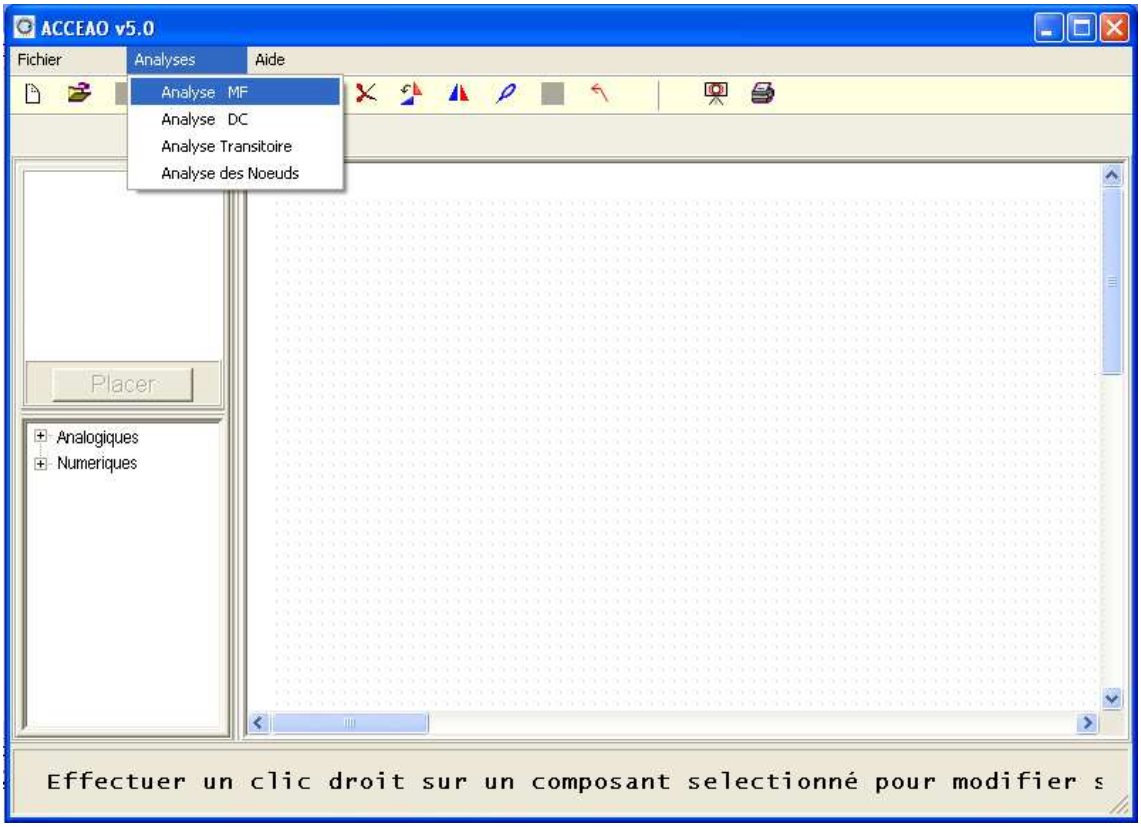


Figure 4. 3 : Les sous-menus existant dans le menu Analyse

d) Le menu **Aide**

C'est le menu contenant l'aide sur l'utilisation du logiciel ACCEAO et les renseignements le concernant.

Il contient donc deux sous-menus à savoir : **Aide sur ACCEAO** et **A Propos**.

- Le sous-menu **A Propos** offre les informations générales sur le logiciel, et sa date de création.

Effectuer un clic gauche sur le sous-menu en question et une fenêtre affichant ces informations apparaît (Fig.4.4).

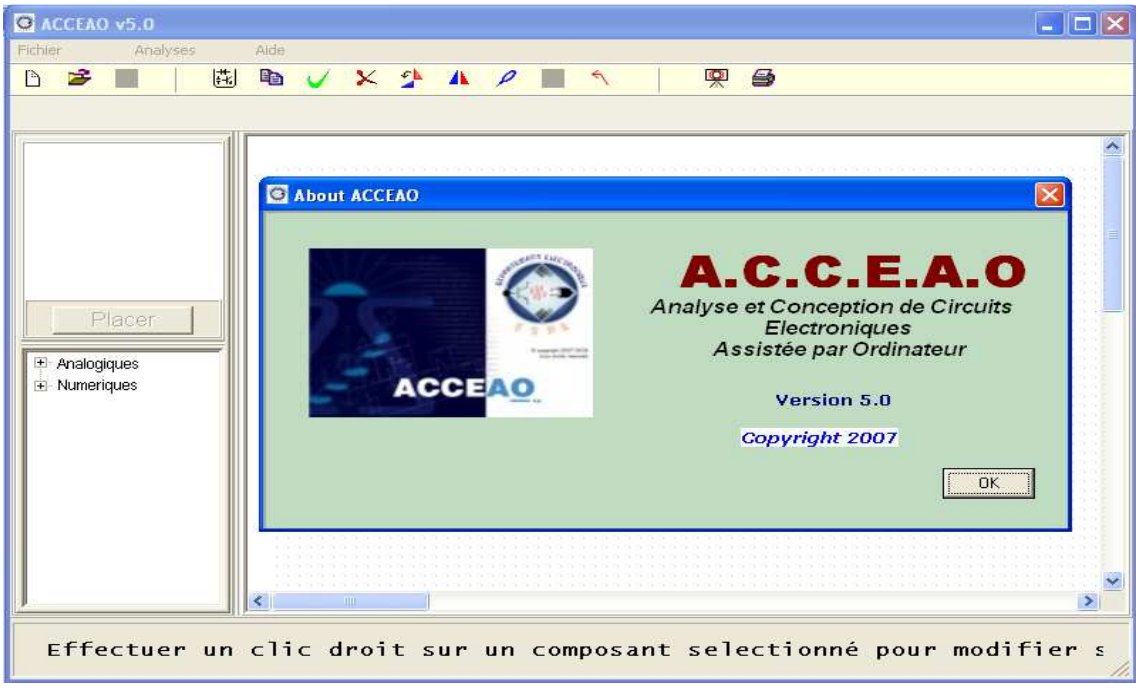


Figure 4. 4 : Affichage de la fenêtre **A Propos**

- Le sous-menu **Aide sur ACCEAO** affiche une fenêtre contenant le guide d'utilisation du logiciel (Fig.4.5). Pour naviguer dans cette fenêtre d'aide, un simple clic sur l'un des titres proposés conduit au détail de cette rubrique.

Pour accéder à ce sous-menu :

- Entrer dans le menu **Aide**, puis choisir la commande **Aide sur ACCEAO**,
- Cliquer sur une rubrique dans le sommaire de la nouvelle fenêtre, de couleur verte

- Une autre méthode de recherche s'offre en choisissant le menu **Index** de cette fenêtre d'aide.

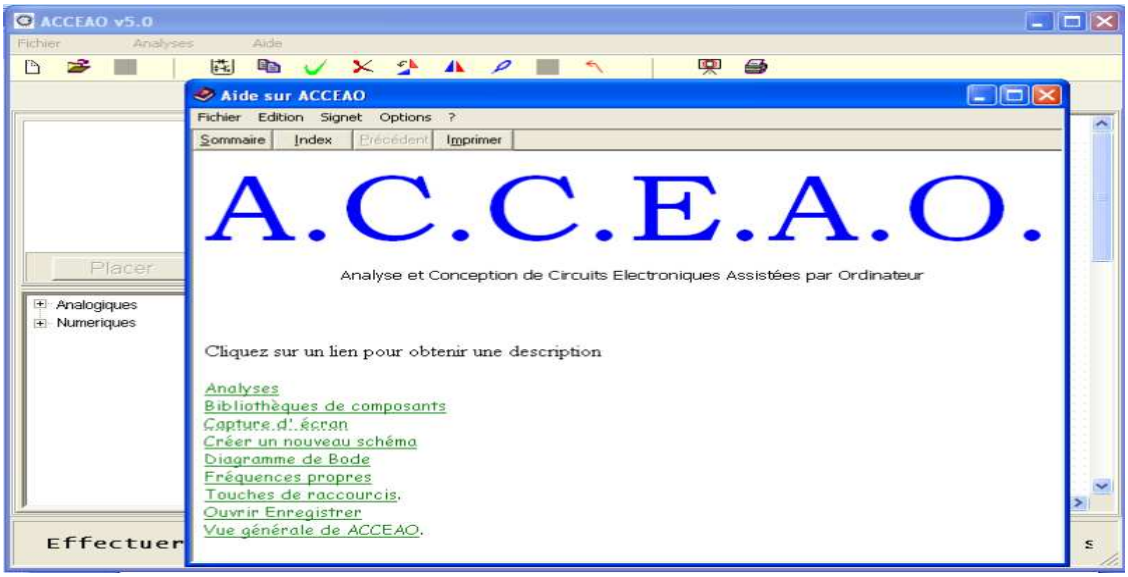


Figure 4. 5 : Affichage du fichier d'aide

4.2. La bibliothèque de composants

Affichée sous forme d'une liste, elle contient tous les composants électroniques existant du logiciel. Cette liste arborescente de noms de composants peut être consultée par l'utilisateur par un clic gauche (Fig.4.6).

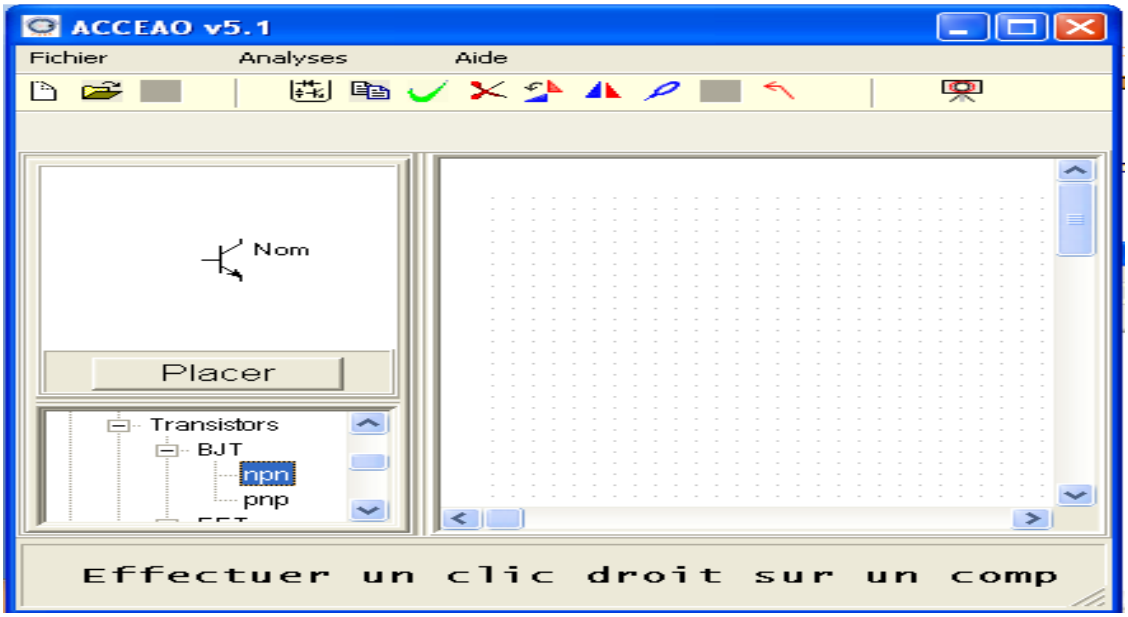


Figure 4. 6 : Consultation de la bibliothèque de composants (cas du transistor NPN)

4.3. Manipulation générale

La manipulation du logiciel relate les actions nécessaires à effectuer par son utilisateur, de la conception du circuit à l'analyse.

a) Etapes de schématisation d'un circuit

Pour tracer un circuit sur une fenêtre vierge, l'utilisateur doit procéder comme suit :

- Prendre un composant de la bibliothèque (Fig.4.7) :

  Pour cela, double cliquer sur le nom du composant choisi, le composant sera inséré sur le curseur de la souris. Une autre méthode consiste à sélectionner le nom du composant, puis l'appui sur le bouton **Placer** insère le composant sélectionné.

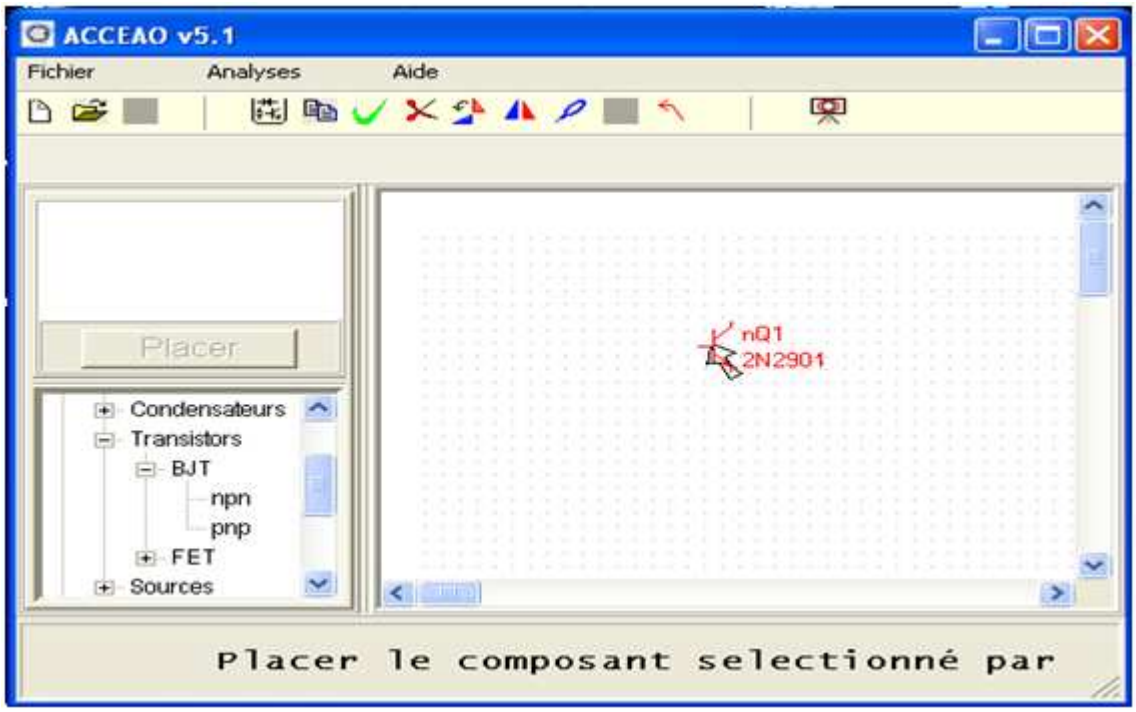


Figure 4. 7 : Prise d'un composant dans la bibliothèque

- Ensuite, poser le composant sur le champ de traçage. La grille sur le champ est utile pour aligner les composants à placer. En mode insertion, des mouvements comme la rotation sont possibles après le

choix du menu correspondant. On accède à ce choix en effectuant un clic droit, toujours en mode insertion.

- Tous les composants placés doivent être reliés par des fils, cette liaison se déroule comme suit :

  Cliquer sur un bout du premier composant, un fil va se créer immédiatement. Puis poser l'autre bout du fil sur un bout du deuxième composant (Fig.4.8).

  Le même procédé doit se répéter jusqu'à ce que tous les composants soient reliés.

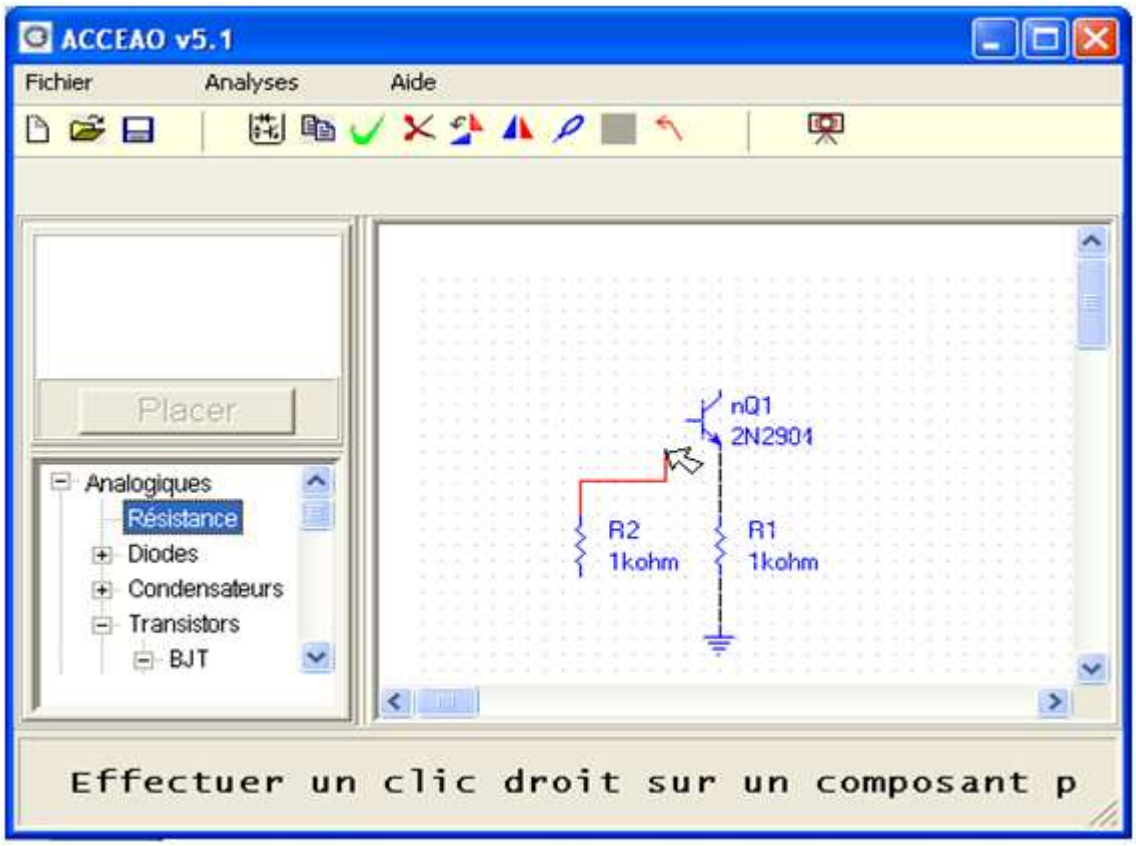


Figure 4. 8 : Liaison de deux composants par un fil

b) Les analyses disponibles pour un circuit et l'affichage des résultats

Pour un circuit donné, quatre analyses sont disponibles. Chacun a sa propre mode d'utilisation.

(i). L'analyse des nœuds :

Pour avoir les numéros de nœuds sur le circuit, un simple clic sur le sous-menu **Analyse des Nœuds** suffit.

(ii). L'analyse en régime permanent

Comme le précédent, le choix du sous-menu correspondant engendre l'affichage des tensions aux nœuds.

(iii).L'analyse en moyenne fréquence :

Cette analyse nécessite l'utilisation des pointeurs de nœuds situés sur la barre d'outils dont les noms sont **Rs** et **Ren**.

Ces deux pointeurs affichent respectivement la valeur en ohms de la résistance de sortie et de la résistance d'entrée du circuit (Fig.4.9).

A remarquer que **Rs** et **Ren** doivent se placer sur des fils lors de cette manipulation. Un clic sur le sous-menu : **Analyse MF** affiche ensuite le gain en tension du circuit avec les valeurs de la résistance d'entrée et de la sortie dans une fenêtre (Fig.4.10).

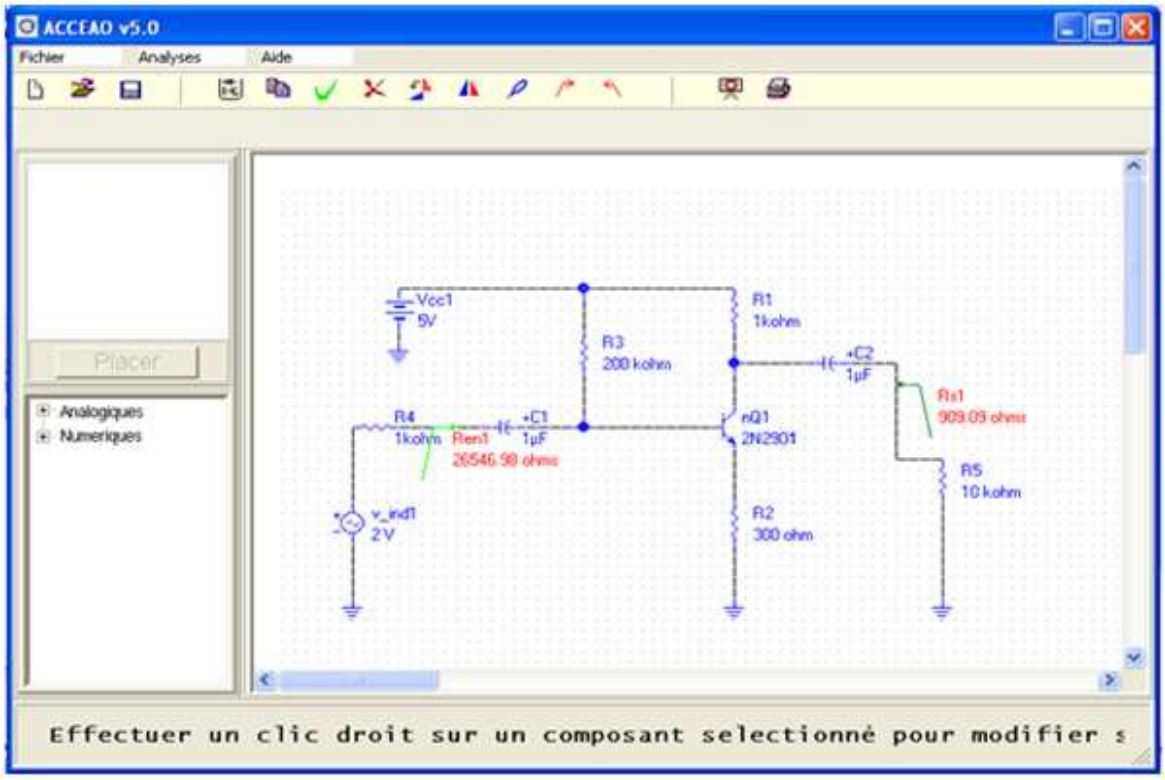


Figure 4. 9 : Utilisation des pointeurs **Ren** et **Rs** sur un amplificateur opérant en classe A

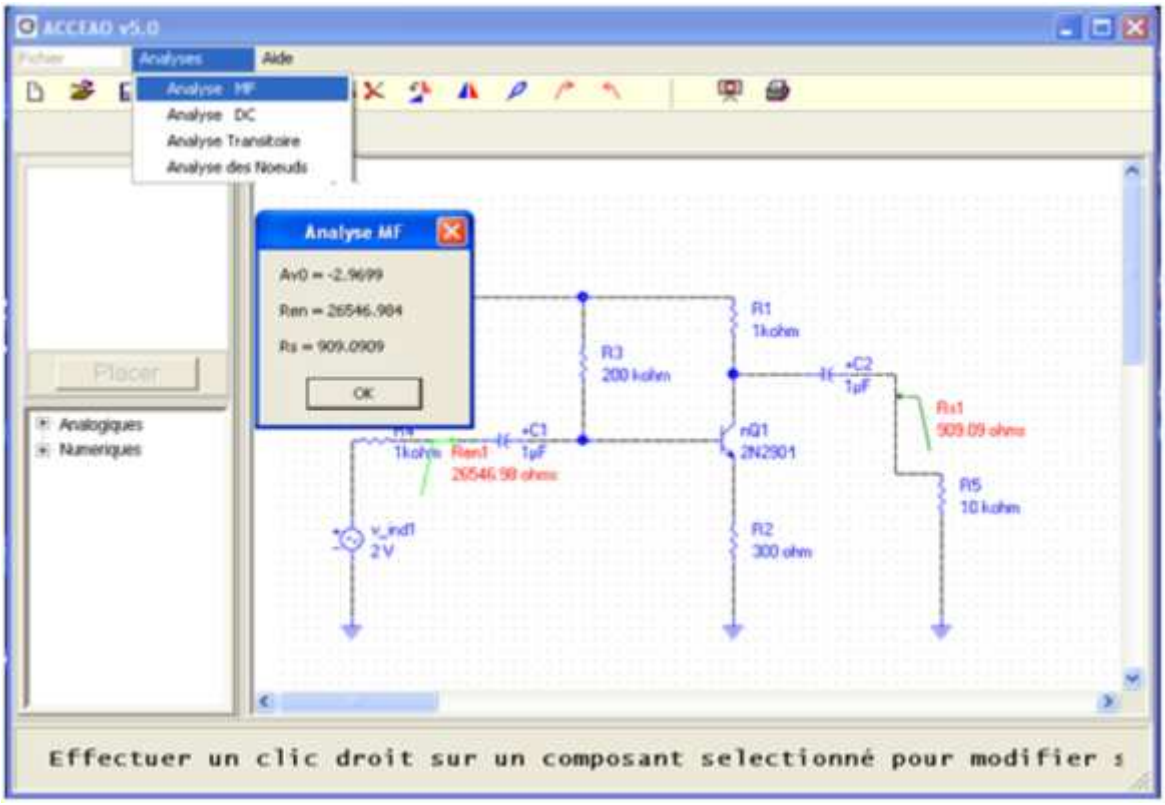


Figure 4. 10 : Affichage du résultat de l'analyse en moyenne fréquence

(iv). L'analyse transitoire

Cette analyse nécessite l'utilisation d'un composant sur la barre d'outils : les sondes. Les sondes servent à obtenir chacune un numéro de nœud.

A chaque prise, on obtient une paire de sondes. Un procédé doit être suivi après cette prise :

- Placer la sonde rouge (borne positive) sur un fil, la noire est la borne négative. On n'est pas forcé de placer la sonde noire car elle est par défaut reliée à la masse (Fig.4.11).
- Prendre autant de sondes, proportionnellement au nombre de signal à observer.

Après apparition de la fenêtre servant à afficher les courbes, il faut remplir les paramètres de temps : **Pas** (T_STEP) et **Durée** (T_STOP). Un appui sur le bouton **Simuler** débute le traçage des courbes (Fig.4.12).

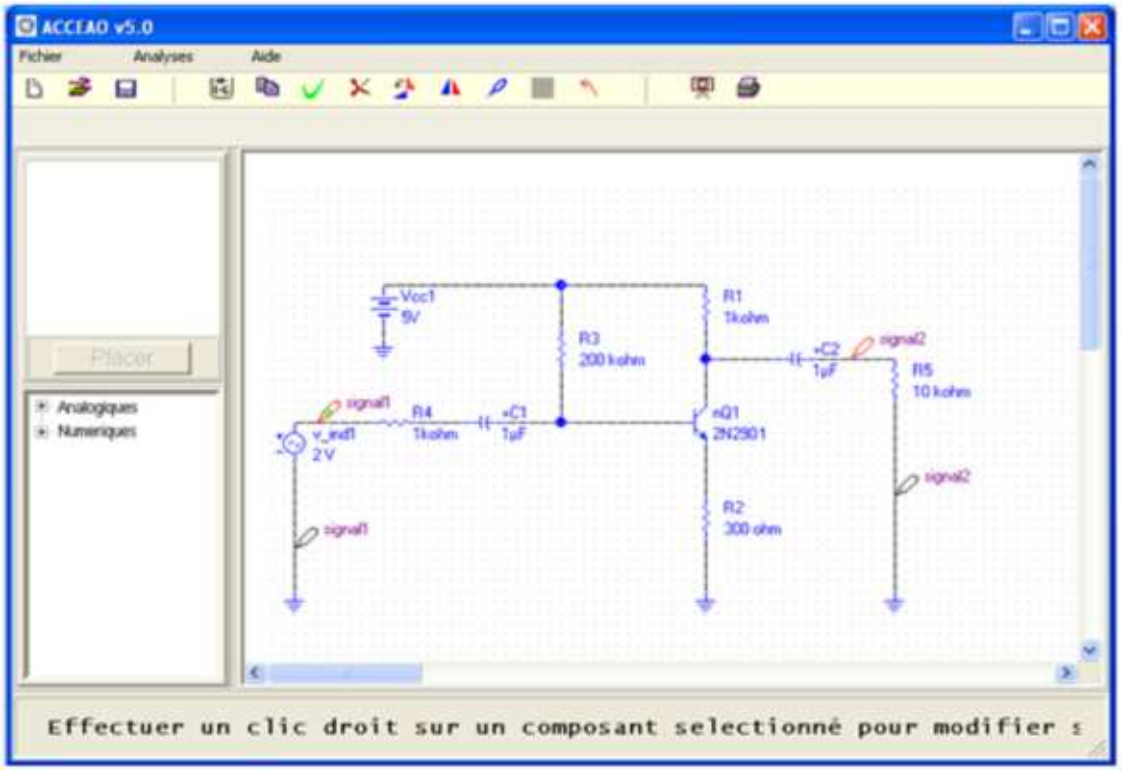


Figure 4. 11 : Utilisation des sondes pour l'analyse transitoire

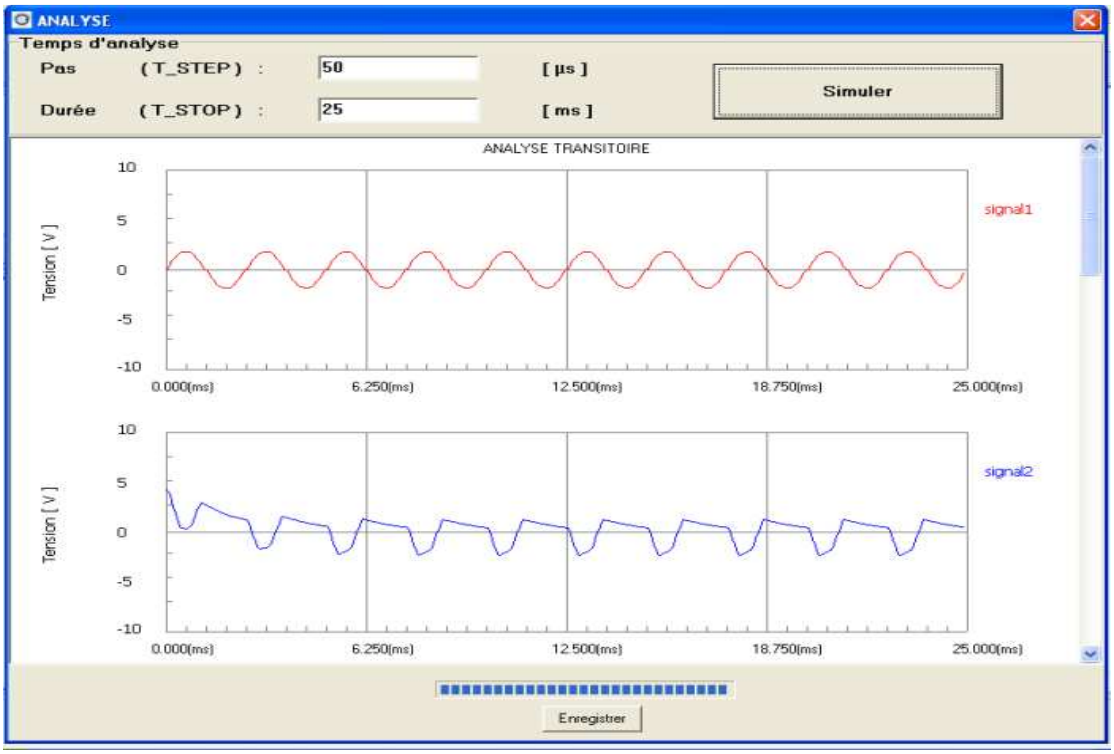


Figure 4. 12 : Affichage des courbes obtenues

c) Les sauvegardes

(i). Sauvegarde de circuits

La sauvegarde d'un circuit est effectuée par le biais de la commande **Sauvegarder** du menu **Fichier**. Un bouton raccourci correspondant est situé sur la barre d'outils. Le fichier sauvé doit porter l'extension **.cao** pour pouvoir être reconnu par le logiciel.

(ii). Sauvegarde d'image

Présente à la fois dans la fenêtre principale (sur la barre d'outils) et en bas de la fenêtre d'affichage de courbes lors de l'analyse transitoire ; cette option permet la copie de l'image de la fenêtre dans un fichier image.

Le choix du bouton en question, fait apparaître une boîte de dialogue demandant à l'utilisateur de saisir le nom du fichier de sauvegarde.

Par défaut, ce fichier a pour extension : **.bmp**.

4.4. La bibliothèque pour les éléments actifs

Cette bibliothèque contient des exemples de composants avec ses caractéristiques pour un type donné. Spécialement pour les éléments actifs comme les transistors et les diodes, elle se situe à gauche des champs de saisie et présente ses éléments sous forme d'une liste de chaines de caractères (Fig.4.13). On effectue un clic droit sur le composant sélectionné, ensuite un clic gauche sur **Editer modèle** pour la faire apparaître.

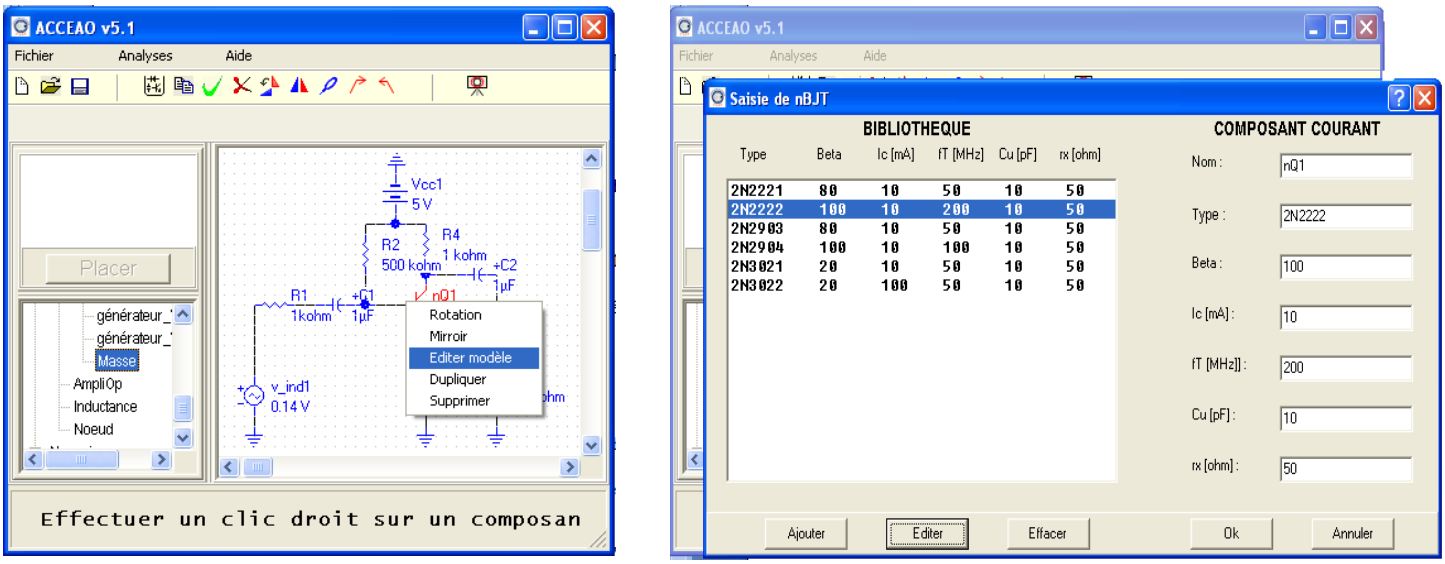


Figure 4. 13 : Affichage de la bibliothèque pour un transistor de type NPN

L'utilisateur peut consulter et changer les valeurs dans cette bibliothèque.

a) Ajouter un élément

Pour ajouter un élément dans la bibliothèque :

- Saisir les caractéristiques de l'élément en remplissant les champs de saisie,
- cliquer sur le bouton : **Ajouter**, un nouveau type est ajouté dans la catégorie.

a) Effacer un élément

Pour supprimer un élément de la bibliothèque :

- Sélectionner par un clic gauche la ligne de la bibliothèque à supprimer, elle sera colorée en bleu,
- choisissez le bouton **Effacer** pour confirmer l'action.

b) Editer l'élément sélectionné

C'est le chemin inverse de l'ajout :

- Sélectionner dans la bibliothèque la ligne à éditer,
- confirmer l'action par l'appui du bouton **Editer**. Les nouvelles caractéristiques s'affichent dans le champ de saisie.

4.5. Installation du logiciel

Pour sécuriser ACCEAO version 5.0, nous avons créé un programme d'installation. A travers ce programme, deux avantages en matière de sécurité sont obtenus.

En premier lieu, un code d'installation permet d'éviter tout piratage du logiciel. Aussi, une fois installé, sa suppression nécessite une étape de désinstallation.

Guide d'installation *:*

Les étapes de l'installation ressemblent à ceux d'un logiciel sous Windows après l'obtention du code. La méthode consiste à suivre les instructions écrites dans chaque fenêtre lors de l'installation.

Les deux points les plus importants dans ces étapes sont :

- La langue d'installation : le français,
- le choix du répertoire d'installation. Par défaut, le nom du répertoire est : C:\Programme Files\ACCEAO (Fig.4.14).

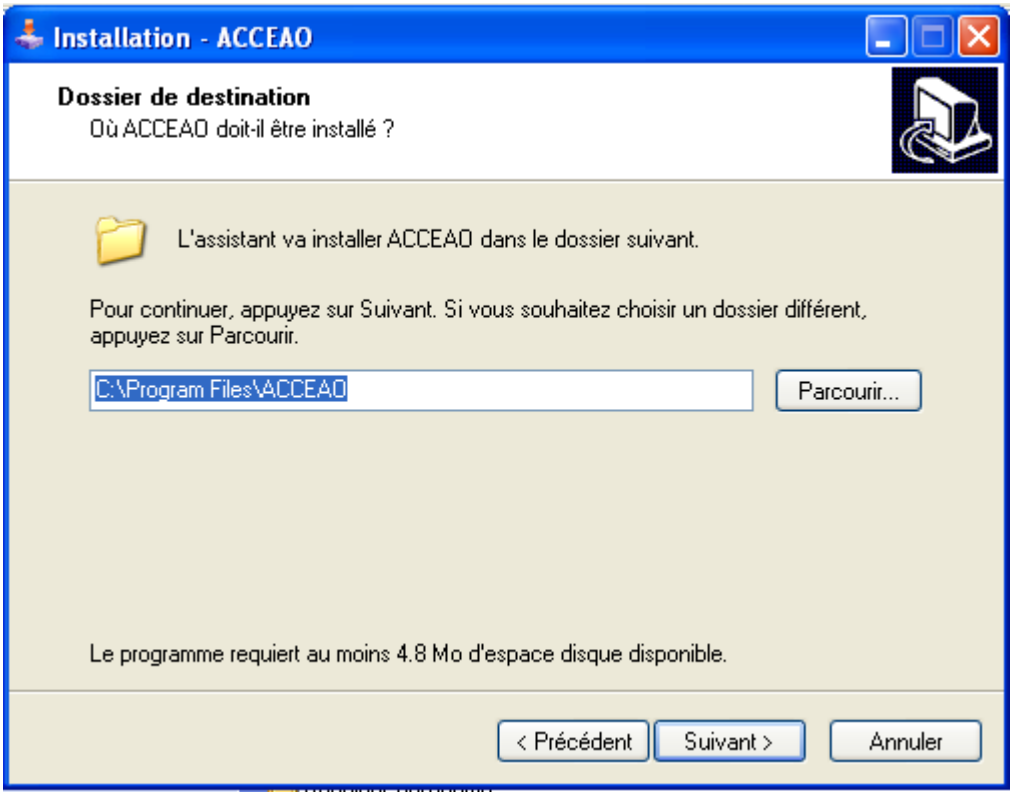


Figure 4. 14 : Fenêtre invitant l'utilisateur à choisir le répertoire d'installation

Configuration minimale nécessaire :

La principale condition requise pour permettre l'installation est le système d'exploitation Windows quelle que soit sa version. Cependant une configuration minimale de l'ordinateur est définie pour permettre son bon fonctionnement :

- Une mémoire vive de 32 Mo,
- Un espace disque de 4.80 Mo,
- Et un processeur de fréquence 400 Mhz.

CONCLUSION

La migration de ACCEAO sous MS-DOS vers le système d'exploitation Windows a été effectuée dans ce mémoire. Nous avons utilisé pour cela l'outil de développement C++ Builder 6.0 de Borland.

L'interface graphique a été totalement redéfinie par rapport à la version 4.3. Aussi, sont introduites de nouvelles fonctions (générateur du nom de composants, détecteur de bouts d'un composant, …) en plus de l'amélioration des fonctions déjà existantes.

Ainsi, cette version a fait profiter au logiciel les avantages offerts par Windows, entre autre sa possibilité de s'exécuter parallèlement avec d'autres programmes. De plus l'accès aux menus par la souris et le clavier est devenu plus fluide grâce notamment à la maîtrise des messages de Windows. Ces modifications ont permis un gain de temps évident lors d'un tracé de circuit.

Cette évolution au niveau de l'interface est accompagnée de l'augmentation de l'efficacité de l'algorithme de calcul.

Cependant, quelques difficultés ont été rencontrées lors de la mise en place de certaines fonctions, en particulier pour le traçage du diagramme de Bode.

Une étude plus approfondie sur ce cas et l'amélioration sur la puissance de calcul pourraient être les sujets lors des prochaines études. Dans la même foulée, la mise en place des fonctions permettant l'analyse des circuits numériques est aussi à considérer.

ANNEXE 1 : MAINTENANCE DE LOGICIEL

MAINTENANCE DE LOGICIEL

La maintenance est l'étape la plus longue dans le cycle de vie d'un logiciel. Elle est définie comme étant l'ensemble des processus d'entretien du logiciel après sa livraison [7], [8].

L'activité de maintenance consiste à effectuer des corrections, des améliorations ou des adaptations spécifiques du logiciel.

Objectif de la maintenance :

Gérer un processus de modification pour éviter que des corrections partielles soient faites en dehors du processus d'itérations : fidéliser le client.

Types de maintenance :

Il existe trois types de maintenance possible, pour un logiciel donné.

- **Maintenance perfective (évolutive)**

Elle consiste à maintenir les fonctionnalités antérieures tout en ajoutant de nouvelles fonctionnalités qui modifient profondément l'architecture.

Exemple : changement de SGBD (Système de Gestion de Bass de Données)…

- **Maintenance adaptative**

Ce type de maintenance consiste à adapter le logiciel aux changements de son environnement. C'est-à-dire, on adapte le logiciel au nouvel environnement technique. Elle est indispensable si lors la période d'utilisation, on se déplace vers un autre système d'exploitation.

- **Maintenance corrective**

Critère important de la qualité qui corrige les anomalies ou erreurs mises à jour par le client et non pas lors des tests de vérification et de validation.

Distribution de l'effort :

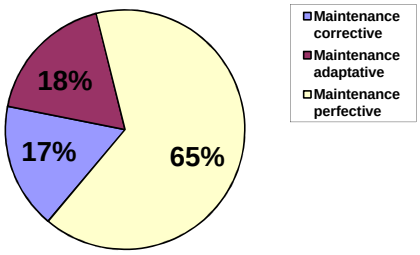


Figure A1. 1 : Diagramme en secteur de la distribution des maintenances

Processus de la maintenance :

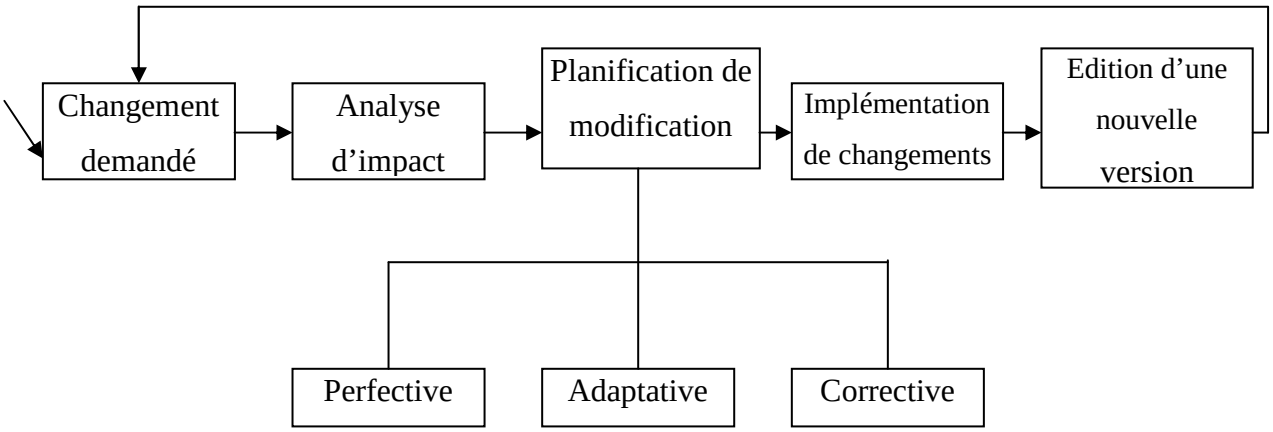


Figure A1. 2 : Schéma du processus de maintenance

Informations nécessaires pour la maintenance :

La facilité de la maintenance requiert la présence de certains paramètres.

- **De l'équipe de développement**
  - Si elle est encore en place,
  - Analyse : spécifications fonctionnelles,
  - Listing de sources,
  - Dossier de test
  - Algorithmes et les références
  - Options de compilations, standard utilisées…
- **De l'utilisateur**
  - Anomalie ou erreur (description précise), ou nouvelle fonctionnalité,
  - Contexte de l'erreur ou fonction,
  - Données du client,
  - Environnement technique…

Cycles de développement d'une correction :

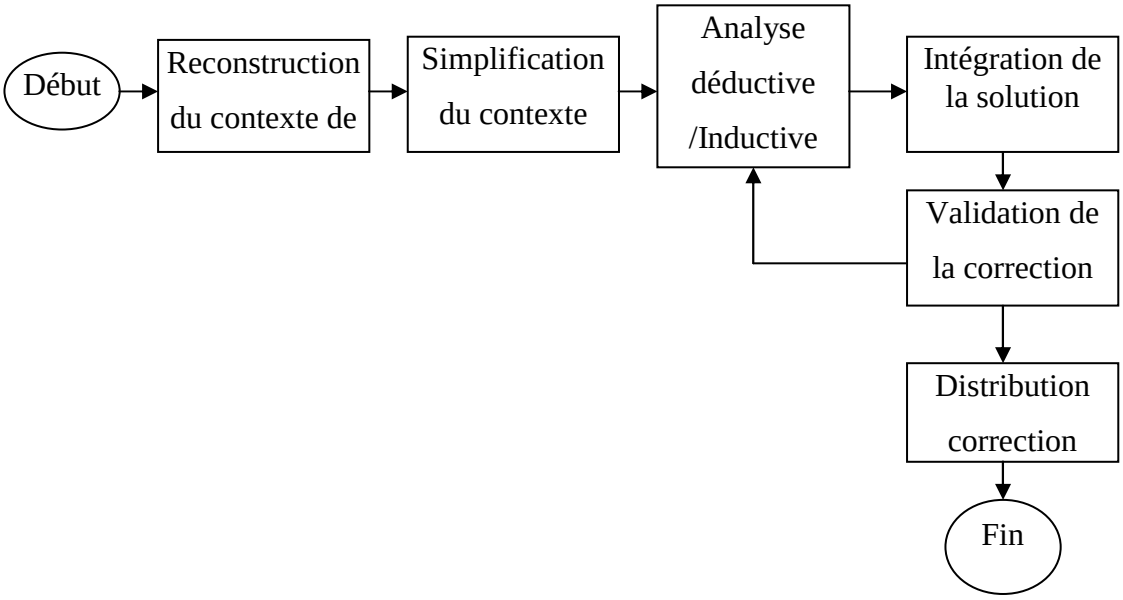


Figure A1. 3 : Schéma du cycle de développement d'une correction

Les effets de la maintenance :

On distingue trois catégories d'effets :

- **Effet de bord du codage**
  - Modification ou suppression d'un sous programme
  - Modification ou suppression d'un identifiant
  - Opérations logiques
  - Test de condition de sortie de boucle…
- **Effet de bord de données**

Il est induit généralement par une modification d'une structure de données ou d'un champ.

- Réduction ou augmentation de la taille d'un tableau ou structure complexe.
- Redéfinition de format de fichier
- Redéfinition de constante locale ou globale
- Réinitialisation d'un pointeur…

- **Effet de bord de la documentation**

Essentiellement l'adéquation entre le code source et les autres documents.

Remarque : Toute modification du code doit être reflétée dans les documents de maintenance, le manuel de l'utilisateur et le document de conception.

Maintenance du code étranger :

- **Définition**

Un code étranger est un programme (qui date de plus de 15 ans généralement) auquel aucun membre de l'équipe de maintenance n'a participé à son développement. Aucune méthodologie du génie logiciel n'a été appliqué (aucune structuré, documentation pauvre et incomplète et pas de sauvegarde de modification…). Que faire dans ce cas là ?

- Etudier le programme avant d'apporter une modification.
- Se familiariser avec le programme.
- Evaluer l'adéquation de la documentation.
- Insérer vos propres commentaires si vous jugez cela utile à la compréhension.

- Ne jamais éliminer un élément du code avant de s'assurer qu'il n'est pas utilisé ailleurs sinon avec beaucoup de précautions.
- Indiquer absolument toute instruction que vous avez changée sur le listing.
- Eviter de partager les variables (locales), déclarer les vôtres pour éviter des collisions.

RE-INGENIERIE :

Dans le domaine du génie logiciel, cela signifie processus qui ne consiste pas seulement à redécouvrir la conception des logiciels existants mais aussi à utiliser cette information pour reconstruire le système existant dans le but d'améliorer toutes ses qualités.

Quand décider la poursuite ou non de la maintenance?

Cela dépend du coût d'un nouveau produit (analyse) par rapport à celui de la maintenance. Si ce dernier est trop élevé, il est alors temps d'arrêter sa maintenance.

Est-il raisonnable qu'une entreprise envisage la réingénierie de tous ses logiciels?

Non, il a des logiciels qui n'auront pas à évoluer, les besoins n'évoluant pas. D'autres au contraire auront besoin d'évoluer rapidement.

MAINTENANCE EVOLUTIVE :

Considérons un programme « spaghettis » (programme non structuré). Que faire pour le maintenir?

- On peut le refaire complètement en utilisant l'atelier de génie logiciel et les principes du génie du logiciel.
- On peut travailler dessus jusqu'à arriver, modification après modification, au changement nécessaire (en introduisant les principes de génie logiciel).
- On peut attendre de comprendre le fonctionnement et la structure interne avant toute modification.
- On peut refaire une conception, implémenter et tester les parties du logiciel qui exigent une modification.

Quelle est la meilleure solution?

Cela dépend de ce que l'on veut réaliser, l'importance du logiciel et de l'entreprise. A priori, la deuxième solution est la moins bonne (modification après modification).

ANNEXE 2 : EXEMPLE DE DATASHEET

Tableau A. 1 : Transistors pour audio et applications générales

	TRANSISTORS											
	TRANSISTORS PETITS SIGNAUX											
	TYPE	POLARITE	BOÎTIER BROCHAGE	VALEURS V_{CE0} V	VALEURS Ic mA	VALEURS P_{tot} T_{amb} mW °C	CARACTÉRISTIQUES h_{FE} (h_{fe}) Ic mA		CARACTÉRISTIQUES f_T MHz typ.	CARACTÉRISTIQUES F dB typ	REMARQUES
TRANSISTORS POUR AUDIO ET APPLICATIONS GÉNÉRALES	BC107	n-p-n	TO-18 (1)	45	100	300 25	125-500	2	>300	2	Faible bruit
	BC108			20			125-900			2	
	BC109			20			240-900			1.2	
	BC140 BC141	n-p-n	T0-39 (1)	40 60	1000	3700 45	40-250	100	> 50		
	BC160 BC161	p-n-p	TO-39 (1)	40 60	1000	3700 45	40-250	100	> 50		
	BC177	p-n-p	TO-18 (1)	45	100	300 25	75-260			-	Faible bruit
	BC178			25			125-500	2	150	-	
	BC179			20			125-500			1,2	
	BC327	p-n-p	TO-92 (2)	45	500	500 25	100-600	100	100	-	Driver et sortie
	BC327A			60							
	BC328			25							
	BC337	n-p-n	TO-92 (2)	45	500	500 25	100-600	100	200	-	Driver et sortie
	BC337A			60							
	BC338			25							
	BC368	n-p-n	TO-92 (3)	20	1000	500 25	85-375	500	60	-	Sortie audio en classe B
	BC369	p-n-p	TO-92 (3)	20	1000	500 25	85-375	500	60	-	
	BC375	n-p-n	T0-92 (2)	20	1000	500 25	60-340	150	150	-	Sortie
	BC376	p-n-p	TO-92 (2)	20	1000	500 25	60-340	150	150	-	
	BC546	n-p-n	T0-92 (2)	65	100	500 25	110-450	2	300	2	Ampli audio
	BC547			45			110-800				
	BC548			30			110-800				
	BC550	n-p-n	TO-92 (2)	45	100	500 25	240-800	2	300	1.4	Faible bruit
	BC556	p-n-p	TO-92 (2)	65	100	500 25	75-475	2	150	2	Ampli audio
	BC557			45			75-800				
	BC558			30			75-800				
	BC559 BCS60	p-n-p	TO-92 (2)	45	100	500 25	125-800	2	150	1.2	Faible bruit
	BC635	n-p-n	TO-92 (3)	45	1000	1000 25	40-250	150	50	-	Driver
	BC637			60			40-160		130		
	BC639			80			40-160				
	BC636	p-n-p	T0-92 (3)	45	1000	1000 25	40-250	150	50	-	Driver
	BC638			60							
	BC640			80							
	2N930	n-p-n	TO-18 (1)	45	30	300 25	100-350 150-600	10	80	2.5 2 .0	Ampli faible niveau et faible bruit
	2N2483 2N2484	n-p-n	TO-18 (1)	60	50	350 25	< S00 <800	10	80	4 3	
	2N4030	p-n-p	T0-39 (1)	60	1000	500 25	40-120	100	>100		Grand signaux, faible bruit, petite puissance
	2N4031			80			40-120		>100		
	2N432			60			100-300		>150		
	2N4033			80			100-300		> 150		

Tableau A. 2 : Transistors pour commutation

	TRANSISTORS PETITS SIGNAUX											
	TYPE	POLARITE	BOÎTIER BROCHAGE	VALEURS				CARACTERISTIQUES				REMARQUES
				V_{CE0} V	Ic mA	P_{tot} mW	T_{amb} at °C	h_{FE} at	Ic mA	f_T MHz typ.	t_{on} ns max	Ic at mA	
TRANSISTORS POUR COMMUTATION	BC516	n-p-n	T0-92 (2)	30	400	625	25	>30.000	20	220			Transistors Darlington
	BC517	n-p-n	T0-92 (2)	30	400	625	25	>30.000	20	220			
	BSS50	n-p-n	T0-39 (1)	45	1000	500	25	>2000	500		1000	500	Transistors Darlington
	BSS51			60									
	BSS52			80									
	PH2222	n-p-n	T0-92 (2)	30	800	625	25	>75	10	>250	285	150	
	PH2222A			40						>300			
	PH2369	n-p-n	T0-92 (2)	15	500	500	500	40-120	10	>500	18	10	
	PH2907	p-n-p	T0-92 (2)	40	600								
	PH2907A			60		625	25	100-300	150	>200	100	150	
	PH5415	p-n-p	T0-92 (2)	200	1000	625	25	30-150	50	>15			Commutation haute tension
	PH5416	p-n-p	T0-92 (2)	300	1000	625	25	30-150	50	>15			
	2N1613	n-p-n	T0-39 (1)	50	500	800	25	40-120	150	>60			Ampli continu grande vitesse
	2N1711	n-p-n	T0-39 (1)	50	500	800	25	100-300	150	>70			
	2N2219	n-p-n	T0-39 (1)	30	800	800	25	100-300	150	>70			Commutation grande vitesse
	2N2219A			40									
	2N2222	n-p-n	T0-18 (1)	30	800	800	25	100-300	150	250			
	2N2222A			40									
	2N2901	n-p-n	T0-39 (1)	50	700	700	25	100-200	100	220			
	2N2904	p-n-p	T0-39 (1)	40	600	800	25	40-120	150	>220	100	150	Commutation grande vitesse et applications driver
	2N2904A			60									

Tableau A. 3 : Transistors à effet de champ

JONCTION - CANAL N											
Type	Valeurs à ne pas dépasser			caractéristiques							Boitier
	V_{DS} V	P_{tot} T_{amb} mW ^{at} °C		$-I_{DSS}$ max nA	I_{DSS} min-max mA	$-V_{(p)\,GS}$ V	$\|Y_{ts}\|$ min f = 1 kHz mA/V	C_{rs} typ pF	F typ dB	V_s max µV	
BC264A	30	300	25	10	2,0-4.5	> 0,5	2,5	1,2	0,5	-	TO-92
BC264B					3,5-6,5		3.0				
BC264C					5.0-8.0		3,5				
BC264D					7,0-12,0		4,0			.	
BF245A	30	300	75	5	2,0-6.5	80	3,0-6,5	1,1	1,5	-	TO-92
BF2458					6-15						
BFZ45C					12-25						
BF247A	25	250	75	5	30-80	0,6-14,5	8.0	3,.5	-	-	TO-92
BF247B					60-140						
BF247C					110-250						
BF256A	30	300	75	5	3-7	-	4,5	0.7	7,5	-	TO-92
BF256B					6-13						
BF256C					11-18						
BF410A	20 $^{+\,+}$	300	75	10	0,7-3,0	typ. 0.8	2,5	0,3	1,5	-	TO-92
BF410B					2,5-7,0	typ. 1,5	4,0				
BF410C					6-12	typ. 2,2	6,0				
BF510 $^{+}$	20	250	65	10	0.1-3.0	typ. 0,8	2,5	0,3	7.5	-	SOT-23
BF511 $^{+}$					2.5-7.0	typ. 1.5	4.0				
BF512 $^{+}$					6-12	typ. 2.5	6,0				
BF513 $^{+}$					10-18	typ. 3,0	7,0				
BFR30 $^{+}$	25	250	65	0,2	4-10	5	1,0-4.0	< 1,5	-	0,5	SOT-23
BFR31 $^{+}$					1-5	25	1,S-4,5				
BFT46 $^{+}$	25	250	65	0,2	0,2-1.5	1,2	1,0	< 1,5	-	0.5	SOT-23
BFW10	30	300	25	0,1	8-20	8	3.5-6,5	0,6	< 2.5	-	TO-72
BFW11					4-10	6	3.0-6,5				
BFW12	30	150	110	0,1	1-5	2.5	2,0	< 0,8	-	0.5	TO-72
BFW13					0.2-1,5	1,2	1,0				

Tableau A. 4 : Diodes à usage général

Type	Valeurs à ne pas dépasser					Caractéristiques		Boitier
						A T_j = 25°C		
	$I_{F(AV)}$ (A)	I_{FRM} (A)	I_{FSM} (A)	I_{FWM} (A)	V_{RRM} (V)	V_F (V) max.	à I_r (A)	
*BYD13-D	1,4	5,5	20	200	-	1,05	1	SOD 81
-G				400				
- J				600				
-K				800				
Δ BOY17-D	1,5	5.5	20	200	-	1,05	1	SOD 87
-G				400				
-J				600				
-K				800				
-M				1000				
'BYD14-D	2	40	50	200	-	1.15	3	SOD 84
-G				400				
-J				600				
-K				800				
-M				1 000				
'BY527	2	12	50	800	1250	1	1	SOD 57
'BYW54	2	12	50	600	600	1	1	SOD 57
BYW55				800	800			
BYW56				1000	1000			
'1N5060	2	12	50	400	400	1	1	SOD 57
1N5061				600	600			
1N5062				800	800			
'BYM56-A	3.5	20	50	200	200	1.25	5	SOD 64
-B				400	400			
-C				600	600			
-D				800	800			
-E				1000	1000			

Tableau A. 5 : Boîtier SOT 23/CMS

Types		Description	.	Valeurs à ne pas dépasser		t_{π} max.	V_F max. (mV) à I_F (mA)			C_d max	Marquage
Conventionnel	CMS			V_R (V)	I_F (mA)	(ns)	10	100	150	(pF)	
1N4148	BAS16	commutation rapide		75	250	6	855	-	1 250	2	A6
BAV19	BAS19	commut. hte tension		100	200	50	-	1 000		5	A8
BAV20	BAS20	commut. hte tension		150				1 000	-	5	A81
HAV21	BAS21	commut. hte tension		200			-	1000	-	5	A82
BAX12	BAS29	commutation rapide diode simple		90	250		750	900	-	35	L20
BAX12	BAS31	commutation rapide deux diodes séries	2X	90			750	900	-	35	L21
BAX12	BAS35	commutation rapide douhle diode An. Com.	2X	.						.	L22
1N4148	BAV70	Cathod. Com. double diode	2X	70	250	6	855	-	1 250	7,5	A4
1N4148	BAY99	deux diodes en série	2X								A7
1N4148	BAW56	An. Com, double diode	2X							2	A1

· 2X : Caractéristiques relatives à une diode (dans le cas des composant à 2 diodes).

Tableau A. 6 : Redresseurs ultra-rapides - **25/150 NS** - épitaxies

Type	Valeurs à ne pas dépasser					Caractéristique				Boîtier
	$I_{F(AV)}$ max. (A)	V_{RRM} max. (V)	V_{RWM} max (V)	I_{FRM} max. (A)	I_{FSM} max. (A)	t_π max. (ns)	V_F max. (V)	I_F (A)	T_j (°C)	
*BYV26-A**	1	200	-	10	30	30	2,5	1	25	SOD 57
- B**		400								
- C**		600								
-D**		800				75				
-E**		1000								
'BYD73-A	1,75	50	-	15	25	25	0,95	1	25	SOD 81
- B		100								
-C		150								
-D		200								
-E	1,7	250		13		50	1,05			
-F		300								
-G		400								
*BYV36-A	1,6	200	-	10	30	100	1,35	1	25	SOD 57
-B		400								
-C		600								
-D	1,5	800				150	1,45			
-E		1000								
Δ·BYD77-A	2	50	-	15	25	25	0.95	1	25	SOD 87
-B		100								
-C		150								
-D		200								
-E	1,85	250	-	13		50	1.05			
-F		300								
-G		400								
*BYM26-A	2.3	200	-	19	45	30	2.65	2	25	SOD 64
-B		400								
-C		600								
-D		800				75				
-E		1000								
*BYD74A	2,4	50	-	21	50	25	8,94	2	25	SOD 84
-B		100								
-C		150								
-D		200						-		
-E	2.15	250				50	105			
-F		300								
-G		400								
*BYV27- 50	2	50	-	15	50	25	1,07	3	25	SOD 57
-100		100								
-150		150								
-200		200								
*BYV28- 50	3,5	50	-	25	90	30	1.1	5	25	SOD 64
-100		100								
-150		150								
-200		200								

· A avalanche **Non épitaxie, doute diffusés Δ Nouveauté.

REFERENCES

[1] E 410 « Programmation en Langage Orientée Objet », Cours de 4ème Année, Département Electronique, ESPA, 2006.

[2] E 331 « Programmation en C », Cours de 3ème Année, Département Electronique, ESPA, 2005.

[3] Loïc YON, « Introduction à Borland C++ Builder 6 », http://www.kiux.net.

[4] RAMAROLAHY Tovohery et RANDRIAMALALARIVELO Andrison P, « ACCEAO version 4.2 », Mémoire de fin d'études, Département Electronique, ESPA, 2004.

[5] RAMASOMBOHITRA Nivonjy Nomen'Ahy et RABEANTOANDRO Anjaranotahina Jhonson, « ACCEAO version 4.3 », Mémoire de fin d'études, Département Electronique, ESPA, 2005.

[6] E 302 « Analyse et Conception de Circuits Electroniques », Cours de 3ème Année, Département Electronique, ESPA, 2004.

[7] E 411 « Génie Logiciel I», Cours de 4ème Année, Département Electronique, ESPA, 2006.

[8] Di Gallo Frédéric, « Cours de génie logiciel », CNAM BORDEAUX 1999-2000.

Auteur :

RAFIDINIAINA Tsiriantsolofo ***/Adresse***: II E66EDB Ampanotokana Tana-101

Titre : **“ACCEAO version 5.0 : Nouvelle version sous Windows”**

Nombre de pages : 57

Nombre de figures : 31

Nombre de tableaux : 06

RESUME

La migration du logiciel ACCEAO vers l’environnement Windows est réalisée à l’aide de l’outil de développement Borland C++ Builder 6.0. Cette maintenance a permis à cette nouvelle version 5.0 de ACCEAO d’être plus facile à manipuler et d’être plus performante du point de vue calcul par sa rapidité et sa précision.

Mots-clés : **ACCEAO**, migration sous Windows, maintenance, Analyse de Circuits Electroniques.

Rapporteur : Monsieur RABESANDRATANA ANDRIAMIHAJA Mamisoa