IREDSI

BTS INFORMATIQUE INDUSTRIELLE

SESSION 2000

EPREUVE

Etude d'un Système Informatisé

DEUXIEME PARTIE

CONCEPTION

Ce document comprend 19 pages.

Sujet : pages 1 à 16 (papier blanc)
Documents réponses : pages 17 à 19 (papier couleur) **(à rendre avec la copie)**

DEUXIEME PARTIE : CONCEPTION du SYSTEME

Durée 4h 30mn **Coefficient 3.5/5**

SUJET :

AVERTISSEMENT AU CANDIDAT

- Cette seconde partie de l'épreuve se divise en cinq sous-parties indépendantes (Questions 2 à 6).
- Le paragraphe 1 ne comporte pas de questions mais sa lecture est indispensable.

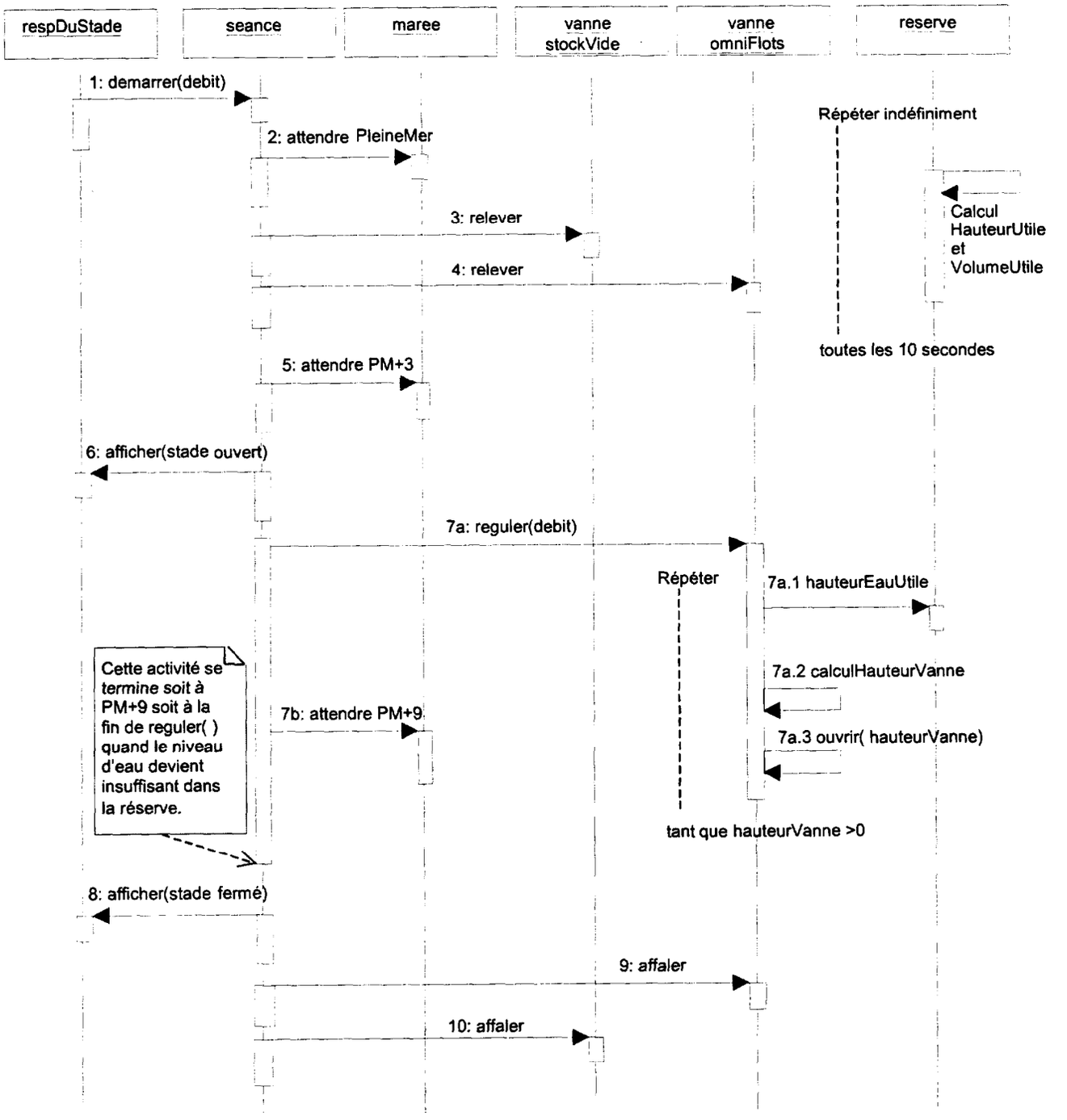
BAREME indicatif (/70 points):

QUESTION 2		QUESTION 3		QUESTION 4					QUESTION 5			QUESTION 6	
2.1	2.2	3.1	3.2	4.1	4.2	4.3	4.4	4.5	5.1	5.2	5.3	6.1	6.2
5 pts	5 pts	5 pts	5 pts	4 pts	4pts	4 pts	4 pts	9 pts	5 pts	4 pts	6 pts	3 pts	7 pts
10 pts		**10 pts**		**25 pts**					**15 pts**			**10 pts**	

1 CHOIX DE CONCEPTION

1.1 Diagramme de séquence

Le diagramme de séquence suivant représente un **exemple**, volontairement simplifié pour l'examen, d'une séance d'utilisation du stade d'eau vive. Les procédures de sécurité, entre autres, ne sont pas abordées, celles de dialogue avec le responsable du stade sont limitées au strict minimum.



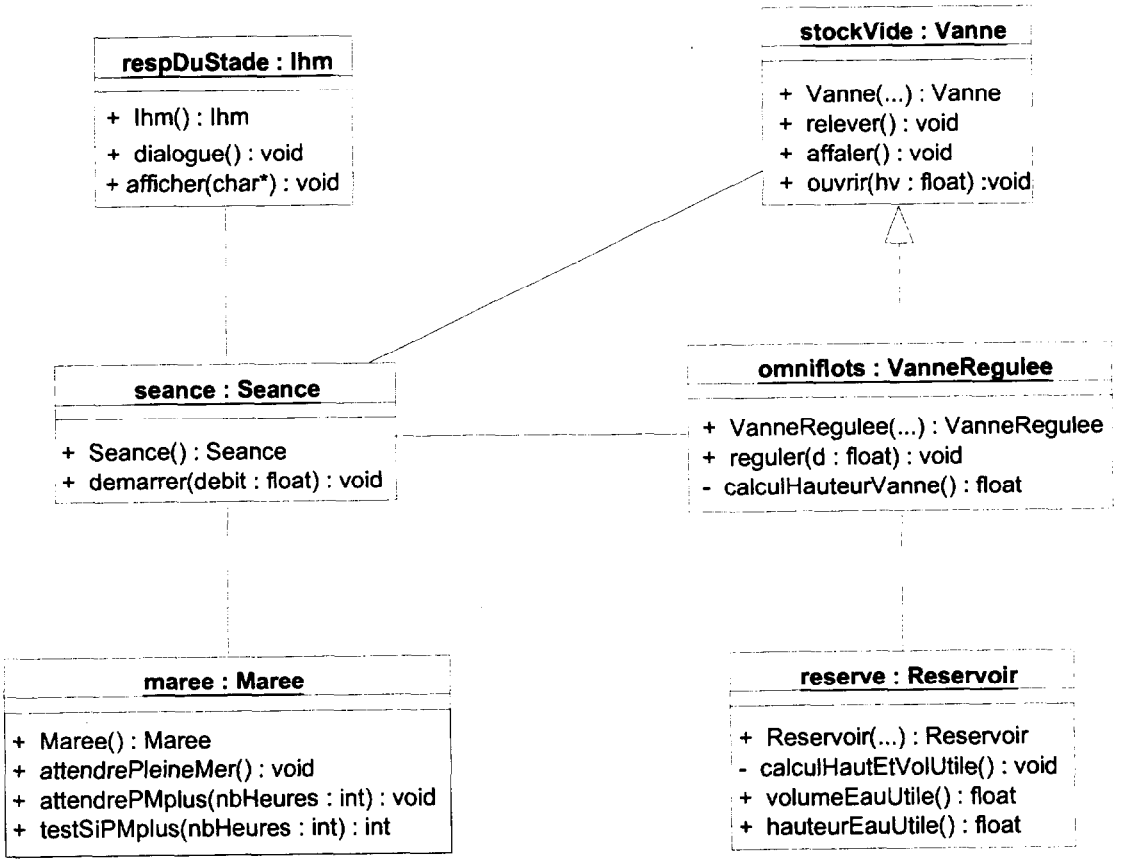
Nota:

→ Message synchrone caractérisé par le fait que l'objet émetteur attend une réponse de l'objet destinataire pour continuer son activité.

7a, 7b : Messages concurrents dans l'activité 7.
7a.1, 7a.2, 7a.3 : Messages consécutifs dans l'activité 7a.

1.2 Diagramme de structure statique

A ce stade de la conception, on décide d'adopter le diagramme de structure statique suivante. D'autres choix étaient bien sur envisageables, mais on impose celui ci pour la suite de l'étude.

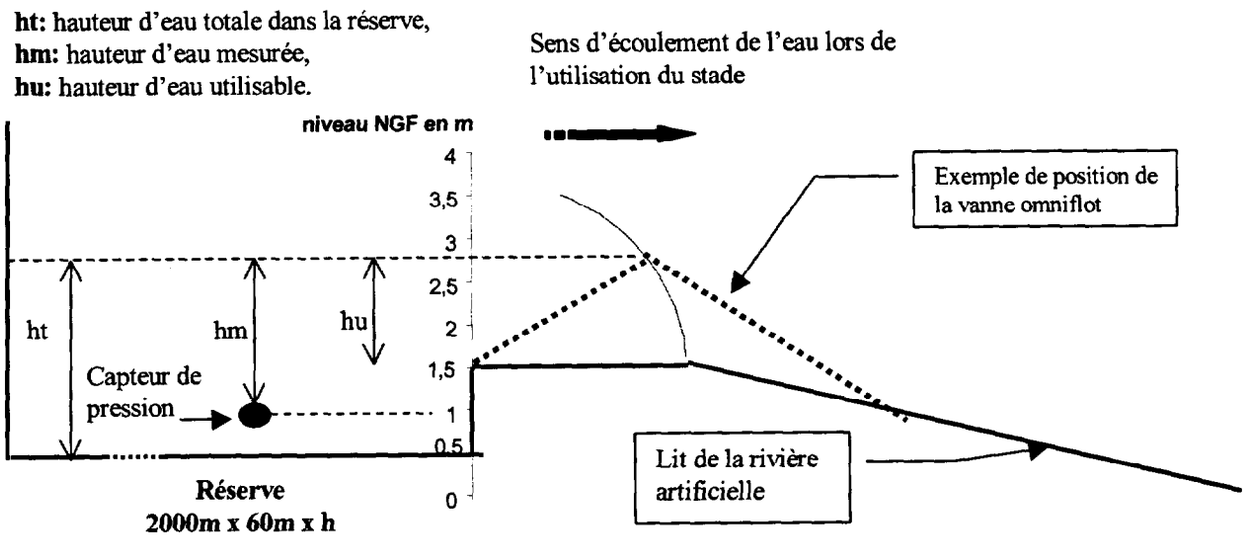


Nota : Certaines classes peuvent posséder des opérations qui ne sont pas utilisées dans le *cas d'utilisation* étudié dans ce sujet.

1.3 Description de la réserve

La **réserve** est considérée comme un parallélépipède rectangle de surface 2000m sur 60m et dont le fond se situe à une hauteur NGF de 0,5m.

Pour déterminer le volume d'eau disponible on mesure le niveau de l'eau dans la réserve. Pour cela, on utilise un capteur de pression hydrostatique qui est placé à 50 cm du fond (hauteur NGF=1 m).



Capteur de pression utilisé: Marque ROGERT référence xy22

Il est constitué d'un boîtier cylindrique en acier inoxydable contenant la membrane de mesure, un condensateur à armature cylindrique et un préamplificateur.

La pression hydrostatique de l'eau est convertie en une variation de capacité (via le déplacement mécanique de la membrane). Le préamplificateur utilise cette variation pour délivrer en sortie une tension analogique proportionelle à la hauteur **hm** de la colonne d'eau placée au dessus de ce capteur.

Un dispositif compensateur corrige l'influence dues aux variations de pression atmosphérique.

Caractéristiques:

- Echelle de mesure (après réglage et étalonnage) : hauteur **hm** de 0 à 5m.
- Tension de sortie correspondante: 0 à 10 volts.
- Ecart de linéarité <1%, reproductibilité <0,2%, dérive du zéro <0,5%
- Température d'utilisation de 0 à 50°C, classe de protection IP 68.

1.4 Description des vannes

La vanne *omniflots* de largeur 7 m et de hauteur 2m20 est commandée par un vérin hydraulique pouvant transmettre une force de 500 000N avec une pression maximum de 250 bars. La sortie de la tige de ce vérin est commandée par un régulateur hydraulique qui assure une ouverture ou fermeture progressive de la vanne (la vitesse normale d'entrée ou de sortie de la tige du vérin est de 0,25 $m.mn^{-1}$).

La commande de ce régulateur nécessite une tension de consigne comprise entre 0v (tige rentrée) et *tensionMax* = +10 volts (tige sortie). La **position** de la tige du vérin est **proportionelle** à cette **consigne**.

Un capteur potentiométrique, intégré au vérin, permet à tout moment de connaître la position du vérin et de savoir si le mouvement demandé est bien réalisé. Ce capteur, qui nécessite une alimentation de 10 volts, délivre une tension proportionelle à la sortie de la tige du vérin : 0v ➔ tige rentrée, +10 volts ➔ tige sortie.

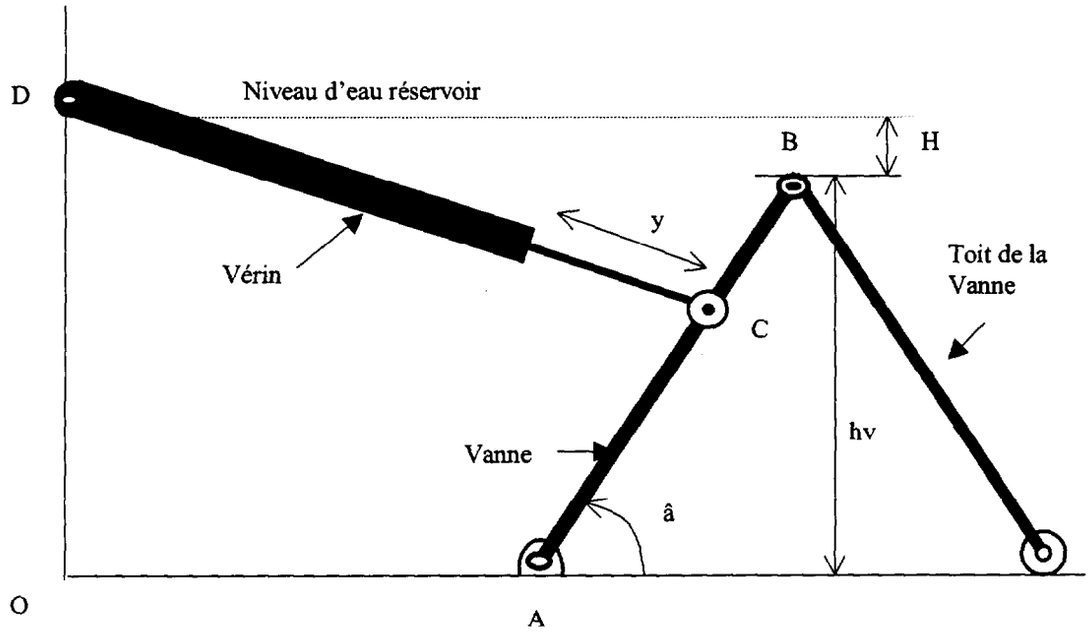
De plus pour des questions de sécurité, deux détecteurs de proximité type Tout Ou Rien, moulés dans le vérin, permettent de savoir si le vérin arrive en butée (*captTigeSortie* ou *captTigeRentree*).

Caractéristiques de la vanne Omniflots:

Données: $AB = 2{,}20m$ $AC = 1{,}85m$ $OA = 1{,}95m$ $OD = 2{,}30m$

2m80 (lgVérinTigeRentrée = *lgMin*) < DC < 4m 45 (lgVérinTigeSortie = *lgMax*).

Course du vérin: y maxi (*course*) = 1m 65.



Pour pouvoir régler le débit d'eau qui dépend de la hauteur de la lame d'eau **H**, il est nécessaire d'exprimer la **sortie de tige du vérin y** en fonction de la **hauteur hv** souhaitée pour la vanne.

$$y = \sqrt{\left(\left(OA + AC \,.\, \left(\sqrt{AB^2 - hv^2} \,/\, AB \right) \right)^2 + \left(OD - AC \,.\, \left(hv \,/\, AB \right) \right)^2 \right)} - lgVérinTigeRentrée$$

Inversement, pour connaître la position de la vanne en fonction de la valeur transmise par le capteur potentiométrique, il faut exprimer **hv** en fonction de **y**.

$$hv = AB \,.\left(\left(\sqrt{X^2.OD^2 - (OD^2 + OA^2).(X^2 - OA^2)} - X \,.OD \right) / (OA^2 + OD^2) \right)$$

avec $X = \left(\left(y + lgVérinTigeRentrée \right)^2 - OA^2 - OD^2 - AC^2 \right) / 2 \,.\, AC$

La commande de la vanne *stockVide* utilise le même principe, à la différence près que sa largeur est de 20m.

1.5 Interfaçage avec le micro-ordinateur industriel

Les différents actionneurs et capteurs seront gérés par un module IP ADA08 (dossier technique pages 30 à 50) monté sur le canal B de la carte unité centrale LX162 (dossier technique 5.3 page 22).

Raccordement utilisé: (Voir page 47 du dossier technique)

Réserve	**Fonction**	**Canal**	**N° broches**
Capteur Niveau Eau	CAN	Voie 0 EVI0 (0v à +10volts)	2 , 1 (AGND)
Vanne omniflots			
Commande régulateur hyraulique	CNA	Voie 0 EVO0 (0v à +10volts)	12, 11(AGND)
Alimentation capteur potentiométrique	CNA	Voie 2 EVO2 (0v à +10volts)	14, 11(AGND)
Curseur capteur potentiométrique	CAN	Voie 2 EVI0 (0v à +10volts)	4 , 11 (AGND)
ButéeTigeSortie	PORT A	PA0 (0v si repos, 5v si action)	32, 31(GND)
ButéeTigeRentrée	PORT A	PA2 (0v si repos, 5v si action)	34, 31(GND)
Vanne stockVide			
Commande régulateur hyraulique	CNA	Voie 4 EVO4 (0v à +10volts)	16, 11(AGND)
Alimentation capteur potentiométrique	CNA	Voie 6 EVO6 (0v à +10volts)	18, 11(AGND)
Curseur capteur potentiométrique	CAN	Voie 6 EVI6 (0v à +10volts)	8 , 11 (AGND)
ButéeTigeSortie	PORT A	PA4 (0v si repos, 5v si action)	36, 31(GND)
ButéeTigeRentrée	PORT A	PA6 (0v si repos, 5v si action)	38, 31(GND)

2 Questions « Configuration du matériel»

2.1 Configuration IPIC (Annexe 5.5 page 26 du dossier technique)

Un circuit Asic, composant programmable, nommé « IPIC » réalise l'interface entre le module IP et les bus internes de la carte LX 162 (voir « Lx162 Block Diagram » page 23 du dossier technique).

Sachant que le module ADA08:

- est monté sur le canal IP B de la carte Lx162 ;
- peut, par sa fonction *Port*, demander une interruption sensible à un niveau bas sur la broche *IntReq0* et que cette demande doit être transmise avec un niveau 4 à l'unité de traitement .
- nécessite un « *Recovery-time* » de 2 microsecondes.

Indiquer les registres de l'IPIC (noms et adresses) à configurer ainsi que les valeurs, justifiées, à charger dans ces registres.

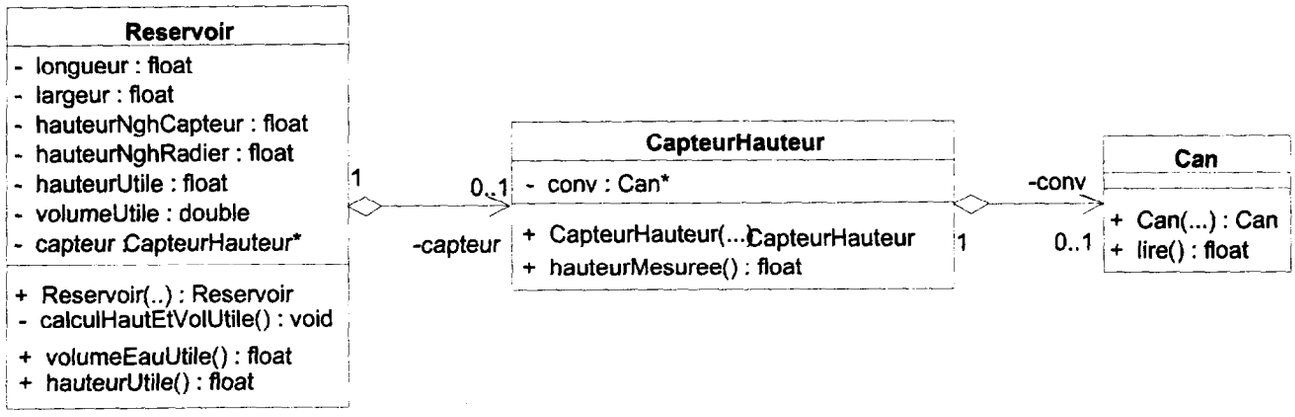
2.2 Adressage ADA08 (Annexe 5.6 du dossier technique)

En fonction des configurations ci-dessus et de la documentation du module ADA08 (pages 34 et 35 du dossier technique), **indiquer les adresses complètes** (sur 32bits):

- de base du module ADA 08 ;
- de base des circuits MP7613, LM12458 et CIOZ8536 ;
- du registre «DACREG » du MP7613.
- du registre «DAC code» du MP7613 permettant de fixer la tension d'alimentation du capteur potentiométrique de la vanne *omniflots*.

3 Questions « Objet réserve »

Pour la mise en œuvre de l'objet « *réserve* », on décide d'élaborer une classe « Réservoir » dont on instanciera un objet avec les caractéristiques spécifiques imposées par l'application.



La classe *« Réservoir »* utilise un capteur de type *« CapteurHauteur »* pour connaître la hauteur d'eau **hm** en mètres. Ce capteur utilise un *« Can »* pour connaître la tension en volts délivrée par le capteur de pression placé dans la *« réserve »*.

Nota : Dans ce diagramme de classe, les attributs privés *capteur* de la classe *Reservoir* et *conv* de la classe *Can* ont été représentés deux fois (une fois en attribut de classe, une fois comme nom de l'agrégation). Il est bien entendu qu'il n'existe à chaque fois qu'un seul attribut.

La réalisation des questions suivantes nécessite la lecture de la description de la réserve (paragraphe 1.3 page 4).

3.1 CapteurHauteur : : hauteurMesuree()

Sachant que la classe *Can* est disponible et que son opération *lire( )* retourne un réel exprimant la tension en volts appliquée à l'entrée analogique correspondante, **réaliser le codage en C++ de l'opération** *CapteurHauteur : : hauteurMesuree( ) : float* qui doit retourner un réel exprimant la hauteur d'eau mesurée (**hm**) en mètres.

3.2 Reservoir : : calculHautEtVolUtile()

Afin que le niveau d'eau mesuré ne soit pas perturbé par le clapot et la houle, on effectue 20 mesures sur une période de 10 secondes et on moyenne ces résultats pour obtenir le niveau d'eau réel.

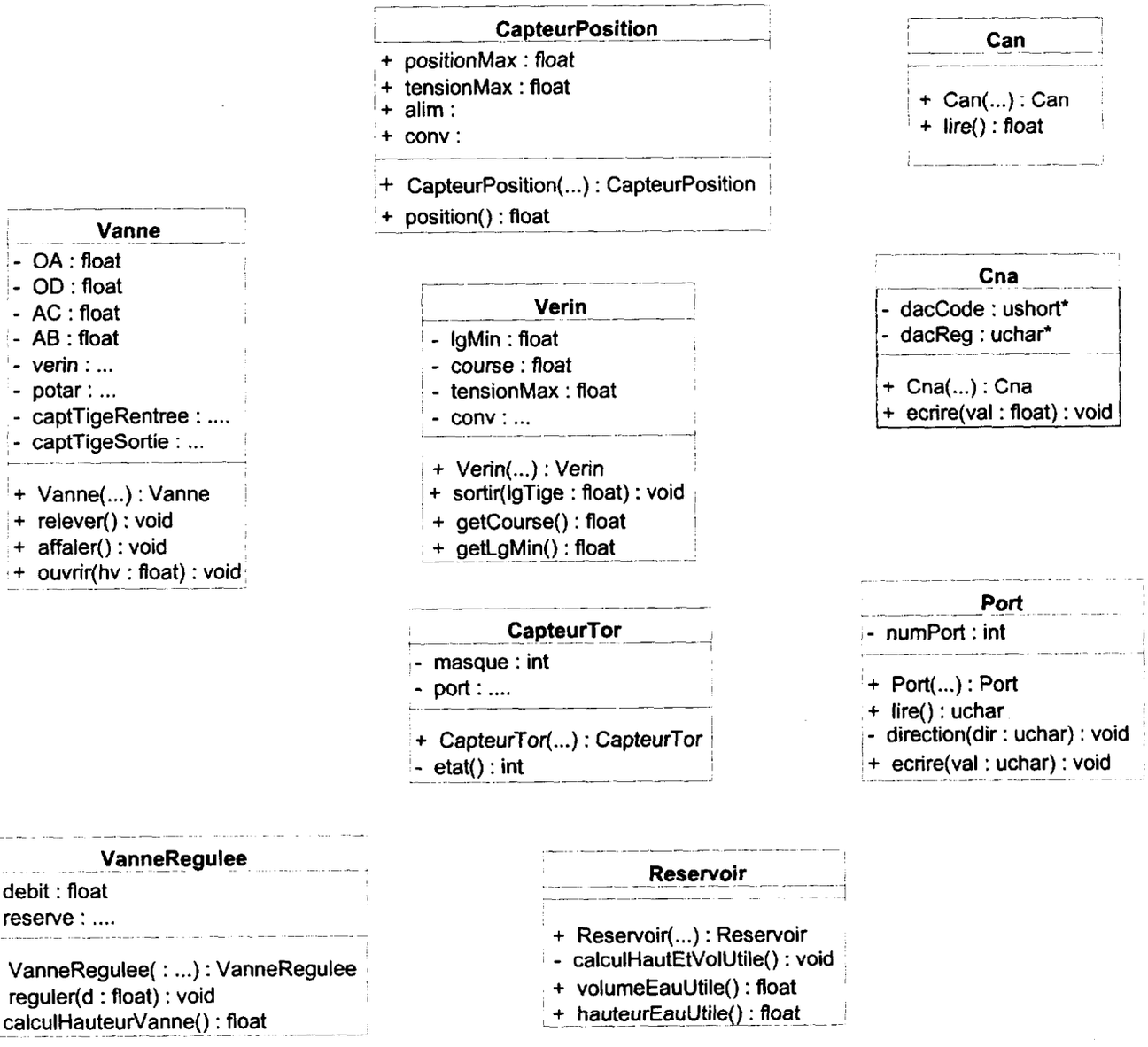
Les attributs privés de *« Réservoir »* ayant été correctement initialisés par son constructeur, on demande de **réaliser le codage C++ de l'opération** *« calculHautEtVolUtile »* qui doit, toutes les 10 secondes, mettre à jour les attributs *« hauteurUtile »* et *« volumeUtile »*.

Ces attributs doivent contenir le niveau d'eau **hu** par rapport au radier et le volume d'eau (en m^3) utilisable dans le réservoir (longueur * largeur * hu).

Nota : On pourra utiliser une méthode statique de la classe « *Thread* » du noyau *NTR++* (dossier technique paragraphe 5.1 page 14) pour effectuer la temporisation entre deux mesures.

4 Questions « objets vannes »

Pour la mise en œuvre des objets *« vanne stokVide »* et *« vanne omniflots »* on décide d'élaborer une classe de base *« Vanne »* disposant des opérations *relever( ), affaler( )* et *ouvrir( )* et une classe plus élaborée *« VanneRegulee »* qui permet en plus d'obtenir un débit (en $m^3.s^{-1}$) régulé en fonction d'une consigne et du niveau d'eau dans la *« réserve »*.



4.1 Diagramme de classe

Sachant :

- que chaque vanne utilise un vérin, un capteur de position potentiométrique et deux détecteurs de proximité,
- que ces différents objets sont interfacés par des composants *cna, can* et *port* disponibles sur le module d'entrées sorties ADA08,

représenter en utilisant le langage UML les relations (association, agrégation, héritage, ..) entre ces différentes classes.

On précisera, si nécessaire, leur nom et la multiplicité et on commentera, brièvement, les choix effectués.

(Utiliser le document réponse 1 page 17 en fin de sujet.)

4.2 Test de fonctionnement du Cna MP7613

On désire vérifier le bon fonctionnement d'un module Cna MP7613 en utilisant un *metteur au point (debugger).*

- **Quelle tension délivre ce convertisseur** si on écrit la valeur 0x8003 dans le registre de données correspondant (*DAC code 0*)? (documentation MP7613 de ADA08 pages 43 et 44 dossier technique).
- **Quelle valeur hexadécimale** doit on écrire dans ce même registre pour obtenir en sortie une tension de –5 volts?

4.3 Cna : : ecrire()

En vous aidant du diagramme de classe de la page précédente et de la documentation sur le composant MP7613 de ADA08 (dossier technique pages 43 et 44) :

réaliser le codage en C++ de l'opération *Cna: :ecrire(val :float) : void* qui doit, à partir de val (tension en volts), piloter le convertisseur numérique analogique MP7613 correspondant.

4.4 Vanne : : ouvrir()

Réaliser le codage en C++ de l'opération *Vanne : : ouvrir(hv :float) : void* qui doit permettre d'ouvrir la vanne à une hauteur **hv** (voir description de la vanne paragraphe 1.4 page 5 de ce document). Après avoir commandé le régulateur hydraulique du vérin, on doit vérifier que le mouvement demandé soit terminé avant de quitter cette opération.

Nota : Les attributs privés utilisés dans les classes *Vanne* et *Verin* sont définis, *en italique*, dans la description des vannes paragraphe 1.4 page 5 de ce document.

4.5 Classe VanneRegulee

L'étude de la régulation du débit de la rivière, a produit l'équation suivante :

$Q = m . L . H . \sqrt{2 . g . H}$ d'après le théorème de Bernouilli

avec : $m = 0{,}525 .(1 + 1 / (1000.H + 1.6)).(1 + 0{,}55 . H^2 / (H + L)^2)$ d'après une formule empirique.

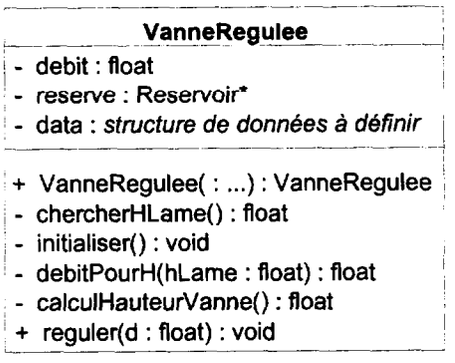
L : largeur en m de la vanne ,
H : hauteur de la lame d'eau au dessus du sommet de la vanne en m
g : accélération de la pesanteur en $m.s^{-2}$,
Q : débit en $m^3 . s^{-1}$

On constate que connaissant H il est assez facile d'obtenir Q, par contre exprimer H en fonction de Q semble mathématiquement plus ardu.

Comme la régulation de débit nécessite de déterminer H à partir de Q, on se propose de calculer par avance tous les couples (Q,H) pour les débits compris entre 4 $m^3 s^{-1}$ et 14 $m^3 s^{-1}$ par pas de 0.1 $m^3 s^{-1}$. La précision sur les débits doit être de ± 0,05 $m^3 s^{-1}$.

Les informations nécessaires seront stockées dans une structure de données.

On considère qu'il **existe** une opération privée ***debitPourH*** qui permet de calculer le débit Q en fonction de la hauteur H. Le prototype UML de cette opération est ***debitPourH(hLame : float) : float***



4.5.1 **Proposer une structure de données** permettant de stocker les informations précédentes.

Nota : L'algorithme suivant n'est pas indispensable pour répondre aux questions 4.5.3 et 4.5.4

4.5.2 **Ecrire l'algorithme** détaillé de l'opération privée ***initialiser( )*** permettant de remplir la structure de données.

4.5.3 **Par quelle opération publique** de la classe *VanneRegulee* sera appelée cette opération ***initialiser( )*** ?

4.5.4 **Ecrire l'algorithme détaillé** de l'opération privée ***chercherHLame( )*** permettant de retourner la hauteur H de la lame d'eau correspondant au débit fixé en attribut.

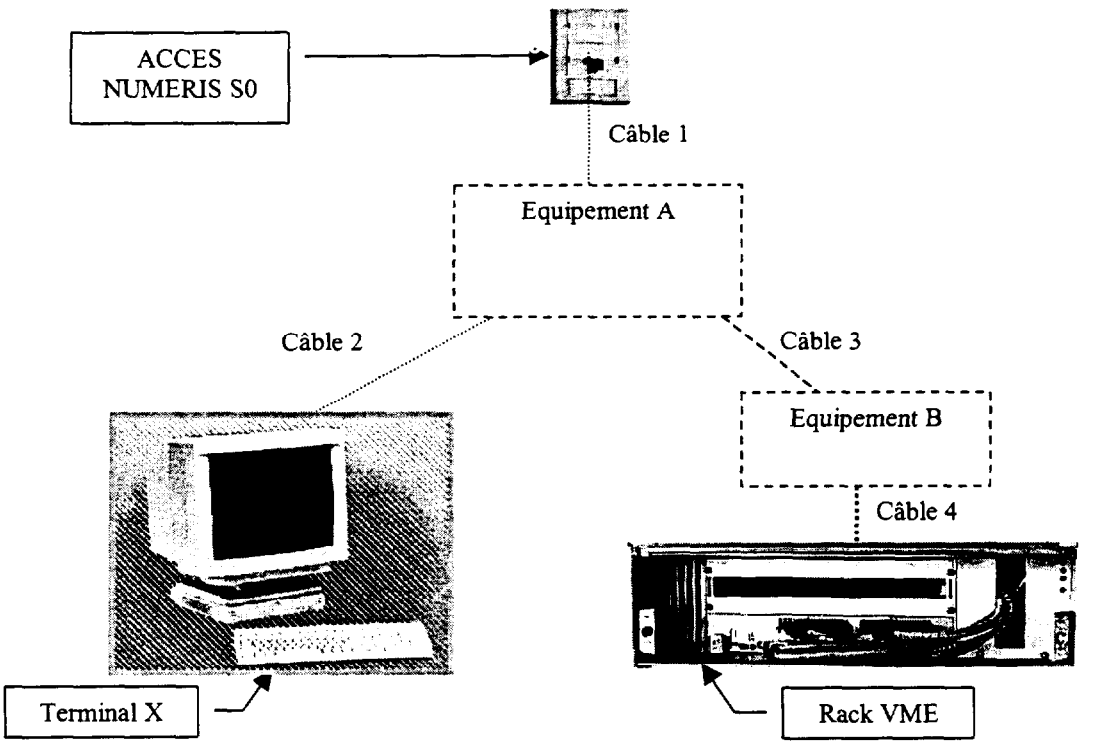
5 QUESTIONS « OBJET MAREE »

L'objet *« marée »* doit nécessairement connaître les horaires de marée. Pour cela, on décide d'interroger le serveur Web du Service Hydrographique et Océanographique de la Marine (SHOM). Ce service public, est responsable sur le plan national de l'information nautique : collecte, validation, diffusion des informations utiles aux navigateurs, civils ou militaires, professionnels ou plaisanciers. Son serveur Web (http://www.shom.fr) fournit, entre autres, les horaires de pleines mers et de basses mers du jour.

5.1 Choix des matériels réseaux nécessaires

On désire interconnecter les dispositifs suivants (documentations en annexe):

- Rack VME contenant la carte CPU LX 162 muni d'une interface Ethernet 10 Mbits/s AUI
- Terminal X de type Explora 450 muni d'une interface Ethernet 10/100 BT
- Accès S0 Numéris (il permet l'accès Internet et donc au serveur du Shom)



Choisir dans les documentations fournies (annexe 7.2 pages 15 et 16) les matériels nécessaires à l'interconnexion de ces différentes machines. On remplira le tableau du document réponse 2 fourni page 18.

5.2 Plan d'adressage IP

Le prestataire de service, choisi pour l'accès à Internet, attribue à notre site l'adresse IP 194.234.125.1
Les adresses des autres équipements doivent être en conformité avec la RFC 1918 (annexe 7.2.3 page 16).
Indiquer sur le schéma (document réponse 2 page 18), pour chaque équipement concerné, le plan d'adressage IP que vous proposez.

5.3 Etude de la communication avec le serveur du Shom

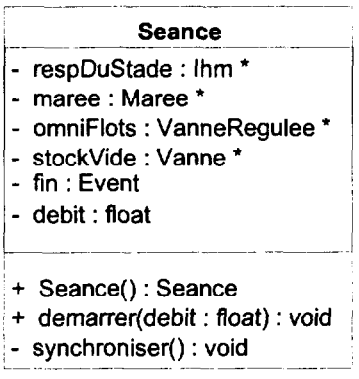
Pour tester la communication avec le serveur du Shom, on utilise un navigateur du commerce et un analyseur réseau pour se connecter au site *http://www.shom.fr*. On obtient les trames décrites en annexe 7.1 (pages 13, ...).

En utilisant la documentation HTTP (dossier technique paragraphe 5.2 page 18) et le document réponse 3 (page 19), **analyser les trames relevées lors de cet échange entre le client (navigateur) et le serveur (SHOM).**

6 Question « classe Seance »

6.1 Seance : : demarrer( )

En utilisant la documentation sur le noyau temps réel NTR++ (dossier technique 5.1 page 14) et en respectant les diagrammes de séquences et de classes (paragraphes 1.1 et 1.2 pages 2 et 3 de ce document) , **réaliser le codage en C++ de l'opération :** *Seance : : demarrer(debit :float) :void* qui doit créer un Thread exécutant le code de la méthode privée *synchroniser( ).*



6.2 Seance : : synchroniser( )

Réaliser le codage de l'opération privée *Seance : : synchroniser( ) : void* qui permet l'enchaînement de l'ensemble des activités de l'objet *« seance »* décrites dans le diagramme de séquences (paragraphe 1.1 page 2 de ce document).

On portera un soin particulier au fait que cette opération doit à un moment donné se mettre en attente :

- **soit de la fin de l'opération *maree : : attendrePMplus(9),***
- **soit de la fin de l'opération de la vanne *omniflots : : reguler(debit)***

Nota:

- La classe *Seance* dispose des attributs définis dans le diagramme ci-dessus. Ces attributs sont initialisés par son constructeur.
- Vous avez le droit de créer une (ou des) méthode(s) privée(s) supplémentaire(s).
- Vous avez le droit de créer un ou plusieurs objets du noyau NTR++ (Event, Thread, ...).