

# C A P E T

CONCOURS EXTERNE

Section : GENIE ELECTRIQUE  
Option : Informatique et Télématique

**ETUDE D'UN SYSTEME  
et/ou  
D'UN PROCESSUS TECHNIQUE**

**CORRIGE**

**+ BAREME**

## BAREME:

Réponse 1	a	1 pt
	b	2 pts
	c	1 pt
Réponse 2	a	1 pt
	b	1 pt
Réponse 3	a	1 pt
	b	1 pt
Réponse 4		5 pts
Réponse 5	a	1 pt
	b	3 pts
	c	1 pt
Réponse 6	a	1 pt
	b	1 pt
Réponse 7		3 pts
Réponse 8		8 pts
Réponse 9	a	1 pt
	b	1 pt
	c	2 pts
Réponse 10	a	1 pt
	b	1 pt
	c	2 pts
Réponse 11		2 pts
Réponse 12	a	1 pt
	b	1 pt
	c	1 pt
Réponse 13	a	1 pt
	b	1 pt
Réponse 14	a	1 pt
	b	1 pt
Réponse 15	a	1 pt
	b	3 pts
Réponse 16	a	2 pts
	b	2 pts
	c	2 pts
	d	9 pts
Réponse 17		8 pts
Réponse 18	a	5 pts
	b	3 pts
	c	2 pts
Réponse 19	a	1 pt
	b	2 pts
Réponse 20	a	10 pts
	b	2 pts

**Réponse 1:**

a) L'outil de communication est le PIPE.

b) Déclaration:

```
#include <string.h>
#include <stdio.h>

struct SAutomate {
    int adresse ;
    char typeDefault ;
    char date[10] ;
    int niveauUrgence ;
};
typedef struct SAutomate defaultAutomateT ;

struct SDecalage {
    char typeDefault ;
    char date[10] ;
    int niveauUrgence ;
};
typedef struct SDecalage defaultDecalT ;

struct SProcess {
    char typeDefault ;
    int numZone ;
    char date[10] ;
    int niveauUrgence ;
};
typedef struct SProcess defaultProcessT ;

struct SEtain {
    char typeDefault ;
    char date[10] ;
};
typedef struct SEtain defaultEtainT ;

struct SDefault {
    int origine ;
    union Default {
        defaultAutomateT dAut ;
        defaultDecalT dDec ;
        defaultProcessT dPro ;
        defaultEtainT dEtain ;
    } UDefault ;
};
typedef struct SDefault defaultGenereT ;
```

```
//exemple d'utilisation
void main() {
    defGenereT de,
                defGenere ;
    de.origine = 4 ;
    de.UDefaut.dEtain.typeDefaut = '2' ;
    strcpy(de.UDefaut.dEtain.date, "abc") ;
    defGenere = de ;
    if(defGenere.origine == 4) {
        printf("Defaut etain \n") ;
        printf("%c ",defGenere.UDefaut.dEtain.typeDefaut) ;
        printf("%s ",defGenere.UDefaut.dEtain.date) ;
    }
}
```

- c) L'identification de l'origine du défaut se fera sur le test du champ origine de la structure Sdefaut.

### Réponse 2:

- a) Couche Physique: assure le transfert des éléments binaires des trames sur le support électrique.  
 Couche Liaison: la couche MAC gère les accès réseau par la méthode CSMA/CD. La fonction de la couche LLC est reportée en partie sur la couche transport.  
 Couche Réseau: IP Internet Protocol. Acheminement des données regroupées en paquets à travers le réseau.  
 Couche Transport: contrôle le transfert des informations de l'émetteur vers le récepteur. UDP et TCP se situent à ce niveau.  
 Couche Application: Messageries, transfert de fichier, terminal virtuel, ...
- b) Le mode connecté (TCP) assure un fonctionnement plus fiable. Il est à utiliser lorsque la quantité de données est importante.

### Réponse 3:

Le fichier /etc/hosts interrogé par la fonction gethostbyname par exemple, pour obtenir des informations nécessaires sur la machine distante.  
 Le fichier /etc/networks interrogé par la fonction getnetbyname par exemple pour obtenir des informations nécessaires si plusieurs réseaux sont reliés.  
 Le fichier /etc/services constituant une base de données répertoriant et associant les numéros de port connus des protocoles de transport. La fonction getservbyname permet de consulter ce fichier.  
 Le fichier /etc/protocol contient les informations relatives aux différents protocoles Internet connus. La fonction getprotobyname utilise ce fichier.

## Réponse 4:

```

/*****
* NOM      : clientTcp.c
* TYPE     : APPLICATION
* SUJET    : Squelette d'un client TCP
*****/
*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>
int creer_socket(int, int *, struct sockaddr_in *);
struct sockaddr_in adresseServeur,
                 adresseClient ;

main(int argc, char * argv[]) {
/*
* ENTREES      : nom de la machine serveur, numéro de port de service
* SORTIES      : néant
* E/S          : néant
* RESULTAT     : retourne un descripteur correspondant à la socket créée
* DESCRIPTION  :
*
* GLOBALES    : adresse
*/

int port,
    socketClient ;
struct hostent * hp ;
/* contrôle du nombre des paramètres
*/
if(argc < 3) {
    fprintf(stderr, "erreur sur le nombre de paramètres \n");
    exit(2);
}
/* recherche de l'adresse Internet de la machine serveur
*/
if((hp = gethostbyname(argv[1])) == NULL) {
    fprintf(stderr, "machine %s inconnue \n", argv[1]);
    exit(2);
}
/* création et attachement de la socket client sur un port
*/
port = atoi(argv[2]);
if((socketClient = creer_socket(SOCK_STREAM, &port, &adresseClient)) == -1) {
    fprintf(stderr, "Création socket client impossible \n");
    exit(2);
}
/* préparation de l'adresse du serveur
*/
adresseServeur.sin_family = AF_INET ;
adresseServeur.sin_port = htons(atoi(argv[2]));
memcpy(&adresseServeur.sin_addr.s_addr, hp->h_addr, hp->h_length);
/* demande de connexion au serveur
*/
if(connect(socketClient, &adresseServeur, sizeof(adresseServeur)) == -1) {
    perror(" erreur de connect _n");
    exit(2);
}
/* zone de travail du programme client
*/
close(socketClient);
}
/***** Fin du Fichier clientTcp.c *****/
*/

```

```

/*****
* NOM      : serveurTcp.c
* TYPE     : APPLICATION
* SUJET    : Squelette d'un serveur TCP
*****/
*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

int creer_socket(int, int *, struct sockaddr_in *);
main(int argc, char * argv[]) {
/*
* ENTREES      : numéro de port de service
* SORTIES      : néant
* E/S          : néant
* RESULTAT     : retourne un descripteur correspondant à la socket créée
* DESCRIPTION  :
*
* GLOBALES    : adresse
*/
int port,
    lgAdresse,
    descSocket,
    socketService ;
struct sockaddr_in adresse ;

/* contrôle du nombre des paramètres
*/
if(argc < 2) {
    fprintf(stderr, "erreur sur le nombre de paramètres \n");
    exit(2);
}

/* création et attachement de la socket client sur un port
*/
port = atoi(argv[1]);
if((descSocket = creer_socket(SOCK_STREAM, &port, &adresse)) == -1) {
    fprintf(stderr, "Création socket client impossible \n");
    exit(2);
}
/* déclaration d'ouverture de service
*/
if(listen(descSocket, 1) == -1) {
    perror("listen erreur \n");
    exit(2);
}
/* attente de connexion
*/
socketService = accept(descSocket, &adresse, &lgAdresse);
close(descSocket);
/* zone de travail du programme serveur
*/
close(socketService);
}
/***** Fin du Fichier serveurTcp.c *****/
*/

```

**Réponse 5:**

- a) Détachement du terminal: création d'une nouvelle session (setsid) sans être leader d'un groupe.
- b) Après acceptation d'une connexion le serveur se décharge sur un processus fils qui va gérer la gestion du dialogue avec le client. Le serveur est à l'écoute d'une nouvelle demande de connexion.
- c) Il est nécessaire de prévoir la prise en compte de la terminaison du processus fils qui aura lieu lorsque la connexion sera fermée (sinon saturation de la table des processus).

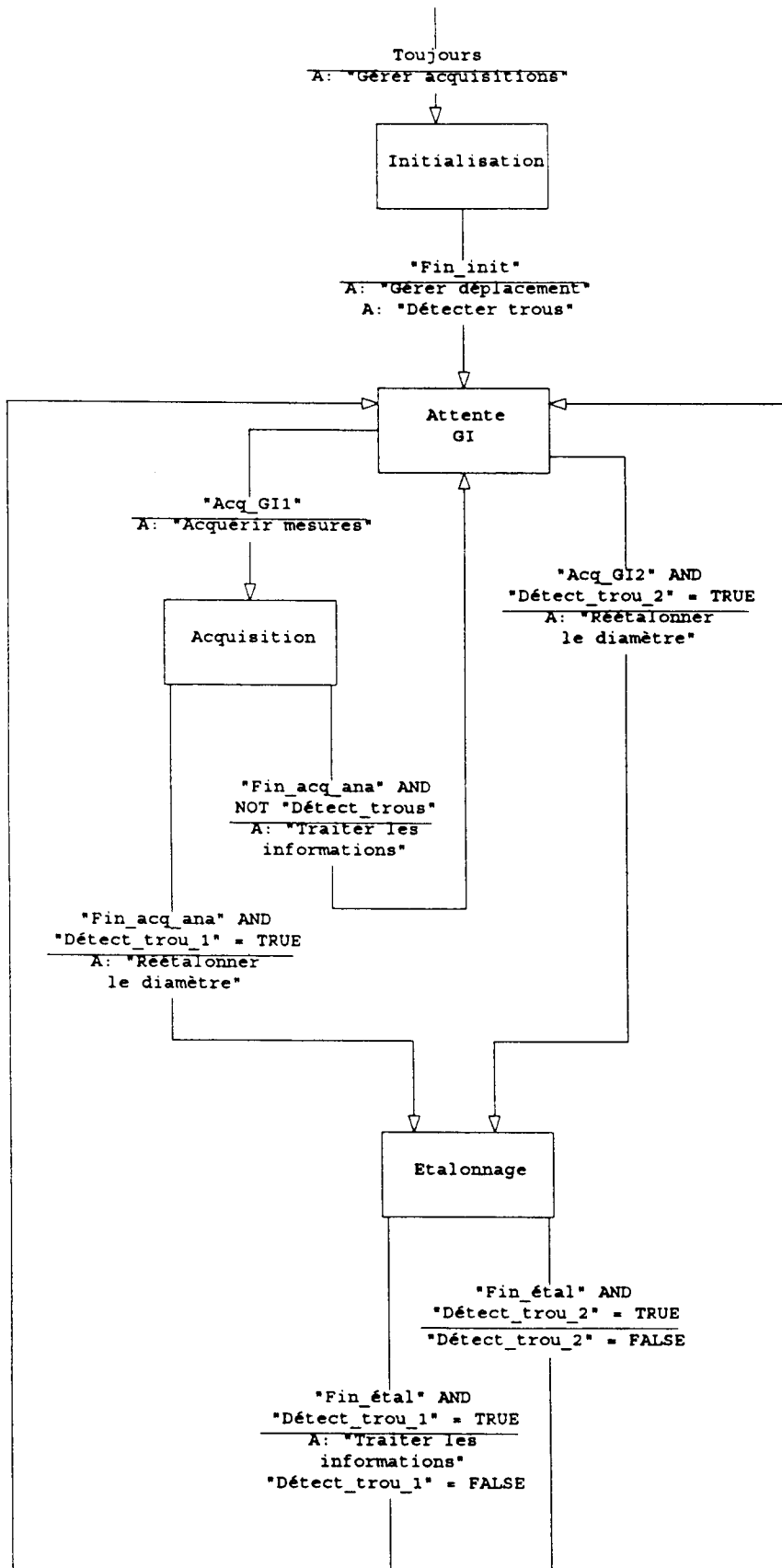
**Réponse 6:**

- a) `servgestion stream tcp nowait root /etc/servgestiond servgestiond`
- b) La boucle `while(TRUE)` de notre serveur monopolise des ressources. Le serveur `inetd` prend en charge l'attente sur les différents ports associés aux services en utilisant la primitive `select`. Au niveau du programme serveur une modification doit être apportée. La socket obtenue au résultat de `accept` est considérée comme l'entrée standard et toutes les opérations sur la socket doivent utiliser ce descripteur.

**Réponse 7:**

Nom du contrôle	Type interne/externe	Origine	Activ. Déclenche. Inhibe
Fin_acq_ana	Interne	Acquérir mesures	A Traiter les informations si pas Délect trous A Réétalonner le diamètre si Délect trou 1 = TRUE
Detect trous	Interne	Délecter trous	
Fin étal	Interne	Réétalonner le diamètre	A Traiter les informations
Acq_G11	Interne	Gérer déplacement	A Acquérir mesures
Acq_G12	Interne	Gérer déplacement	A Réétalonner le diamètre si Délect trou 2 = TRUE
Fin_init	Interne	Gérer acquisitions	A Gérer déplacement A Délecter trous

## Réponse 8:



## Réponse 9:

- a) Interruptions vectorisées: réception du signal /DTACK. Choix par l'utilisateur du numéro de vecteur (64 à 255) à charger dans le registre vecteur du périphérique, il est ensuite fourni par ce dernier.  
Interruptions auto-vectorisées: réception du signal /VPA. Le numéro de vecteur est imposé par le niveau d'interruption (niveau 1 à 7 soit n° vecteur 25 à 31).
- b) Interruptions locales: Interruptions auto-vectorisées.  
Interruptions VME: Interruptions vectorisées.
- c) Parmi les 7 registres (ICR1 à ICR7 VME Bus Interrupt Control Register), c'est le registre ICR7 qui est concerné par les interruptions sur la ligne /IRQ7 du bus VME.  
L'adresse du registre ICR7 est: \$FEC0101F  
Son contenu: 0xxx x111 Its validées (B7 = 0) et niveau 7 (bits B<sub>2</sub> B<sub>1</sub> B<sub>0</sub> à 1).

## Réponse 10:

- a) Sélection de l'adresse de la rangée par le signal /RAS, sélection de l'adresse de la colonne par le signal /CAS. L'adresse est complète, le transfert de la donnée peut s'effectuer.
- b) Adresse: 10 bits pour la rangée, 10 bits pour la colonne (A<sub>0</sub> à A<sub>9</sub>) soit 2<sup>20</sup> combinaisons.  
Données: 32 bits (I/O 1 à I/O 32). La capacité d'une banque est de 1 Méga mots de 32 bits (ou 4 Moctets).
- c) Ces signaux permettent d'accéder aux octets, mots ou longs mots. Les transferts pouvant être alignés ou non.

/WRBY3	/WRBY2	/WRBY1	/WRBY0	Taille opérande	Offset par rapport à adresse de base de la mémoire
1	1	1	1	Aucun accès	
0	1	1	1	octet	+ 0
1	0	1	1	octet	+ 1
1	1	0	1	octet	+ 2
1	1	1	0	octet	+ 3
0	0	1	1	mot	+ 0
1	0	0	1	mot	+ 1 (non aligné, changement de A <sub>2</sub> )
1	1	0	0	mot	+ 2
0	1	1	0	mot	+ 3 (non aligné)
0	0	0	0	long mot	+ 0 (aligné)
					+ 2 (non aligné, changement de A <sub>2</sub> )

## Réponse 11:

/OEBA	/ABEN	/LADI	/LADO	Etat buffer d'adr.	Sens du transfert	Mode de fonctionnement
1	1	x	x	H.I.		
0	1	1	x	Latched	EXT ⇒ Carte	Esclave
0	1	0	x	Transparent	EXT ⇒ Carte	Esclave
1	0	x	1	Latched	Carte ⇒ EXT	Maître
1	0	x	0	Transparent	Carte ⇒ EXT	Maître

## Réponse 12:

a) \$2820 ⇒ 0010 1000 001x xxxx en binaire soit des cavaliers sur:

	A <sub>15</sub>	A <sub>14</sub>	A <sub>13</sub>	A <sub>12</sub>	A <sub>11</sub>	A <sub>10</sub>	A <sub>9</sub>	A <sub>8</sub>	A <sub>7</sub>	A <sub>6</sub>	A <sub>5</sub>
S1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0

b) Adresse de base de la carte: \$ FFFF 2820

c) Codes modificateurs d'adresse: \$29 (Mode utilisateur) et \$2D (Mode superviseur).  
Cavalier S2 coté A pour accès dans les 2 modes.

## Réponse 13:

Les 4 conversions prennent environ 50 μs. Le déplacement de la tôle est de:

$$(12 \text{ m/s}) \times (50 \text{ } \mu\text{s}) = 0,6 \text{ mm.}$$

Déplacement négligeable devant les 50 cm correspondant au déplacement de la tôle entre 2 acquisitions successives.

## Réponse 14:

a) Registre HLR = 0000 0000

Registre LHR = 0000 0011      It front montant sur E17 et E16.

b) Effacer l'indicateur positionné lors de la prise en compte de la demande d'interruption.

RSTITIN = 1111 1110 (\$FE) pour effacer l'indicateur correspondant à l'entrée E16.

RSTITIN = 1111 1101 (\$FD) pour effacer l'indicateur correspondant à l'entrée E17.

Valeur à écrire à l'adresse \$FFFF 283F avant le retour au programme interrompu (Avant instruction Retour d'interruption RTE).

## Réponse 15:

a) Niveau 7 sur le bus VME soit la valeur 07 dans le registre ITLEV (Adresse \$FFFF 282E).

b) Au moins les registres en caractères gras.

PIC Maître ICW1 = \$11

**ICW2 = \$50**

ICW3 = \$40

ICW4 = \$01

**OCW1 = 1000 1111 (\$8F)**

**IR6, COUT5, COUT4**

OCW2 = \$40

OCW3 = \$48

PIC Esclave ICW1 = \$11

**ICW2 = \$58**

ICW3 = \$06

ICW4 = \$01

**OCW1 = 1111 1100 (\$FC)**

**E17, E16**

OCW2 = \$40

OCW3 = \$48

Pour la commande fin d'interruption:

**\$20 ⇒ OCW2 Maître si It. TIMER      \$20 ⇒ OCW2 Maître et Esclave si Its En.**

c) Numéro de vecteur:      It. Entrée E16: \$58

It. Entrée E17: \$59

It. Timer 5: \$55

It. Timer 4: \$54

## Réponse 16:

a) Nombre de positions du buffer circulaire:

$N = 900 / 0.50 = 1800$   
 On prendra  $N = 2000$ ;  
 Organisation du stockage voir ci-contre.  
 Volume du stockage:  
 $\text{Vol} = N * \text{sizeof}(\text{struct mesures}) + 5 * \text{sizeof}(\text{short})$   
 + en-tête (éventuellement)

$\text{Vol} = 20010 \text{ octets} + \text{en-tête}$

b) Déclarations en C:

```
#define N 2000
```

```
struct Mesures
```

```
{
    short epais_centre;
    short epais_prof_1;
    short epais_prof_2;
    short epais_etain;
    unsigned char presence_trou_1;
    unsigned char presence_trou_2;
}
```

```
struct Stock_mesures
```

```
{
    short index_epais;
    short index_etain;
    short index_trou_1;
    short index_trou_2;
    short index_cisaille;
    struct Mesures tab_mes[N];
}
```

```
struct Stock_mesures *ptab;
```

c) positions initiales des pointeurs:

position épaisseur :	0	position trou2:	1196
position étain:	560	position cisaille:	1236
position trou1:	612		

d) Définition de la fonction écriture:

```
Ecriture_mesures(struct Mesures info , struct Stock_mesure *p_mes ) {
```

```
    short i_epais,
           i_etain,
           i_trou1;
```

```
    Prendre (sema_mode);
```

```
    i_epais = p_mes->index_epais;
```

```
    i_etain = p_mes->index_etain;
```

```
    i_trou1 = p_mes->index_trou_1;
```

```
    Vendre (sema_module);
```

```
    /*Mise à jour des positions des index
```

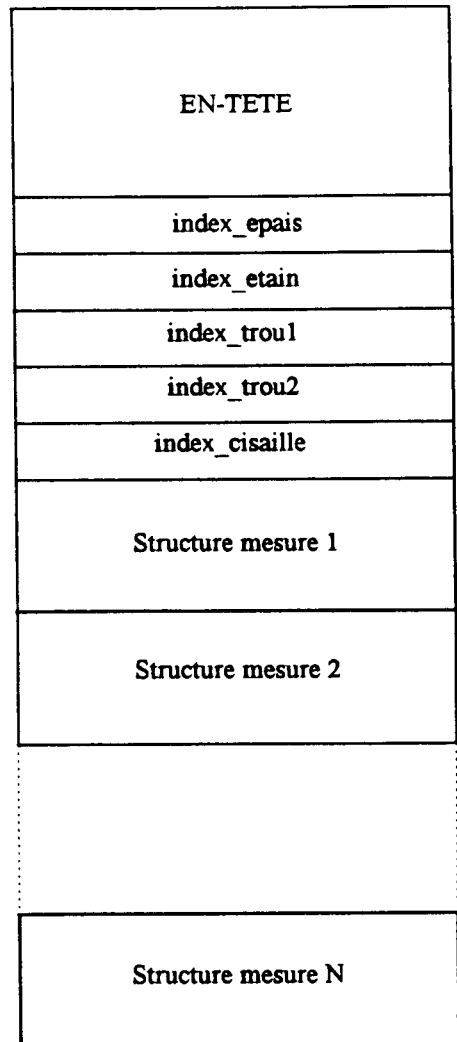
```
    */
```

```
    if(i_epais == 0) i_epais = N;
```

```
    else i_epais--;
```

```
    if(i_etain == 0) i_etain = N;
```

```
    else i_etain--;
```



```

/*la gestion de l'index trou1 peut être considérée comme facultative
*/
if(i_trou1 == 0) i_trou1 = N;
else i_trou1--;

/*Prise de la ressource module*/
Prendre (sema_mode) ;
/* Stockage des mesures: le champ présence_trou_1 est laissé à zéro*/
p_mes->tab_mes[i_epais].epais_centre = info.epais_centre ;
p_mes->tab_mes[i_epais].epais_prof_1 = info.epais_prof_1 ;
p_mes->tab_mes[i_epais].epais_prof_2 = info.epais_prof_2 ;
p_mes->tab_mes[i_etain].epais_etain = info.epais_etain;
/*Stockage des index*/
p_mes->index_epais = i_epais ;
p_mes->index_etain = i_etain ;
p_mes->index_trou_1 = i_trou1;

/*Libération de la ressource module*/
Vendre (sema_module) ;
}

```

**Réponse 17:**

Plusieurs solutions possibles.

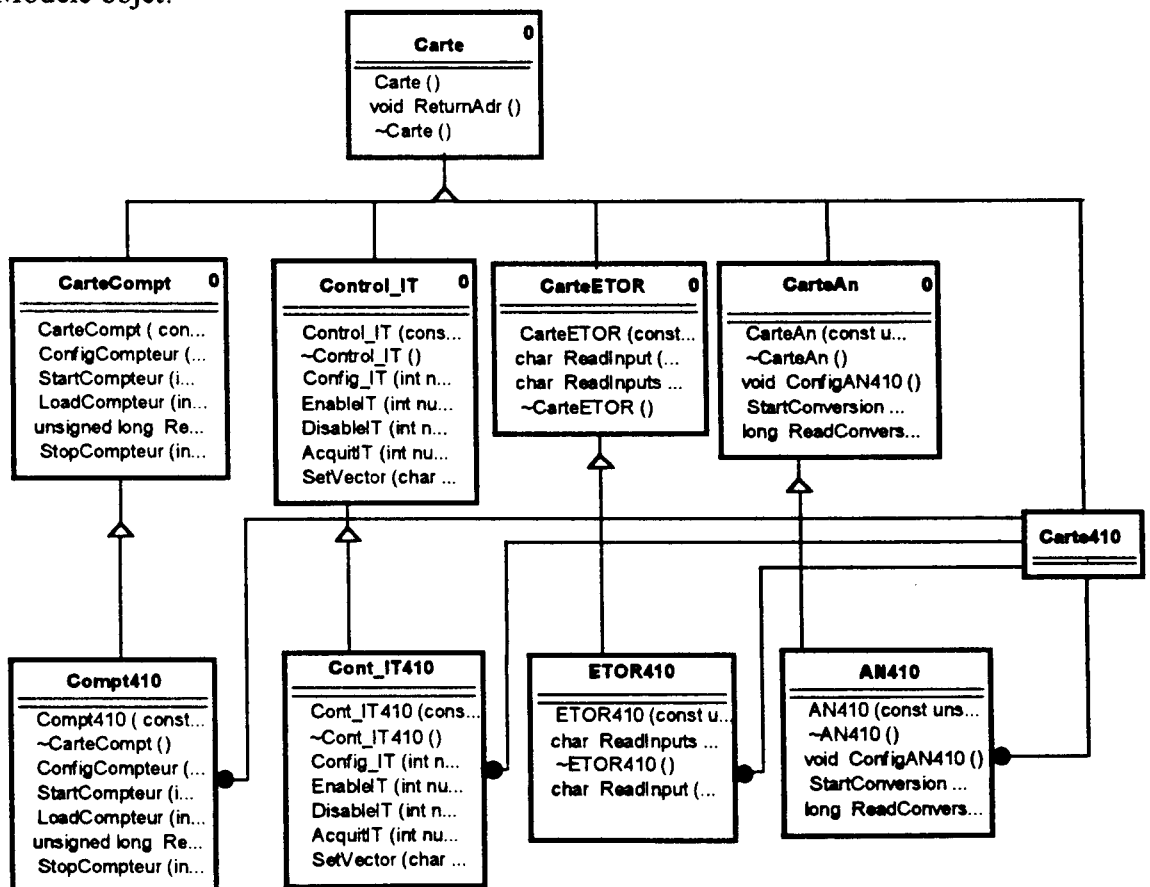
Vérifier l'utilisation de plusieurs files (Epaisseur, étain, trou 1, trou 2).

Utilisation soit d'une structure, soit d'une classe, ...

Vérifier la cohérence de la réponse et l'utilisation correcte des modèles.

**Réponse 18:**

a) Modèle objet:



## b) Constructeur Carte410

```

Carte410::Carte410(const unsigned long adr_base) : Carte(adr_base) {
    ptr_it = new Cont_IT410(adr_base);
    ptr_compt = new Compt410(adr_base);
    ptr_an = new AN410(adr_base, 4);
    ptr_eter = new ETOR410(adr_base, 3, 8);
}

```

## c) Instanciation de l'objet carte:

```

void main() {
    Carte410 objet410(0xFFFF2820);
    .....
    /*lancement de la conversion sur la voie 0*/
    objet410.RetCarteAN()->StartConversion(0);
}

```

**Réponse 19:**

a) Non, elle possède une fonction membre virtuelle pure. Elle est donc abstraite.

```

b) RS232 :: RS232(char *nP = "COM1", long vB = 9600, char p = 'n', int tD = 8, int bS = 1) {
    strcpy(NomPort, nP);
    vitesseBauds = vB;
    parite = p;
    tailleDonnee = tD;
    bitStop = bS;
}

```

**Réponse 20:**

```

#ifndef _WINDOWSPORT_HPP
#define _WINDOWSPORT_HPP

#include "RS232.hpp"
class WindowsPort : public RS232 {
private:
    int identPort;
public:
    WindowsPort(char * nP, long vB, char p, int tD, int bS) : RS232(nP, vB, p, tD, bS);
    char Lire(void);
    void Ecrire(char);
    ~WindowsPort();
};
#endif

```

```
#define TAILLE_BUFFER_RECEPTION 100
#define TAILLE_BUFFER_EMISSION 100

#include "windowsport.hpp"
WindowsPort :: WindowsPort(char * nP, long vB, char p, int tD, int bS) {
    DCB dcb ;
    char buffer[30] ;
    identPort = OpenComm(nP, TAILLE_BUFFER_RECEPTION, TAILLE_BUFFER_EMISSION) ;
    sprintf(buffer, "%s:%ld,%c,%d,%d", nP, vB, p, tD, bS) ;
    BuildCommDCB(buffer, &dcb) ;
    SetCommEventMask(identPort, EV_TXEMPTY) ;
    SetCommState(&dcb) ;
}
char WindowsPort :: Lire(void) {
    char carLu ;
    int result ;
    result = ReadComm(identPort, &carLu, 1) ;
    if(result > 0) return carLu ;
}
void WindowsPort :: Ecrire(char c) {
    WriteComm(identPort, &c, 1) ;
}
WindowsPort :: ~WindowsPort() {
    while(!GetCommEventMask(identPort, EV_TXEMPTY)) { } ;
    CloseComm(identPort) ;
}
```