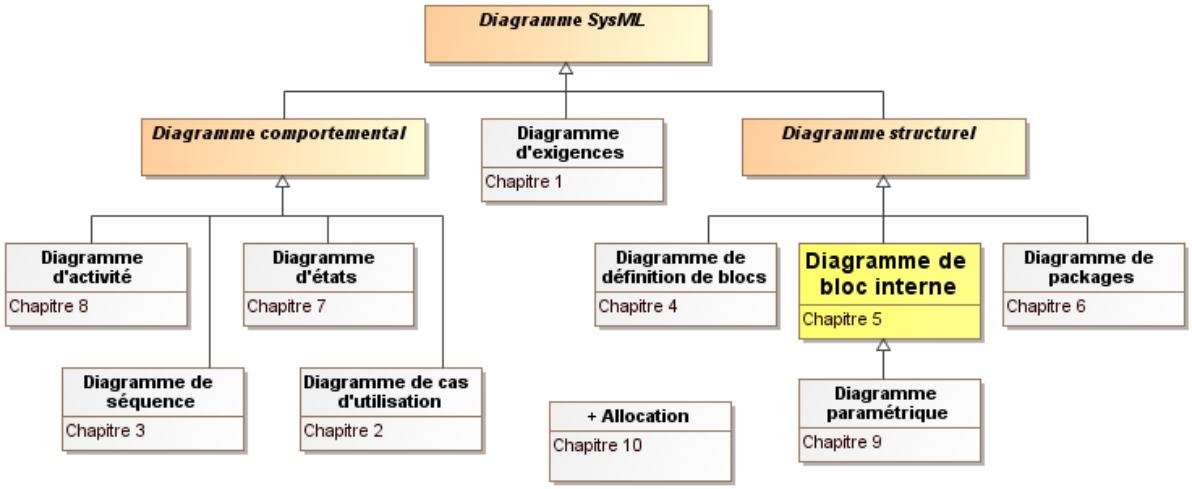

Système Information Voyageurs (SIV)

ACTIVITE 6

Utilisation MagicDraw

Programmation liaison RS232

SOMMAIRE

Sommaire

PRESENTATION

1 PRESENTATION DU TP:

L'objectif de ce TP est de mettre en œuvre la liaison RS232 entre le calculateur et le bloc BC 1004 afin de transmettre le paramétrage de ligne au boitier BC 1004.

Dans ce TP nous allons procéder de la façon suivante :

- A partir du diagramme SysML du système SIV nous allons faire le diagramme de classe correspondant à la liaison
- Ensuite nous allons apprendre à programmer en C++ la liaison RS 232 sous Windows
- Après quoi nous allons tester notre application .
- Enfin nous transposerons notre code C++ en C# puisque le calculateur utilise du C#

Public visé :

Etudiants en BTS SN options. IR

1.1 RESSOURCE MATERIEL :

Un pc de bureau ou un portable équipé de 4Mo de RAM un disque dur de 250 Go minimum système d'exploitation Windows XP Seven

1.2 RESSOURCE LOGICIELS :

MagicDraw version **17.0.3 sp1** ou supérieur lien de téléchargement : www.magicdraw.com/download

Plugin Sysml suivre le même lien de téléchargement.

Visual C++ ou NetBeans plugin C++ ou tout autre éditeur de texte

1.3 PRE REQUIS :

Connaissance de base du formalisme SysML, UML, liaison basic RS 232

2 PRESENTATION :

Le but et d'avoir un diagramme cohérent entre SysML et UML dans un même projet en effet un bloc interne correspond soit à une description plus fine mettant en œuvre du matériel soit une application informatique par conséquence dans le cas d'une analyse informatique une bloc interne contiendra l'analyse UML du système avec ses diagrammes de classes, ses diagrammes d'activité, diagrammes de séquence etc. A l'ensemble de cette analyse correspondra le code C++

La figure 1 représente cette encapsulation.

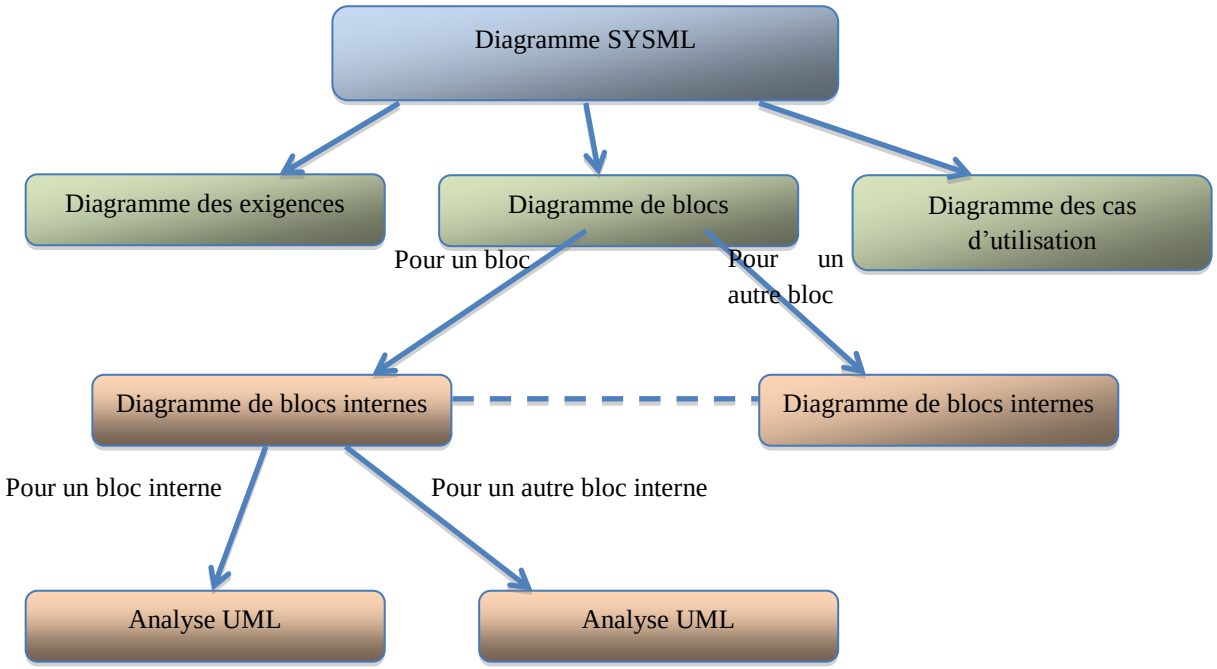

Figure 1 Encapsulation des différents diagrammes

NOTE 1

A un bloc interne peut correspondre plusieurs analyses UML suivant que l'on étudie

Attention à ne pas descendre trop bas dans l'analyse au risque d'avoir des diagrammes incompréhensibles trop de précisions tue la précisions.

3 MISE EN ŒUVRE DU DIAGRAMME DE CLASSE UML :

- A l'aide de MagiDraw ouvrir le projet SystemeInformationVoyageur
- Dans le menu Diagrammes choisir Diagramme de classes pour ouvrir la fenêtre de la figure 2

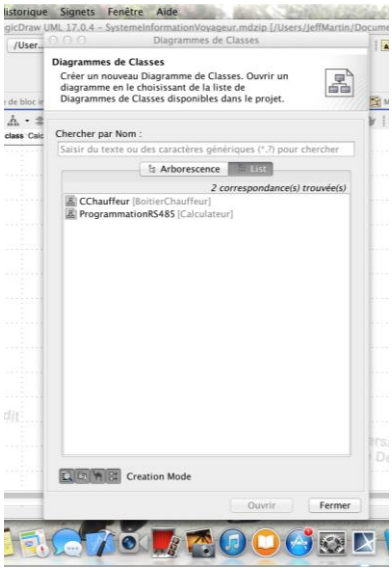

Figure 2 Création d'une classe

- Choisir l'onglet ***Arborescence*** pour voir apparaître tous les blocs interne
- Repérer le bloc ***Calculateur*** et le sélectionner. Puis valider votre choix en effectuant un clic sur le bouton **Créer**. Pour ouvrir la fenêtre de la figure 3
- Dans le champ **Nom** saisir le nom du diagramme de classe par exemple ***ProgrammationRS232***. Puis valider votre choix pour voir apparaître dans la fenêtre de la figure 3 l'arborescence de votre diagramme. Effectuer un **clic** sur le **bouton fermer** pour revenir sur la fenêtre principale ouverte sur le nouveau diagramme de classe comme le montre la figure 4

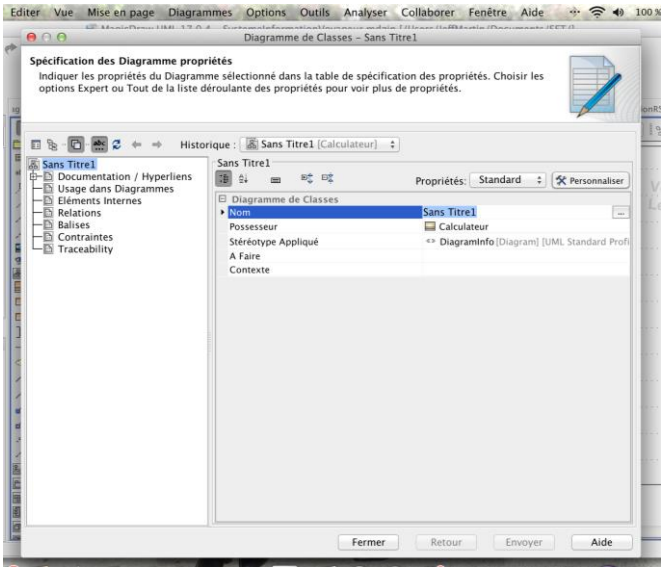

Figure 3 création d'un nouveau diagramme de classe

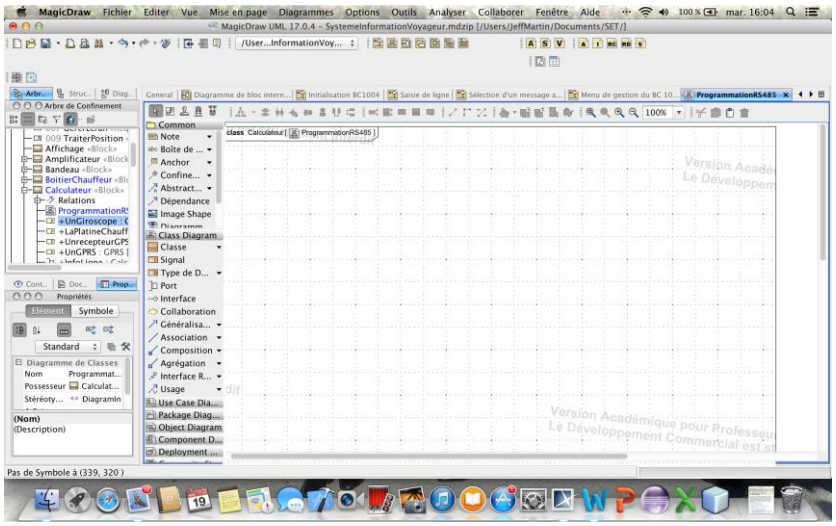

Figure 4 Feuille diagramme de classe

3.1 ANALYSE DE L'APPLICATION :

Le calculateur doté de son application informatique va envoyer via une liaison série une trame contenant les informations nécessaires au paramétrage des messages délivrés sur les girouettes

Par conséquence on doit concevoir et programmer une application permettant de paramétrer et gérer le port com du calculateur une difficultée supplémentaire est que le dit calculateur ne supporte que des applications développées en C# pour faire notre étude nous allons développer l'application sous visual C++ et ensuite nous transposerons notre code en C#

NOTE 2

Cette forme de développement C++ puis C# et choisit car le langage référence du BTS et le C++ mais bien entendu dans le cas d'un développement directe on aurait programmé en C#.

- Sur la feuille MagicDraw UML (diagramme de classe) sélectionner le symbole Classe et déposer le sur la feuille.
- Ouvrir la rubrique spécification et donner lui le nom CApplication. Puis refermer cette feuille.
- Placer un second symbole de classe et donner lui le nom CCommunication.
- Enfin placer un troisième symbole de classe et donner lui le nom de CBoitier

NOTE 3

Ce diagramme de classe à pour objectif de traduire qu'une application va communiquer avec un boitier BC1004 via un support de communication. Ce support de communication n'étant pas pour l'instant spécialisé.

- **La classe application** représente le système calculateur dans sa globalité il n'est pas question pour nous de construire la totalité de l'application SIV donc la classe application sera celle qui **contiendra le main**. Placer le pointeur de souris sur la classe CApplication. Effectuer un clic droit pour ouvrir le menu contextuel
- Dans le menu contextuel choisir l'option **Insérer nouveau opération** et saisir le nom **main.**

NOTE 4

La classe CBoitierBC1004 restera telle quelle. En effet elle n'est présente que pour comprendre le but et l'interaction réalisée par la classe communication. Il n'est pas nécessaire de développer une application pour tester le code de la liaison série.

3.1.1 CLASSE CCOMMUNICATION :

Cette classe représente le fait que pour communiquer nous allons devoir paramétrer l'interconnexion entre le calculateur équipé du port RS232 et le boitier BC1004.

La classe communication se veut être une classe générale en effet pour communiquer entre deux équipement on peut le faire par une liaison Série mais on peut tout aussi bien le faire par une liaison Ethernet ou CAN etc.

Quelque soit le support utilisé on devra envoyer un message et recevoir un message. C'est ce que nous allons signifier à notre classe Communication

- Comme pour la fonction main de la classe application vous allez indiquer à la classe **CCommunication** qu'elle possède la méthode ***EnvoyerMessage*** et ***RecevoirMessage***

3.1.2 SPECIALISATION DE LA CLASSE CCOMMUNICATION :

Notre communication est réalisée par une liaison particulière qui est une liaison série. C'est ce que nous allons montrer sur notre diagramme de classe.

- Placer un nouveau symbole en dessous de la classe communication et donnez lui le nom de CRS232.
- Cette fois nous allons définir complètement notre classe puisque c'est elle qui servira de support à la création des objets de ce type. Comme vous l'avez vu dans l'étude de la mise en œuvre d'une liaison série ce type de support doit être paramétré. On doit notamment connaître la taille des données pour chaque trame, le nombre de bits de stop etc. Comme vous l'avez fait précédemment ouvrir le menu contextuel de la classe CRS232 et ajouter les différents paramètres nécessaires au paramétrage d'une liaison série. Ces paramètres seront des attributs.
- Ouvrir la fenêtre Spécifications comme le montre la figure 5.

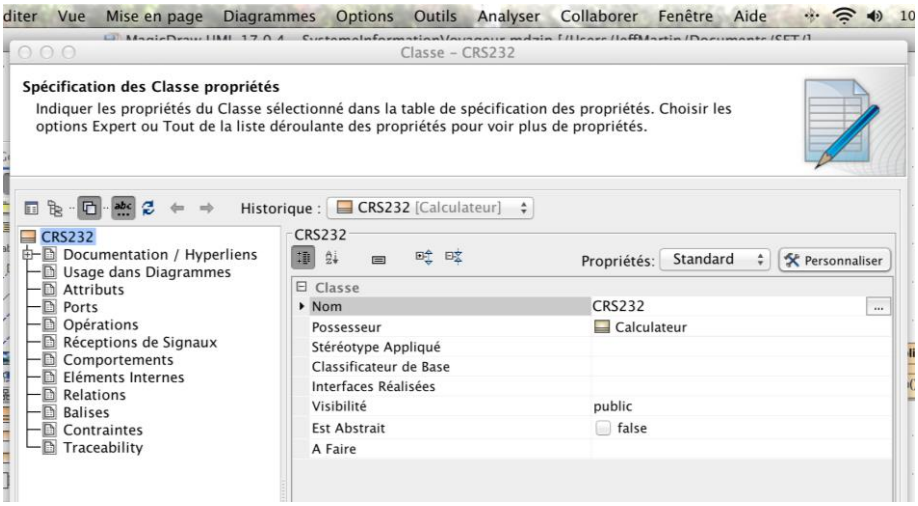

Figure 5 paramétrage de la classe CRS232

- Dans l'arborescence de la classe CRS232 développez la rubrique **Attributs** pour ouvrir une nouvelle fenêtre de dialogue.
- Effectuer un clic sur le bouton **Créer** pour ouvrir la fenêtre ***spécification des propriétés propriété***.et renseigner les différents champs de l'interface. Vous constatez que les différents attributs de la classe apparaissent avec leur propre spécificité.
- De la même manière que vous venez de déclarer les attributs vous allez déclarer les différentes méthodes de la classe CRS232. Recommencez la même séquence que pour la création des attributs mais cette fois ci en choisissant la rubrique **Opérations**
- Vous devez maintenant avoir un diagramme de classe semblable à celui de la figure 6

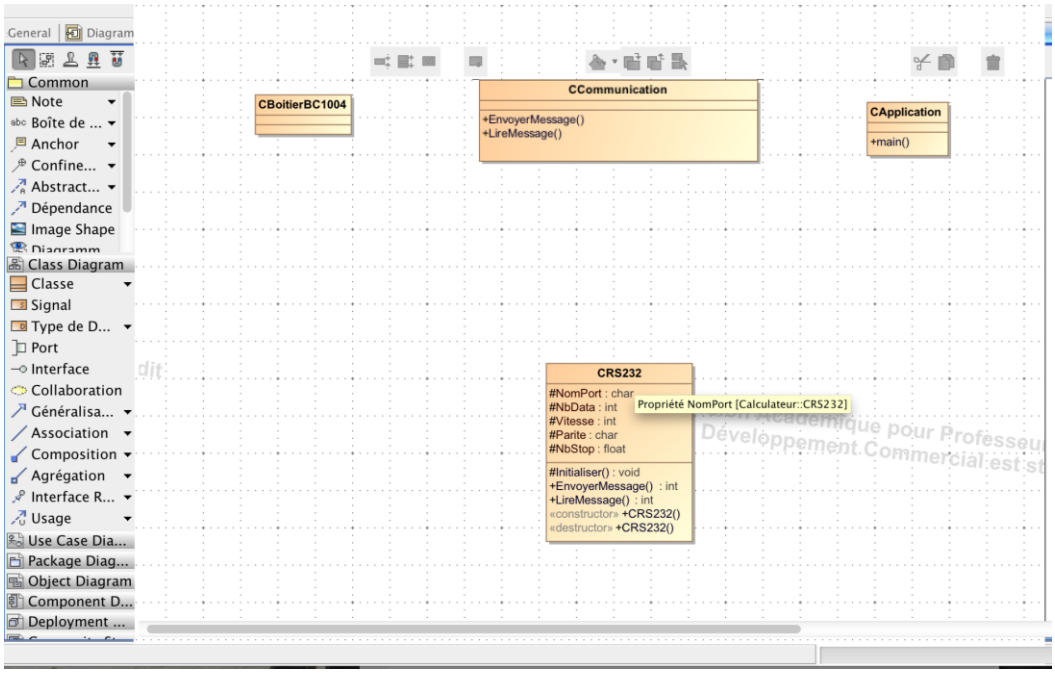

Figure 6 Evolution 1 du diagramme de classe

- Il ne nous reste plus maintenant qu'à indiquer que la classe CRS232 est une spécialisation de la classe CCommunication. Repérer dans les outils de construction du diagramme de classe le symbole représentant la spécialisation et ou généralisation et placer ce symbole sur votre diagramme entre les deux classes concernées. Attention la flèche indique la classe « mère ».

NOTE 5

En observant notre diagramme on constate que dans la classe CCommunication nous avons deux methodes EnvoyerMessage et LireMessage et l'on retrouve ces deux méthodes dans la classe CRS232

Ceci est normal mais la représentation n'est pas correcte. En effet de façon générique on peut dire que

quelque soit le support de communication utilisé il faudra envoyer le message et lire le message

La manière de lire le message sera spécialisée suivant le support utilisé. On ne lit pas de la même façon un message transporté par une liaison série et une liaison Ethernet.

Cette remarque nous oblige donc à **redéfinir la méthode** dans chaque **classe héritée** de la classe CCommunication.

On peut dire alors que les deux **méthodes** de la classe CCommunication sont **virtuelles** ceci nous oblige à le préciser dans la classe principale.

Mais comme il faut **spécialise**r ces deux méthodes pour **chaque support** il faut donc les **re-déclarer dans chaque classe héritée** ce qui nous oblige à dire que les deux **méthodes de la classe mère sont virtuelles pures** c'est à dire qu'au niveau du **code chaque méthode virtuelle pure sera égale à 0**

Cette déclaration particulière à pour conséquence que **la classe mère devient abstraite** et par conséquence **elle ne peut produire un objet** on dit qu'elle **n'est pas instanciable.**

- Ouvrir la fenêtre propriété de la classe CCommunication pour indiquer que la classe est virtuelle et indiquer également que les deux méthodes sont virtuelles. Vous constatez que le nom de la classe ainsi que les méthodes sont passées en italique.

3.1.3 FINIR LE DIAGRAMME DE CLASSE :

Maintenant que nous avons vu comment créer et paramétrer une classe et que nous avons observé le cas particulier des classes abstraites nous allons compléter le diagramme de classe pour finir la représentation de notre projet qui consiste à envoyer un message contenant les paramètres de ligne au boitier BC1004.

- Placer une association entre CBoitierBC1004 et CCommunication et une autre entre CCommunication et CApplication ; Noter que la première association est orientée de CCommunication vers CBoitierBC1004
- A l'aide de la boite de dialogue **Spécifications** nommer l'association indiquer les noms de rôle ainsi que les cardinalités vous devez avoir un diagramme semblable à celui de la figure 7

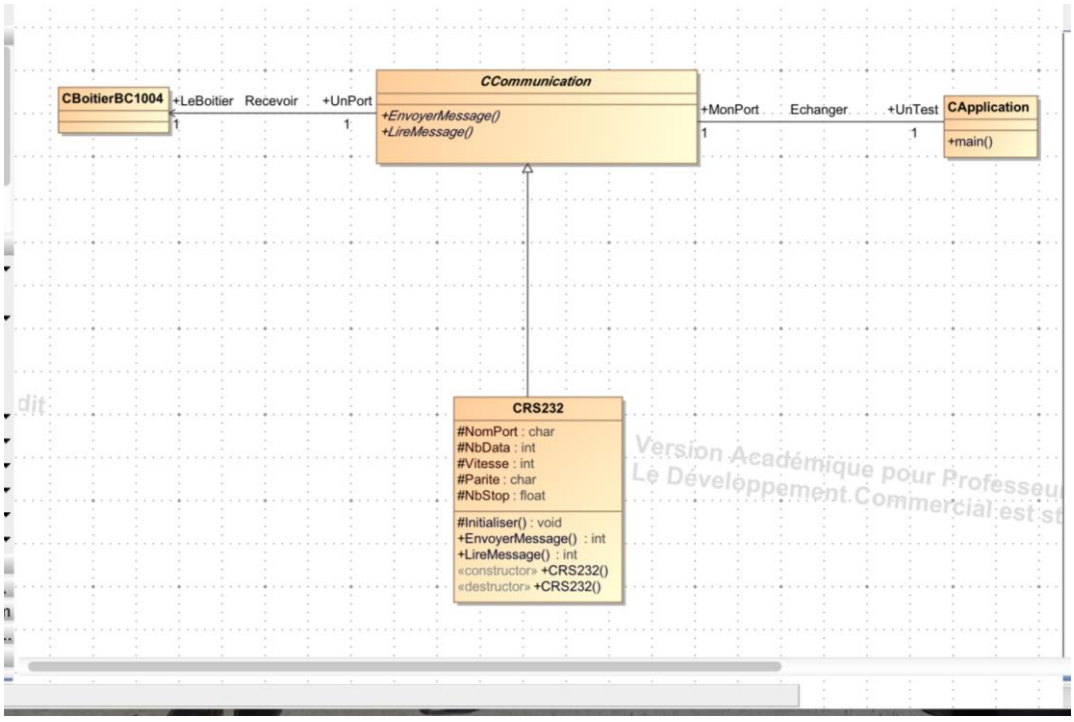

Figure 7 diagramme de classe de l'application Communication RS 232

3.1.4 PROGRAMMATION DE LA LIAISON RS232 SOUS WINDOWS :

Maintenant que nous avons déclaré les différents attributs et méthodes nécessaires à la programmation d'une liaison série il nous reste à adapter notre étude au système d'exploitation utilisé en effet il est construit de telle manière qu'il gère avec certaine particularité les différents ports de communication. Pour ce faire nous allons compléter notre code pour pouvoir enfin produire le code et tester notre application.

L'objectif n'est pas de compléter la classe CRS232 mais de spécialiser une classe Crs232Win qui reprendra alors toutes les caractéristiques de la classe mère à laquelle on ajoutera la spécificité de Windows

- Placer une nouvelle classe Crs232Win qui sera une spécialisation de CRS232.
- Windows gère ces ports de communication à l'aide d'identifiant qui correspond au processus en cours d'exécution puisque Windows travaille en multi tâche le type de IDCommunication est un HANDLE Créer un attribut IDCommunication de type HANDLE
- Windows a besoin d'une image de la configuration du port pour cela il utilise une structure de type DCB créer une variable dcb de typeDCB.

3.1.5 GENERATION DU CODE C++ DU DIAGRAMME DE CLASSE :

Maintenant que nous avons notre diagramme de classe il faut passer à l'écriture du code correspondant. Deux solutions s'offrent à nous soit tout écrire avec les risques d'erreurs soit utiliser la génération automatique du code correspondant. Bien entendu il restera à saisir le contenu des méthodes mais cela n'est pas un handicap important c'est vraiment votre travail.

- Lire le document TP UML .pdf qui vous guidera pas à pas pour générer le code C++ correspondant.
- Ajouter les éléments non précisés par la génération du code
- Tester.

3.1.6 CODE C# :

Après avoir testé votre application en C++ convertir votre code en C# pour l'exploiter directement à partir du calculateur SIV.

Un exemple de code C# vous est fourni en annexe.