BTS INFORMATIQUE ET RESEAUX POUR L' INDUSTRIE ET LES SERVICES TECHNIQUES

Session 2006

EPREUVE E.4
Etude d'un système informatisé

COMMANDE AUTOMATISEE DES CENTRALES HYDRAULIQUES DU RHIN

Document Réponse (12 pages)

B.1.1 Plages horaires =

Heure de début	Heure de fin	Nature de l'éclusée
00H00	07H00	Rétention
07H00	09H00	équilibre
09H00	14H00	Lâcher
14H00	17H30	équilibre
17H30	20H30	Lâcher
20H30		équilibre
	00H00	Rétention

B.1.2 Valeur du marnage = ...

B.1.3 Puissance électrique Kaplan = ...

B.2.1 Répartition choisie = ...
Nature combinaison = ...

B.2.2 Répartition choisie = ...
Nature combinaison = ...

B.2.3 Répartition choisie = ...
Nature combinaison = ...

B.2.4 Page chargée = ...
Répartition choisie = ...
Nature combinaison = ...

B.3.1 Caractéristiques générales :

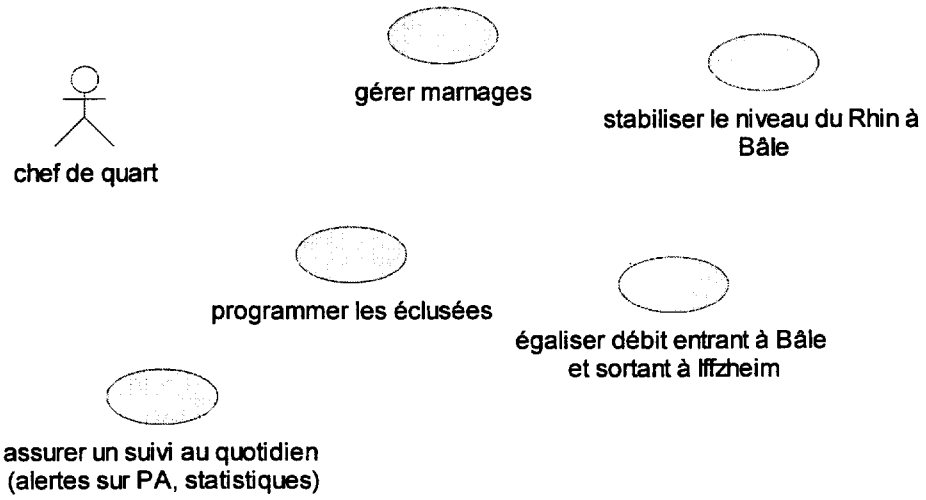
Avantages	Inconvénients

B.4.1 Tableau comparatif des différentes liaisons :
Cochez les bonnes réponses.

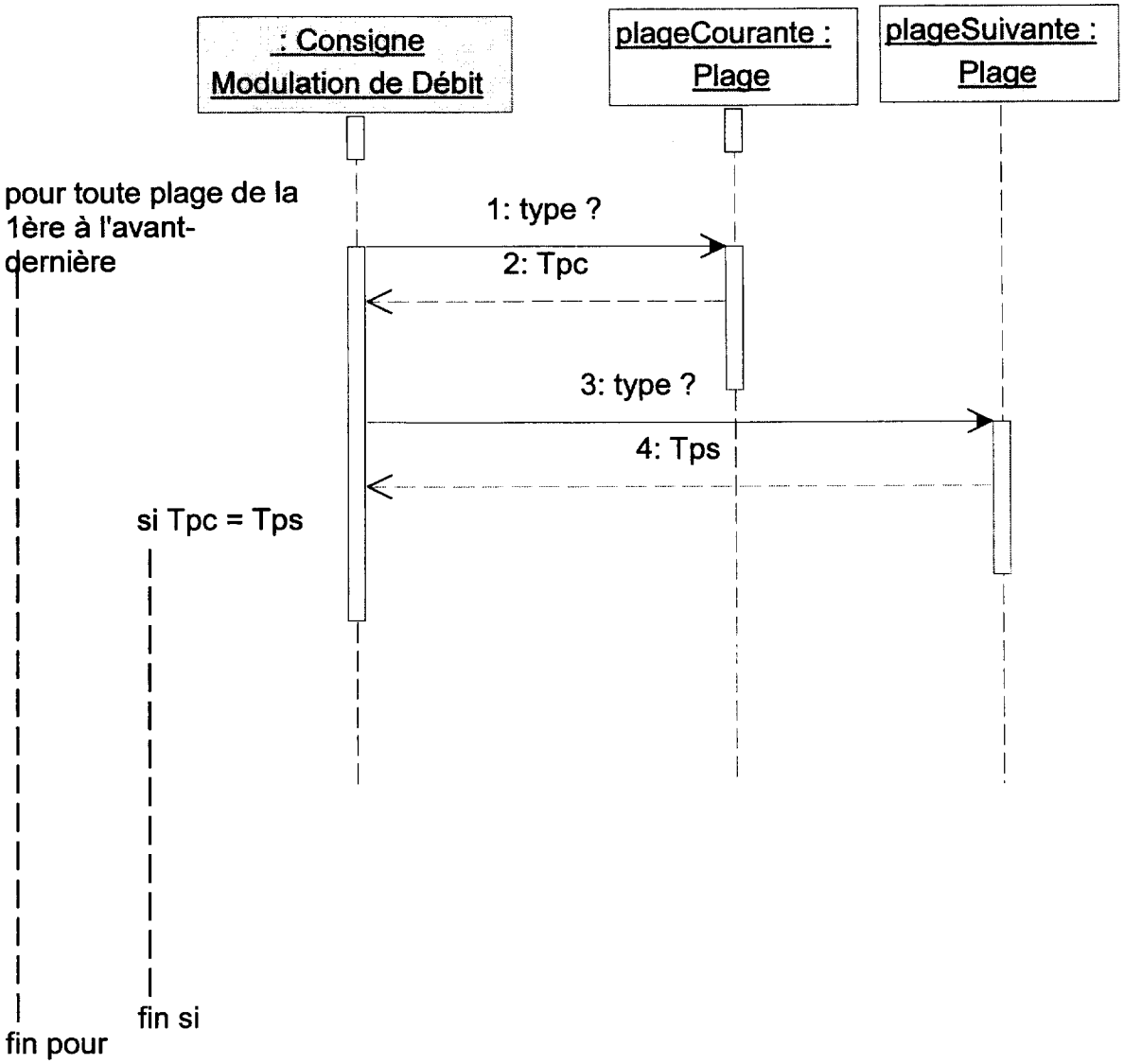
Liaisons	Topologie		Mode		Distance	
	Point à point	Multipoint	Différentiel	Unipolaire	< 100 m	> 1000 m
RS-232						
RS-422						
RS-485						

B.4.2 Temps de transmission = ...

C.2.1 Diagramme des Cas d'Utilisation :



C.3.1 Diagramme de Séquence (scénario de regroupement des plages) :



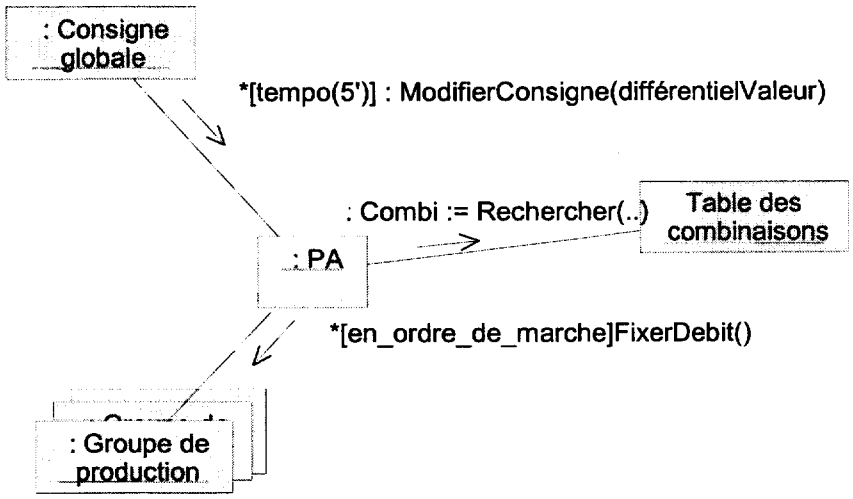
C.4.1 Nature synchrone/asynchrone des messages (scénario de régulation du débit) :

.....
.....
.....
.....
.....

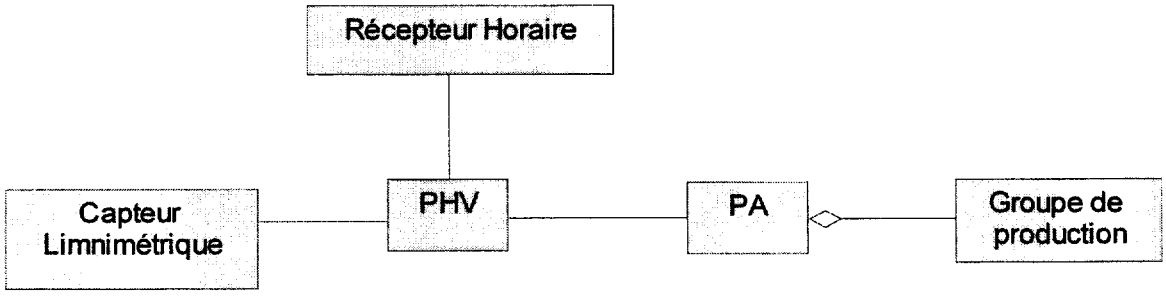
C.4.2 Signification du message *[en_ordre_de_marche]FixerDebit()

.....
.....
.....

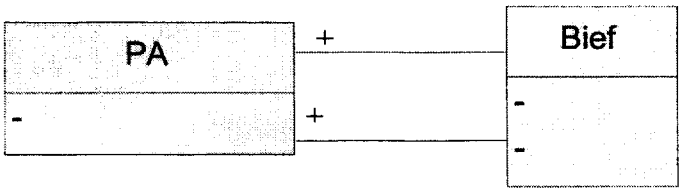
C.4.3 Numérotation des messages échangés par les objets



C.5.1 Indication des cardinalités sur Diagramme de Classes



C.5.2 Diagramme de Classes issu du Diagramme d'Objets



C.5.3 Nature de la relation entre les classes

Message, Consigne_débit_total et Informations_remontées :

.....
.....
.....

Cela signifie que ... (COCHER UNE CASE)

☐ **un message est spécialisable soit en une consigne de débit total, soit en une somme d'informations remontées ;**

☐ **une consigne de débit total correspond aux informations remontées;**

☐ **un message est composé à la fois d'une consigne de débit total et des informations remontées correspondantes.**

C.5.4 Une Consigne_débit_total est donc composée (au sens UML) de ...

.....
.....
.....

C.5.5 La contrainte *{ordonnée}* ne serait plus nécessaire si ...

.....
.....
.....

D.1.1 Déclaration de la classe CKaplan :

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

D.1.2 La classe CGroupe est abstraite car :

.....

Conséquences :

.....

D.1.3 Rôle du pointeur this :

.....

D.1.4 Cochez avec une croix les méthodes appelées :

Méthode appelée \ Méthode(s) exécutée(s)	Méthode de la classe **CGroupe**	Méthode de la classe **CKaplan**
Constructeur : CGroupe *groupe = new CKaplan(i);		
Méthode **fixeDebit : groupe->fixeDebit(debit);**		
Méthode **getResultats : groupe->getResultats(&debit, &puissance);**		

D.1.5 Lien entre la classe CPa et la classe CCombinaison:

```
class CPa
{
private:
  .....
        int              nbGroupes;
        CGroupe          **groupes;
        ........
public:
        CPa( CPhv *, int, int *, int);
        ~CPa(void);
        ..................
};
```

Réponse ici

D.1.6 Attribut privé (private) : .....

.....

Attribut protégé (protected) : .....

.....

Pourquoi l'attribut typeTurbine doit-il être protected ?

.....

.....

.....

D.2.1 Réception de la consigne du PHV :

```
while (1)
{
 tMessagePa * message = (tMessagePa *)messagerie.retirer();
 //***********************************************
 // Réception message consignes du PHV
 //***********************************************
 if (..... == PHV)
 {
   int debitConsigne =
          .....
       + .....;

   // les paramètres de la méthode sont la consigne globale et le booléen indiquant un début d'éclusée.
   combinaison.fixeDebit (debitConsigne,
              .....);

   Ligne c = combinaison.demandeCombinaison();
   repartitDebit(c);
  }
 //***********************************************
 // Réception message de l'opérateur du PA
 //***********************************************
      // ne rien mettre ici
} // fin while
```

D.2.2 Taille de la structure = ...

D.2.3 Instruction donnant la taille = ...

D.2.4 Version correcte = ...

Raisons pour lesquelles les autres versions sont incorrectes =

.....

.....

.....

.....

.....

.....

.....

D.2.5 Sémaphore d'exclusion mutuelle = ...

D.2.6 Méthode chargePages

```
bool CCombinaison::chargePages (void)
{
        ifstream   param;              // fichier des débits unitaires
        Ligne      ligne;              // une ligne de combinaison (débits unitaires + puissance)
        int        debit, nbLignes;    // info dans ligne entête de chaque page
        string     texteDebit, texteNbLignes; // labels dans ligne entête de chaque page
        bool       retcode = true ;

        //******** allocation mémoire pour table des débits  **********************
        table = new Page[nbPages] ;

        //******** Ouverture du fichier des débits **********************
        param.open(FICHIERS_PAGES) ;

        //******** Chargement des débits page par page **********************
        while(!param.eof())        {
                //******** Chargement   info & labels de la ligne entête *************
                param >> texteDebit >> debit >> texteNbLignes >> nbLignes;
                //******** Calcul du numéro de page  *************
                int noPage =  .....

                //******** Sortie méthode si fin de fichier inopinée *************
                if ..... {
                        retcode = false ;
                        break ;
                }

                //****** Sortie  méthode si les labels ne sont pas DEBITS & LIGNES **********
                if ..... {
                        retcode = false ;
                        break ;
                }
                for (int noLigne = 0; noLigne <  nbLignes; noLigne++)        {
                        //*** Récupération des débits groupes dans la ligne en cours ********
                        for (int noGroupe=0; noGroupe < NBGROUPES; noGroupe++)
                        //*** Récupération des quotas  la ligne en cours ********
                        .....
                        //*** Récupération de la puissance dans la ligne en cours ********
                        .....

                        //*** Stockage de la ligne dans la page  ********
                        table[noPage].push_back (ligne);
                }
        }
        //******** Fermeture du fichier des débits **********************
        .....
        return retcode;
}
```

D.2.7 Méthode combinaisonAdjacente

```
int CCombinaison::combinaisonAdjacente(int noPage, Ligne ligneCourante)
{

        // le nombre de lignes est donné par le vecteur
        for (unsigned int  noLigne=0; noLigne < ..... ;   noLigne++)
        {
                int nbAjoutsOuRetraits = 0;
                //*** Balayage des groupes ********
                for (int noGroupe=0; noGroupe < NBGROUPES; noGroupe++)
                {
                        if ( ( ligneCourante.debitGroupe[noGroupe] != 0
                           && .....)
                         || (.....
                           && .....) )
                                nbAjoutsOuRetraits++;
                }
                if ( ..... ) // c est une adjacente
                {
                        return noLigne;
                }
        }
        return -1;  // Pas de combinaison adjacente disponible
}
```

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

.....
.....
.....
.....
.....

E.2.1 Classe d'adresse : .....
Justification : .....
.....
.....
.....
Masque réseau : .....
Adresse de diffusion : .....

E.2.2 Nombre d'équipements sur net 1 : .....
Nombre de bits pour l'adresse équipements : .....
Nombre de bits pour l'adresse sous-réseau : .....
Justification : .....
.....
.....
.....
.....
.....

E.2.3 Sous Réseaux

Nom	Adresse Réseau	Masque	Broadcast	Adresse mini	Adresse maxi
net1					
net2					
net3					

E.2.4 Plan d'adressage

Equipement	Adresse IP	Equipement	Adresse IP
Router : eth0			
eth1			
eth2			
PC supervision			
BULL SPS 5			
API-1		API-2	
API-3		API-4	
API-5		API-6	
API-7		API-8	
API-9		API-10	
API-11		API-12	

E.3.1 Remplir les champs contenus dans le tableau du document suivant :

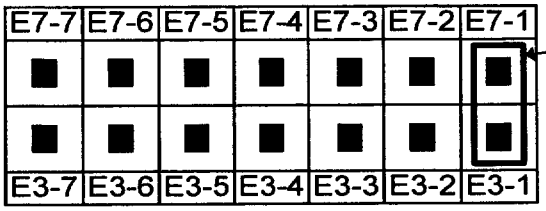
CHAMP (IP)	VALEUR	CHAMP (TCP)	VALEUR
Version :		Port Source :	
Type de service :		Port Destination :	
Identification :		Numéro d'ordre :	
Durée de vie :		Numéro d'accusé de réception :	
Protocole :		URG :	
Somme de contrôle de l'en-tête :		ACK :	
Adresse source:		PSH :	
Adresse destination :		FIN :	
Nombre d'octets que comporte le datagramme IP		Somme de contrôle :	
		Nombre d'octets que comporte le datagramme TCP	

E.3.2 Remplir les champs contenus dans le tableau du document suivant :

CHAMP (MODBUS)	VALEUR	CHAMP (MODBUS)	VALEUR	CHAMP (MODBUS)	VALEUR
Identificateur de transaction		Unit Identifier		Nombre de mots lus	
Identificateur de protocole		Code requête		Numéro du mot lu	
Longueur					

F.1.1 Adresse de base de la carte VIPC610:

Cavaliers sur le connecteur E3-E7 :



note : ce cavalier n'intervient pas dans le choix de l'adresse de base

F.1.2 Adresse de base du module "IP-Serial-A" dans l'emplacement A : .....

.....

Adresse de base du module "IP-Serial-B" dans l'emplacement B : .....

.....

F.1.3 Adresse des registres :

"IP-Serial-A", Channel A, registre Control :.....

"IP-Serial-A", Channel A, registre Data : .....

"IP-Serial-A", Channel B, registre Control : .....

"IP-Serial-A", Channel B, registre Data : .....

F.1.4 Plage d'adresses de l'espace "SHORT I/O" du bus VME vue depuis la carte MVME147 : .....

F.2.1 Prédiviseur de fréquence : .....

F.2.2 Time Constant TC = ...

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
WR12								
WR13								

F.2.3 Registres WR4 et WR5 :

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
WR4	0	0	X	X				
WR5	X			X	X	X	X	X