D. Conception et codage au PA

D.1 Conception :

On donne le diagramme de classe en **annexe .**

Question D.1.1
Donnez la déclaration de la classe **CKaplan**.

- L'application utilise le **polymorphisme** et la classe **CGroupe** est une classe **abstraite**.

Question D.1.2
Indiquez à quoi reconnaît on que la classe **CGroupe** est abstraite et quelles sont les conséquences.

- On relève dans le constructeur de la classe **CPa** l'instruction :

```
this->nbGroupes = nbGroupes;
```

Question D.1.3
Expliquez le rôle du pointeur **this** et son intérêt ici.

- On s'intéresse au constructeur et méthodes de la classe **CKaplan** :

Soit le bloc d'instruction :

```
CGroupe *groupe = new CKaplan(i);
groupe->fixeDebit(debit);
groupe->getResultats(&debit, &puissance);
```

Question D.1.4
Indiquez dans le tableau du document réponse si les méthodes appelées sont celles de la classe **CGroupe** et/ou celles de la classe **CKaplan**.

Question D.1.5
Ajoutez dans la déclaration de la classe **CPa** le lien assurant l'implémentation d'un membre nommé **combinaison**.

- L'attribut **typeTurbine** de la classe **CGroupe** est **protégé** (protected). Il est initialisé par les constructeurs des classes **CKaplan, CBulbe & CHelice**.

Question D.1.6
Indiquez quelle est la différence de visibilité entre **private** et **protected**, au niveau d'objets de classes **CKaplan, CHelice ou CBulbe**.
Pourquoi l'attribut **typeTurbine** doit-il être **protected** ?

D.2 Codage:

- La structure des messages **tMessagePa** reçus par la classe **CPa** est définie par la déclaration ci-dessous :

```
const int NBGROUPES=6;
enum tOrig {PHV, OPERATEUR};

struct tPhv {
        bool debutOuFinPhase;
        int  debitNaturel;
        int  debitModulation;
    };
struct tOperateur {
        int etatGroupe [NBGROUPES];
    };

struct tMessagePa {
    tOrig origineModif;
    union {
        tPhv        messageDuPhv;
        tOperateur messageOperateur;
    };
} ;
```

Question D.2.1
Complétez la portion de code de la tâche **calcule** recevant les consignes de la tâche **scruteLigne** (classe **CPa**).

- Le compilateur 32 bits utilisé implémente les énumérations avec des entiers et produit des entiers de **4 octets**.

Question D.2.2
Indiquez la taille en octets de la structure **tMessagePa** en détaillant le calcul.

Question D.2.3
Indiquez l'instruction du **C++** qui permet de retrouver cette taille.

- Le problème posé est celui du **producteur – consommateur** : la messagerie utilisée par les fonctions **envoie** & **retire** est articulée autour d'un sémaphore de section critique **Mutex**, de 2 sémaphores de synchronisation **SemaVide** initialisé au nombre de messages maximum et **SemaPlein** initialisé à 0 et d'un buffer circulaire.

Version 1		Version 2		Version 3	
envoie ( )	retire ( )	envoie ( )	retire ( )	envoie ( )	retire ( )
P(SemaVide) **P (Mutex)** **enfile (elt)** **V(SemaPlein)** **V(Mutex)**	**P(SemaPlein)** **P(Mutex)** **elt = defile ( )** **V(SemaVide)** **V(Mutex)**	**P(SemaPlein)** **P (Mutex)** **enfile (elt)** **V(SemaVide)** **V(Mutex)**	**P(SemaVide)** **P(Mutex)** **elt = defile ( )** **V(SemaPlein)** **V(Mutex)**	**P (Mutex)** **P(SemaVide)** **enfile (elt)** **V(SemaPlein)** **V(Mutex)**	**P(Mutex)** **P(SemaPlein)** **elt = defile ( )** **V(SemaVide)** **V(Mutex)**

P(sema) signifie prendre le sémaphore **sema**.
V(sema) signifie libérer le sémaphore **sema**.

Question D.2.4
Indiquez quelle version de pseudo code pour les fonctions **envoie** & **retire** est la bonne en justifiant le dysfonctionnement des 2 autres.

Question D.2.5
Indiquez la valeur avec laquelle est initialisée le sémaphore d'exclusion mutuelle

- On donne des exemples des classes **C++** utiles en **annexe 5**.

- Les débits de référence à appliquer aux groupes et donnés en **annexe 2** sont chargés en mémoire depuis un fichier de type **texte** dans le membre **table** de l'objet **combinaison** dont on donne ci-dessous la déclaration ; celui-ci est un vecteur de Lignes.

```
const int NBGROUPES=6 ;
struct Ligne {
        int     debitGroupe[NBGROUPES];
        double puissance;
};

typedef std::vector<Ligne> Page; // table de lignes de combinaisons = 1 page

class CCombinaison
{
private :
        int             nbPages;    // nombre de pages du fichier des débits
        Page            *table;     // table de pages
.....
} ;
```

Question D.2.6
Complétez la méthode **chargesPages** de la classe **CCombinaison**.

La méthode **combinaisonAdjacente** permet à partir d'une combinaison courante de type **Ligne** (voir ci-dessus), de chercher une autre combinaison qui ne diffère que d'un groupe (voir **B2**) dans le membre **table** de l'objet **combinaison.**

Pour accéder au débit du groupe **noGroupe** de la ligne **noLigne** de la page **noPage**, on utilisera la notation :

table[**noPage**][**noLigne**].debitGroupe[**noGroupe**]...

Question D.2.7
Complétez la méthode **combinaisonAdjacente** de la classe **CCombinaison**.

E. COMMUNICATION ET RESEAUX

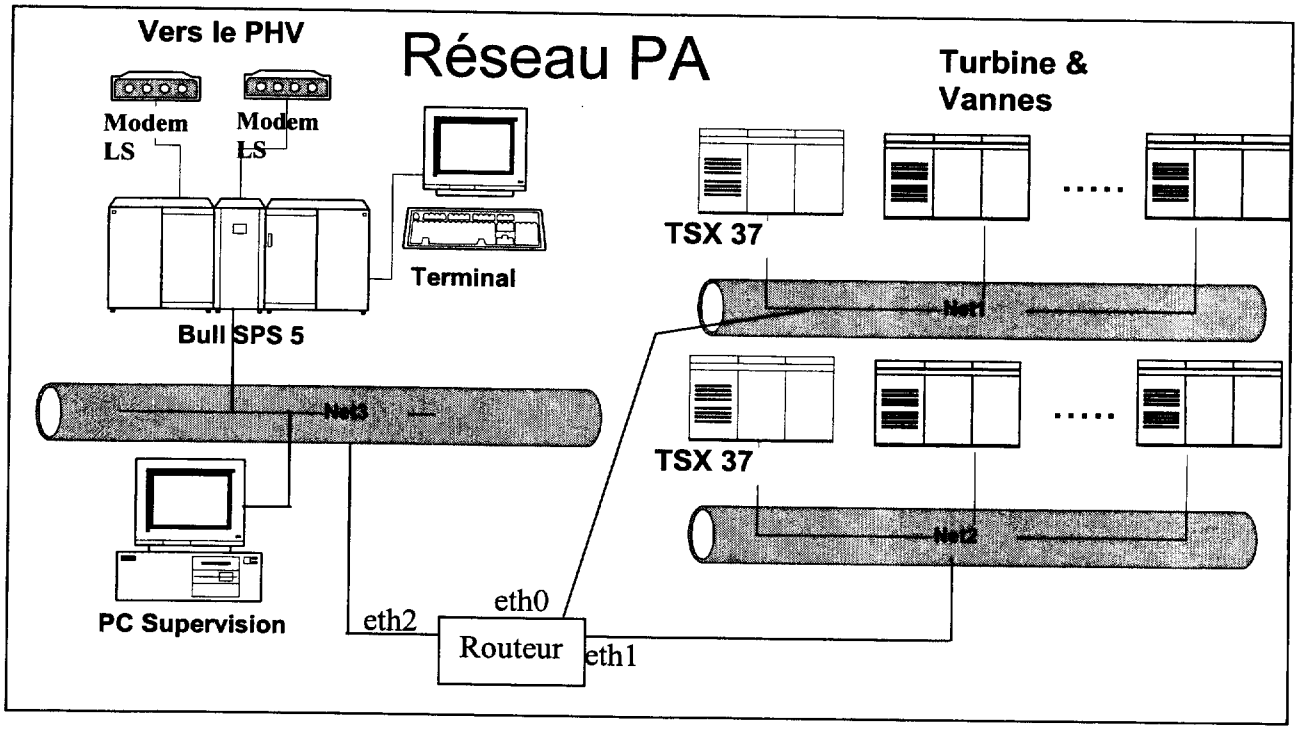
E.1 MODBUS.

Un réseau Modbus relie le calculateur BULL et les API. La liaison est de type série RS485.
Le BULL est maître du bus, les API en sont les esclaves.

Question E.1.1
Dans la configuration présente le réseau Modbus peut-il supporter le nombre d'esclaves ? Justifiez votre réponse.

E.2 Réseaux du P.A.

On désire remplacer le réseau Modbus par un réseau TCP/IP et la console de supervision par un ordinateur de type PC sur lequel se fera la supervision du système. L'architecture voulue du futur réseau est représentée sur la figure suivante.



Pour des raisons de sécurité, les automates sont dédoublés, ainsi chaque groupe est contrôlé par 2 automates, soit un total de 12 automates nommés API-1 à API-12. Le réseau est lui aussi dédoublé, les automates dont le numéro est impair sont connectés au réseau nommé "net1" et les automates dont le numéro est pair sont connectés au réseau nommé "net2". Le réseau "net3" relie le calculateur BULL et le PC de supervision. Les trois réseaux sont connectés entre eux par un routeur disposant de 3 interfaces réseaux (eth0, eth1 et eth2).

L' "adresse réseau" est **192.168.1.0**.

Question E.2.1
Quelle est la classe de cette adresse ? Justifiez votre réponse.
Pour cette adresse réseau, quelle est le "masque réseau" par défaut ?
Quelle est l'adresse de diffusion (broadcast) de ce réseau ?

Pour réaliser l'architecture réseau présentée ci-dessus, on crée 3 sous réseaux.

Question E.2.2
D'après le schéma réseau sur la page précédente, quel est le nombre de machines présentes sur le sous-réseau "net1" ?
En tenant compte de l'adresse réseau et de l'adresse Broadcast, combien de bits d'adresse seront nécessaires pour la partie "adresse machine" de l'adresse IP ?
Combien de bits d'adresse restent ils pour la partie "adresse sous-réseau" de l'adresse IP ?
Justifiez vos réponses.

On choisira finalement d'utiliser 3 bits pour l' "adresse sous réseau" et 5 bits pour l' "adresse machine".

Question E.2.3
Remplir le tableau du document réponse où :
- **adresse réseau** est l' "adresse réseau" du sous réseau
- **masque** est le "masque réseau" du sous réseau
- **broadcast** est l'adresse de diffusion du sous-réseau
- **adresse mini** est l'adresse minimum que peut avoir un équipement dans ce sous-réseau
- **adresse maxi** est l'adresse maximum que peut avoir un équipement dans ce sous-réseau

Question E.2.4
Proposer un plan d'adressage pour l'ensemble du réseau PA.

E.3 Composition des API

Les API sont composés de cartes :
- d'entrées et de sorties tout ou rien
- d'entrées et de sorties analogique
- de communication série RS485
- de communication Ethernet (TSX WMY 100)

Carte de communication Ethernet TSX WMY 100 permet :
- coordination entre automates programmables,
- supervision locale ou centralisée,
- communication avec l'informatique de gestion de production,
- communication avec des entrées/sorties distantes.
- messagerie UNI-TE avec l'ensemble de l'architecture X-WAY,
- messagerie Modbus.

Cette carte permet de gérer un site web embarqué qui comporte des applets java développés par Schneider et utilisables immédiatement. Le serveur web permet aussi de créer ses propres Applets.

L'étude porte sur un échanges entre un maître MODBUS (BULL) et un esclave MODBUS (API) (Trames ETHERNET TCP/IP MODBUS) :
Les trames Ethernet en annexe 6 ont été relevées à l'aide d'un logiciel de capture de trames:

Question E.3.1
Remplir les champs contenus dans le tableau du document réponse en vous aidant de l'échange « Réponse API →PC »

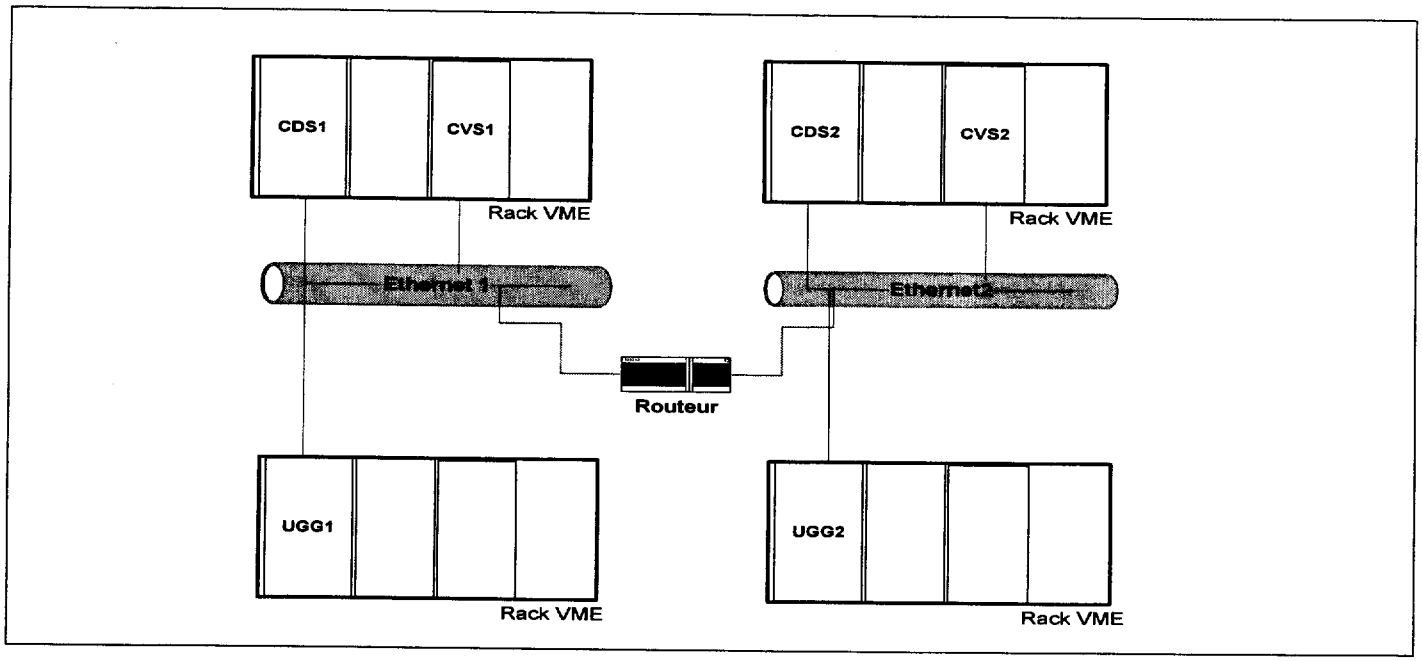
Question E.3.2
Remplir les champs contenus dans le tableau du document réponse en vous aidant des échanges PC→API et du protocole Modbus sur TCP/IP de l'annexe 6.

F. Architecture matérielle du PHV

Le PHV est constitué de :

- 2 serveurs (CDS1 et CDS2)
- 2 unités de gestion graphique (CVS1 et CVS2)
- 2 unités de gestion des liaisons (UGG1 et UGG2)

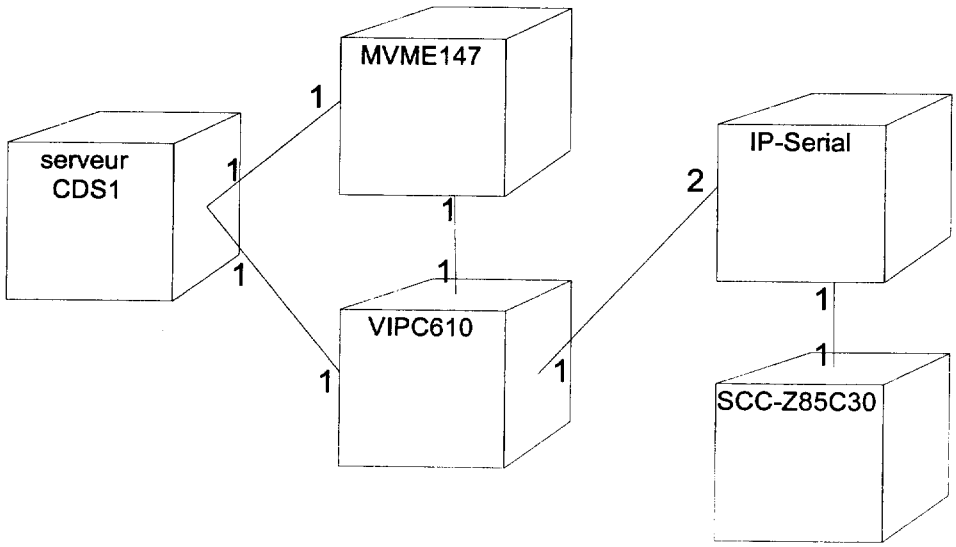
Ces 6 unités sont placées dans 4 racks VME disposés selon le schéma ci-dessous :



Dans cette partie, on ne s'intéressera qu'au serveur **CDS1**. Ce serveur est composé de :

- Une unité centrale : la **carte MVME147** (annexe 7)
- Une **carte VIPC610** (annexe 8) pouvant héberger 4 modules au standard "IndustryPack" (appelé **module IP**).
- 2 modules **IP-Serial** (annexe 9) nommé "IP-Serial-A" et "IP-Serial-B". Ces 2 modules sont installés sur la carte VIPC610. Le composant central de ce module est le **SCC-Z85C30** (annexe 10), ce circuit gère 2 ports séries. Cet ensemble permet au serveur CDS1 de disposer de 4 ports séries RS232 supplémentaires.

L'architecture du serveur CDS1 est décrite pas le diagramme de déploiement suivant :



F.1. Adressage des ports série du CDS1

Le Bus VME (normes IEEE1014) offre à une carte maître (ici la carte MVME147) la possibilité d'accéder aux registres d'une carte esclave (ici la VIPC610) via 3 espaces d'adressage :

Espace d'adressage	Nombre de bits d'adresse utilisés	Taille de l'espace adressable
SHORT I/O	16 (A00 à A15)*	64 ko
STANDARD	24 (A00 à A23)*	16 Mo
EXTENDED	32 (A00 à A31)*	4 Go

*: le bit d'adresse A00 n'est pas présent physiquement sur le bus.

La carte VIPC610 à été configurée pour n'être accessible que dans l'espace d'adressage "SHORT I/O". La configuration de son adresse de base se fait en plaçant ou en retirant des cavaliers ("shunts") sur le connecteur E3-E7 de la carte (voir figure 1 et 2 de l'annexe 8).

Question F.1.1
Indiquer sur le tableau du document réponse où il faut placer des cavaliers pour que l'adresse de base de la carte VIPC610 soit $F000 dans l'espace "SHORT I/O".

On place sur la carte VIPC610 les deux modules "IP-Serial". Le module "IP-Serial-A" est placé dans l'emplacement A de la carte VIPC610, le module "IP-Serial-B" est placé dans l'emplacement B.

Question F.1.2
Quelles sont les adresses de base des 2 modules "IP-Serial" ?

Chaque module "IP-Serial" est composé d'un circuit intégré SCC-Z85C30. L'annexe 9 nous informe que ce circuit est accessible par 4 registres nommés "Channel A, Control", " Channel A, Data", "Channel B, Control" et " Channel B, Data".

Question F.1.3
Quelle sont les adresses des registres du module "IP-Serial-A" dans l'espace "SHORT I/O"?

L'Application s'exécute sur la carte MVME147, le plan mémoire de cette carte est donnée dans l'annexe 7.

Question F.1.4
D'après cette annexe 7, indiquer la plage d'adresses de l'espace "SHORT I/O" du bus VME vue par une application s'exécutant sur la carte MVME147.

F.2. Configuration des ports série d'un module "IP-Serial"

Tous les ports séries seront configurés avec les paramètres suivants :

- 9600 bits/s
- 8 bits de données
- pas de bit de parité
- 1 bit "stop".

La configuration de ces paramètres se fait en écrivant des valeurs dans les registres WR0 à WR15 du SCC-Z85C30 (annexe 10). On accède à ces registres par le registre du contrôle du SCC-Z85C30 vu plus haut.
Dans les questions suivantes, on ne s'intéressera qu'aux valeurs qu'il faut écrire dans les registres WR0 à WR15, on ne s'intéressera pas à la manière d'y accéder.

La fréquence d'émission/réception (baud rate) se calcule grâce à la formule suivante:

TC = ClockFrequency / (2 x ClockMode x BaudRate) –2
ou :

- TC = Time Constante (à écrire dans les registres WR12 et WR13)
- ClockMode = Prédiviseur de fréquence.
- BaudRate = débit de transmission.

La fréquence d'horloge ("ClokckFrequency") reçu par le SCC-Z85C30 est de 3,6864 MHz.
Le prédiviseur ("ClockMode") se choisi parmi les valeurs 1, 16, 32, 64 grâce au registre WR4.

Question F.2.1
Le registre WR4 est initialisé avec la valeur $(00xxxxxx)_B$. En déduire la valeur de prédiviseur de fréquence.

Question F.2.2
Calculer la valeur de TC et en déduire la valeur qu'il faut écrire dans les registres WR12 et WR13.

Les autres paramètres de la liaison série se règlent par les registres WR4 et WR5.

Question F.2.3
Que faut-il écrire dans ces registres pour la configuration voulue (8bits de données, pas de parité, 1 bit stop) ?