Carte réseau sur bus PC104 EID003.

Manuel de Travaux Pratique

DIDALAB
5 Rue du Groupe Manoukian
78990 Elancourt
Tel: 01.30.66.08.88 / Fax: 01.30.66.72.20
ge@didalab.fr

EID003_TP 1 GESTION DU TELNET

1.1 Enoncé du sujet

Objectifs:	Etre capable d'utiliser les utilitaires rangés en bibliothèque, permettant la gestion du Telnet en réception.
Cahier des charges :	**Sujet** Ecrire un programme en assembleur et en C qui réalisera l'ouverture port Telnet, et la réception de caractères avec la carte EID003.

Matériel nécessaire :

Micro ordinateur de type PC sous Windows 95 ou ultérieur,
Carte mère 16/32 bits à microcontrôleur 68332, Réf : EID 210 000
Carte Web : EID00300
Câble de liaison Réseau, et câble RS232, Réf : EGD 000 003
Alimentation AC/AC 8V, 1 A Réf : EGD000001,

Documentations nécessaires :

Document : DMS Web: EID00300

Durée : 1 séance de 3 heures

1.2 Eléments de solution

Le numéro du port TCP/IP pour les applications telnet est 23.
Il faut ouvrir le port avec SetPort(23), pour réaliser des applications telnet avec la carte EID003.

La carte EID003 gère :

- L'ouverture et la fermeture de la communication.
- La réception de donnée à partir du buffer FIFO (réception).

L'utilisateur doit gérer les données entrantes.

Chaque caractère reçu est affiché ainsi que le registre status.
La lettre 'q' met fin à l'application telnet.

Le registre `EID003_RX_COUNT` retourne le nombre de caractères présents dans le buffer de réception `EID003_BUFFER`.

Le fichier EID003.h intègres les sous fonctions suivantes :

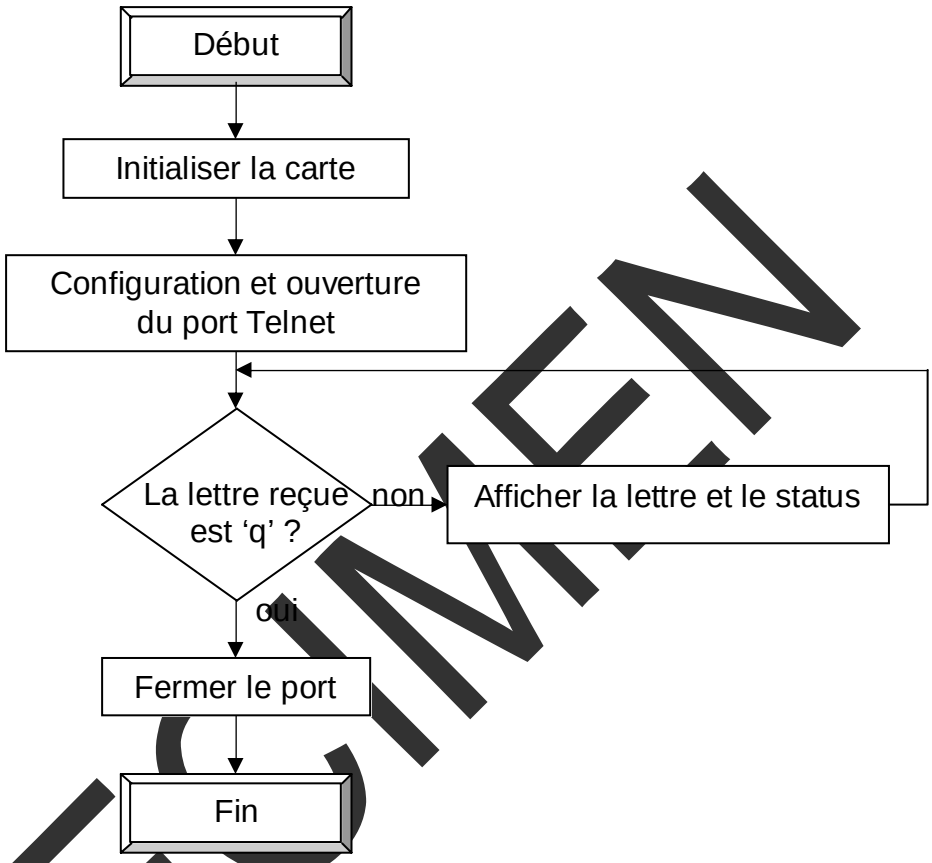
`Init_pc104_eid003();` permet d'initialiser les temps d'accès de la carte web

`LitPort();` permet de connaître le port ouvert.

Une foi le programme lancé, il faudra exécuter l'application telnet de Windows :

L'adresse notée est 10.0.0.24 car il s'agit de l'adresse par défaut de la carte.

1.3 Organigramme principal



1.4 Programme en C

```
/*******************************************************
*     TP SUR LA CARTE EID003                           *
*******************************************************
*     Ecrire un programme en C                         *
*     qui réalise la gestion du Telnet                 *
*******************************************************
*     Nom du FICHIER : EID003_TP1.c                    *
*     **************                                   *
*******************************************************/

#include "eid210.h"
#include "eid003.h"

main()
{
int i;
short port;
char status;
char travail;
clsscr();
Init_pc104_eid003();
SetPort(23);            // PORT TCP/IP telnet
port=LitPort();
printf("Port=%d\n",port);
EID003_CONFIG=0x01;     // ouverture du port
//EID003_CONFIG=0x02;   // vider les buffers
travail=0;
printf("STATUS=%2.2x\n",EID003_STATUS);
while(travail!='q')
      {
            while (EID003_RX_COUNT>0)
                  {
                  travail=EID003_BUFFER;
                  EID003_BUFFER=travail;
                  printf("carac=%c status=%2.2X\n",travail,EID003_STATUS);
                  }
      }
SetPort(0);             // fermeture du port
EID003_CONFIG=0x01;
port=LitPort();
printf("Port=%d\n",port);
}
```

EID003_TP 2 CREATION D'UN CHAT A L'AIDE DU PORT TELNET

2.1 Enoncé du sujet

Objectifs:	Etre capable d'utiliser les utilitaires rangés en bibliothèque, permettant de gérer intégralement le port Telnet.
Cahier des charges :	**Sujet** Ecrire un programme en assembleur et en C qui réalisera l'ouverture port Telnet, l'émission et la réception de caractères avec la carte EID003.

Matériel nécessaire :

Micro ordinateur de type PC sous Windows 95 ou ultérieur,
Carte mère 16/32 bits à microcontrôleur 68332, Réf : EID 210 000
Carte Web : EID00300
Câble de liaison Réseau, et câble RS232, Réf : EGD 000 003
Alimentation AC/AC 8V, 1 A Réf : EGD000001,

Documentations nécessaires :

Document : DMS Web: EID00300
Carte EID210 seule : EID210040

Durée : 1 séance de 3 heures

2.2 Eléments de solution

2.2.1 Mise en œuvre du TP

Le numéro du port TCP/IP pour les application telnet est 23.
Il faut ouvrir le port avec SetPort(23), pour réaliser des applications telnet avec la carte EID003.

La carte EID003 gère :

- L'ouverture et la fermeture de la communication.
- L'émission et la réception de donnée à partir du buffer FIFO (émission et réception).

L'utilisateur doit gérer les données entrantes et sortantes.
Une interface permettra de voir le texte reçu et le texte tapé.
La lettre '$' met fin au dialogue.

2.2.2 Registres utiles :

`EID003_BUFFER` contient le caractère reçu lors d'une lecture, à envoyer lors d'une écriture.

`EID003_RX_COUNT` retourne le nombre de caractères présents dans le buffer de réception `EID003_BUFFER`.

`EID003_TX_COUNT` retourne le nombre de caractères placés dans le buffer d'émission `EID003_BUFFER`.

`EID003_CONFIG` le bit 2 permet d'envoyer le buffer.

`EID003_STATUS` le bit 4 est actif lors de l'envoie.

2.2.3 Les bibliothèques mises à dispositions sont :

Pour gérer la liaison série, il faut la configurer à l'aide des registres `SCCR0`, `SCCR1`, et pour détecter l'appuie sur une touche du clavier, les registre `SCSR`, `SCDR` et le masque `RDRF` sont utiles. Leur utilisation est expliqué dans la documentation EID210 seul (ref : EID210040, TP3), et sont définis dans le fichier qsm.h .

Il est possible de modifier l'adresse TCP, et le masque de sous réseau de la carte : void `config_tcp(int tcp_adr,int tcp_mask)` les paramètres doivent être donnés en hexadécimal. Par exemple, on souhaite mettre l'adresse 10.0.0.25 et le masque 255.255.255.0, il faut donner 0a000019 et FFFFFF00 .

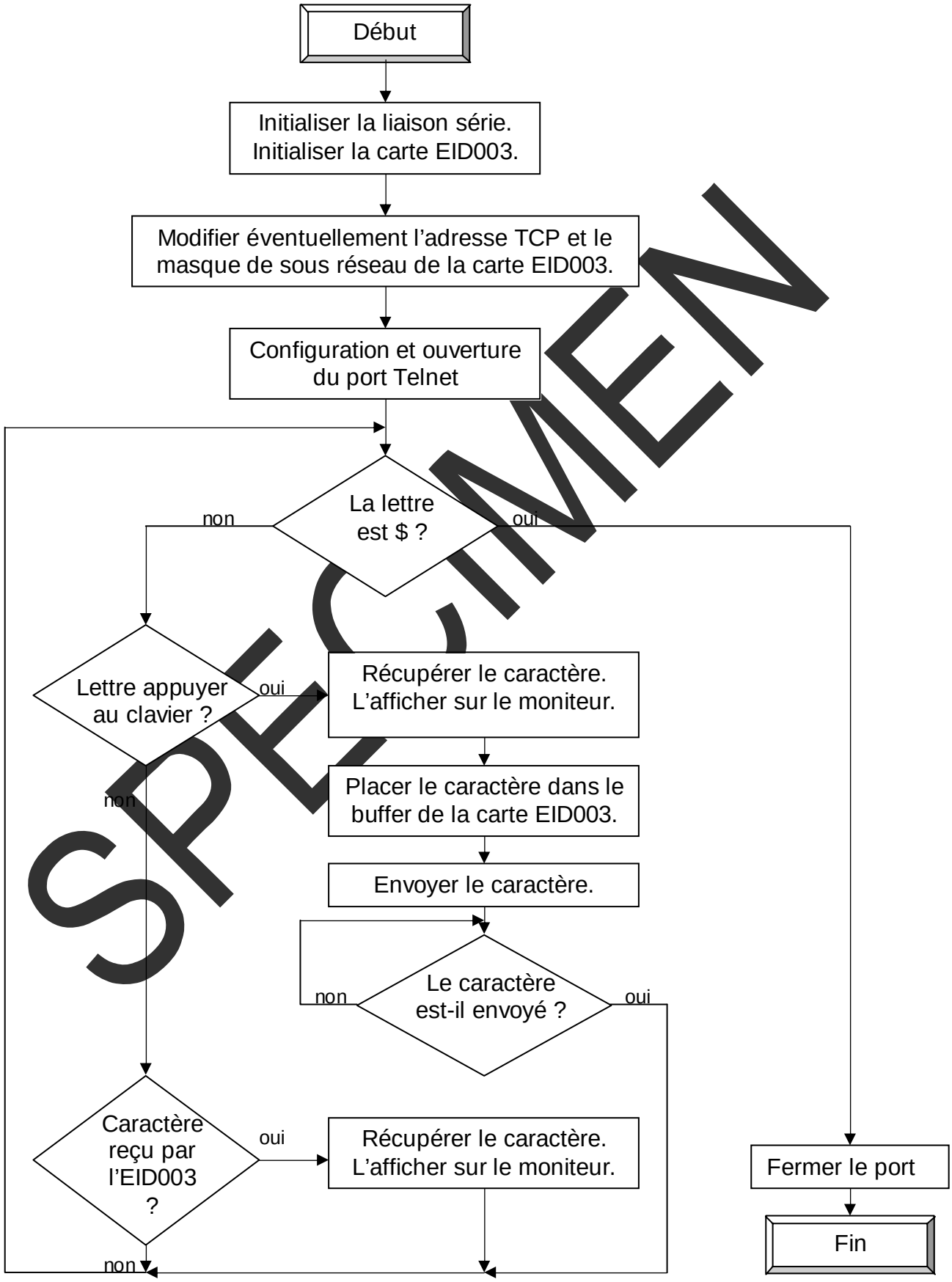
De même il est possible de les lire à l'aide de :

```
int Lit_tcp_adr()
int Lit_tcp_mask()
```

Ces bibliothèques sont rangées dans le fichier EID003_TCPIP.c

Rem : Si l'adresse de la carte est modifiée dans le TP, il faudra alors lancer l'application telnet avec l'adresse correspondante.

2.3 Organigramme



2.4 Programme en C

```
/*************************************************************************/
/*    But du Programme:                                                  */
/* Création d'un chat sur le port telnet                                 */
/*************************************************************************/
#include "eid210.h"
#include "qsm.h"

#include "eid003.h"
#include "eid003_TCPIP.c"

//------------------------------------ Fonction principale
main()
{
      int carac,i;
      char ligrec = 18,colrec = 1,ligemi = 3,colemi = 1;
      clsscr();

     /*Configuration de la liaison série 57600 Bauds en émission+réception*/
      SCCR0 = 9;        // Pour la vitesse de transmission de 57600 bauds
      SCCR1 = 0x000C;   // Pour valider l'émission et la réception

      //Initialisation des temps d'accès pour la carte EID210
      Init_pc104_eid003();

      //Modification éventuelle de l'adresse de la carte
      printf("L'adresse de base de la carte est, par defaut, 10.0.0.24\n");
      printf("pour la modifier en 10.0.0.25, appuyer sur Y\n");
      carac=InRs232();
      if(carac=='Y' | carac=='y') config_tcp(0x0a000019,0xFFFFFF00);
            else config_tcp(0x0A000018,0xFFFFFF00);
      for(i=0;i<100000;i++);
      clsscr();
      //Affichage de l'adresse et du masque de la carte
      carac=Lit_tcp_adr();
      printf("Adresse de la carte : %x \n",carac);
      carac=Lit_tcp_mask();
      printf("Masque de sous réseau de la carte : %x \n",carac);
      for(i=0;i<100000;i++);

      //Configuration + ouverture du port TPC/IP TELNET
      SetPort(23);
      EID003_CONFIG=0x01;

      //Information de fonctionnement
      puts("Appuyer sur une touche pour continuer ...");
      puts("Appuyer $ pour quitter le programme   ...");
      InRs232();

      //Affichage graphique personnalise
      clsscr();
      for(i=0;i<50;i++)
      {
            gotoxy(i,15);
            puts("-");
      }
      gotoxy(0,0);
      puts("Fenetre d'emission :");
      puts("--------------------");
```

```
      gotoxy(0,16);
      puts("Fenetre de reception :");
      puts("----------------------");

//Boucle principale
while(carac!='$')
      {
      if(SCSR&RDRF)     //Si appuie touche
            {
            carac=SCDR; //On récupère le caractère
            EID003_BUFFER=carac;    //On le place dans le buffer d'émission
            EID003_CONFIG = 0x03;   //On l'envoi
            while((EID003_STATUS&0x04)==0x04); //On attend la fin de
l'émission

            //Mise en page sur le moniteur
            gotoxy(colemi,ligemi);
            printf("%c\n",carac);
            puts("                                        ");
            colemi++;
            if(colemi==35)    //Nbs de caractères par ligne
                  {
                  ligemi++;
                  colemi=1;
                  }
            if(ligemi==14)    //Nbs de lignes
                  {
                  colemi=1;
                  ligemi=3;
                  }
            }

      if(EID003_RX_COUNT!=0)  //Si on reçoit un caractère
            {
            carac=EID003_BUFFER; //On le recupére

            //Mise en page sur le moniteur
            gotoxy(colrec,ligrec);
            printf("%c\n",carac);
            puts("                                        ");
            colrec++;
            if(colrec==35)    //Nbs de caractères par ligne
                  {
                  ligrec++;
                  colrec=1;
                  }
            if(ligrec==32)    //Nbs de lignes
                  {
                  colrec=1;
                  ligrec=18;
                  }
            }
      }
SetPort(0);       // fermeture du port
EID003_CONFIG=0x01;
}
```

EID003_TP 3 RECEPTION D'UNE TRAME INTERNET EXPLORER

3.1 Enoncé du sujet

Objectifs:	Etre capable d'utiliser les utilitaires rangés en bibliothèque, permettant de recevoir une trame provenant d'Internet Explorer.
Cahier des charges :	**Sujet** Ecrire un programme en C qui réalisera la réception d'une trame Internet Explorer, l'afficher en identifiant les différentes parties.

Matériel nécessaire :

Micro ordinateur de type PC sous Windows 95 ou ultérieur,
Carte mère 16/32 bits à microcontrôleur 68332, Réf : EID 210 000
Carte Web : EID00300
Câble de liaison Réseau, et câble RS232, Réf : EGD 000 003
Alimentation AC/AC 8V, 1 A Réf : EGD000001,

Documentations nécessaires :

Document : DMS Web: EID00300

Durée : 1 séance de 3 heures

3.2 Eléments de solution

3.2.1 Composition d'une trame Internet Explorer.

Une requête se compose de la façon suivante :
méthode uri versionHTTP ↵
informations↵↵
body(facultatif voir 3.2.1.1)
(le symbole ↵ représente un retour a la ligne et se traduit par le code ascii /r /n)
Exemple :

```
GET /actes.html HTTP/1.1
Accept: application/vnd.ms-powerpoint, image/gif, image/x-xbitmap, image/jpeg,
image/pjpeg, application/msword, application/vnd.ms-excel, application/x-shockw
ave-flash, */*
Referer: http://10.0.0.25/simes.html
Accept-Language: fr
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)
Host: 10.0.0.25
Connection: Keep-Alive

-
```

3.2.1.1 La méthode

Nous allons nous intéresser à 2 type de méthode : le GET et le POST

La méthode GET est la plus courante ; elle permet de demander une page (avec éventuellement des modification). Le body n'est pas présent dans ce cas.

La méthode POST est spécifiquement dédié aux modification au sein d'une page (ex : connaître la position d'un bouton, la validation d'une chekbox,…).
Dans ce cas au sein des informations nous trouverons Content-Lentgh : XX↵
Le XX représente le nombre de caractères présents dans le body

Dans l'exemple ci-dessus la méthode est GET, le Content-Length n'apparaît pas donc le body n'existe pas

3.2.1.2 L'uri

Il s'agit simplement de la page demandées.
Elle apparaît sous la forme /nomdelapage.html
Nous verrons par la suite qu'il peut également s'agir d'une image (bmp, jpg, …) ou d'un son (mp3, wav,…).

3.2.1.3 La version d'HTTP

La carte Web est prévue pour la version HTTP 1.1.
Elle apparaît sous la forme HTTPX.X
Il est également possible de traiter les version 1.0 et 0.9. Attention dans ces cas le protocole de réponse n'est pas le même.

Remarque : chacun de ces 3 éléments est séparé par un espace et la ligne se termine par 1 retour chariot

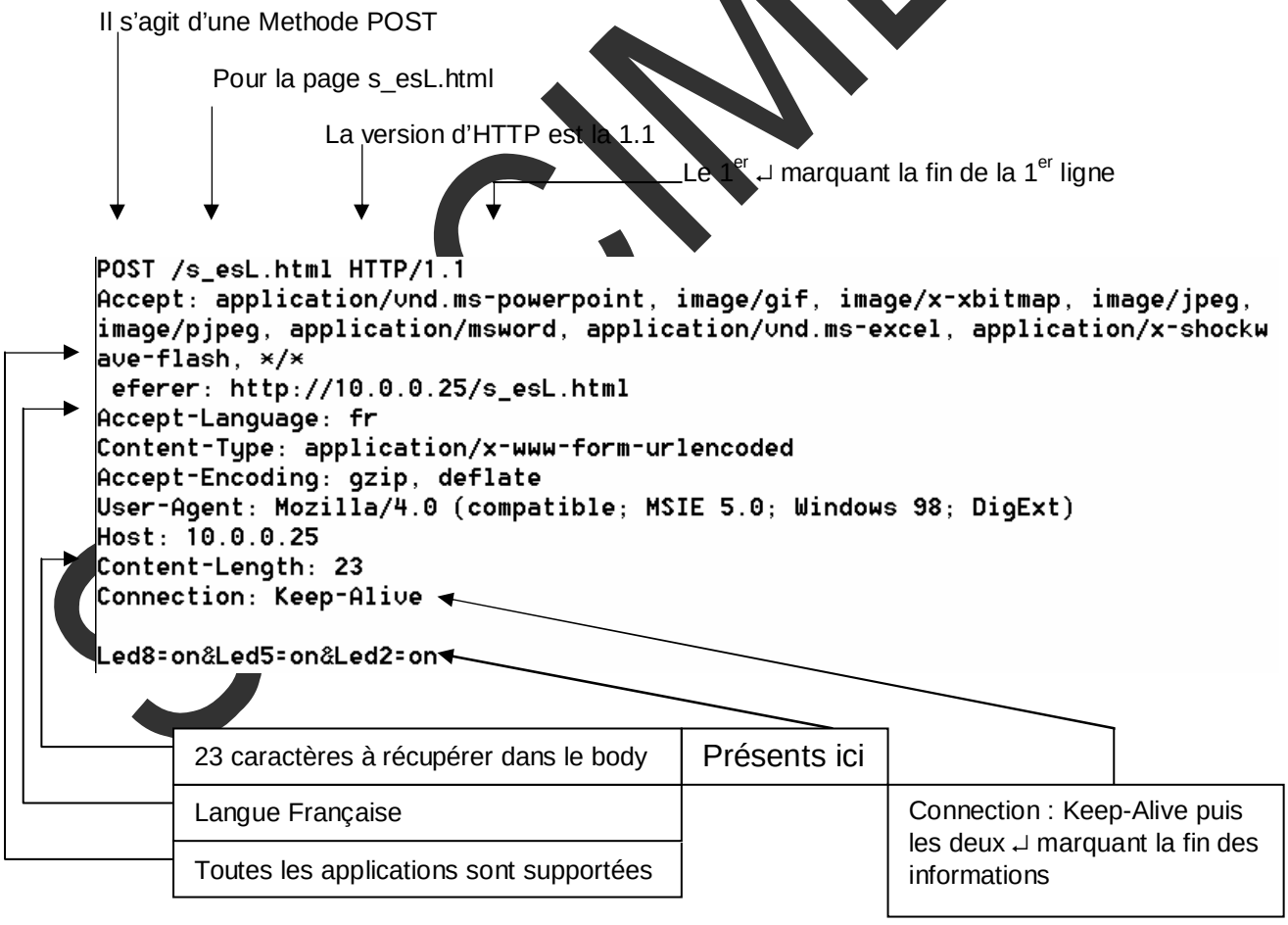
3.2.1.4 Les informations

Dans cette partie, nous pouvons déterminer quelles sont les applications que le client peut gérer.
Dans notre exemple, la ligne Accept : application, nous permet de dire que le client peut traiter les images gif, bitmap, jpeg et les traitement du type word, excel, flash. Le symbole final */* signifie que le client peut traiter toutes les applications.
La ligne Referer nous donne l'adresse précédente tapée par le client.
Accept-Language : il s'agit de la langue par défaut.
Connection : Keep-Alive puis les deux ↵ signale la fin des informations.

3.2.1.5 Le Body

Il apparaît si la valeur de Content-Length : ,dans les informations, est différente de zéro.
En général il est présent lors d'une méthode POST.

3.2.1.6 Exemple commenté



3.2.2 Mise en œuvre du TP

Le numéro du port TCP/IP pour les application Internet Explorer est 80.
Il faut ouvrir le port avec SetPort(80), pour réaliser des applications Internet Explorer avec la carte EID003.

Le programme doit être capable de traiter les méthodes GET et POST, ainsi que la version HTTP1.1.

La langue et les application supportées devront être affichées.
La première ligne devra être identifiée argument par argument et affiché.
La requête devra être affichée entièrement.
Pour cela une structure est définie dans le fichier eid003_HTML.c (il doit être inclus au début du programme).

```
struct
{
char complet[1000];
char methode[10];
char uri[150];
char version[10];
     struct
     {
     char ToutType;
     char gif;
     char bitmap;
     char jpeg;
     char pjpeg;
     char word;
     char excel;
     }accept;
char body[150];
char tmp[100];
int cpt;
}Buffer_reception;
```


Buffer_reception est composé de tous les membre ci dessus.

Par exemple Buffer_reception.complet est un buffer qui occupe les 1000 1er caractère de Buffer_reception, Buffer_reception.methode occupe les 10 caractères suivants, …

Ainsi, lorsque l'on reçoit une requête, chaque caractère reçu sera placé dans Buffer_reception.complet[X]. Ensuite il faudra extraire de Buffer_reception.complet chaque élément et les placer dans Buffer_reception.methode, Buffer_reception.uri, Buffer_reception.version, Buffer_reception.body (dans le cas d'un POST).

Buffer_reception.tmp est un buffer de 100 caractères laissé libre à l'utilisateur.

Buffer_reception.cpt est une variable qui permettra à l'utilisateur de placer chaque caractère au bon endroit dans les différents buffers (cf le X au-dessus).

Buffer_reception.accept est constitué par une structure. Il permettra de connaître les applications acceptées par les client. Pour avoir accès à l'un d'entre eux, il faut faire Buffer_reception.accept.gif .

Les bibliothèques mises à dispositions sont :

`Init();` Efface les variables Buffer_reception et Buffer_emission.

`VideBufferRX();` Vide le buffer de réception tant que `EID003_RX_COUNT` >0 et le place dans Buffer_reception.complet[Buffer_reception.cpt]. Attention il faut impérativement que Buffer_reception.cpt soit correctement initialisé ! ! !

`IndentifieLigneRequete();` Organise Buffer_reception.methode, Buffer_reception.uri, Buffer_reception.version en fonction de Buffer_reception.complet

`IndentifieLigneAccept();` identifie les applications supportées par le client, et l'affiche au moniteur.

`IndentifieLigneLanguage();` identifie la langue supportée par le client, et l'affiche au moniteur.

`IndentifieLigneBody();` identifie le Content-Length : xx et place le contenu du body dans Buffer_reception.body .

`AfficheBuffer(char *buff,int y);` affiche le buffer(1er paramètre) sur la ligne (2eme paramètre).

`int Search_String(char *ptr1,char *ptr2);` Cherche dans le 1er buffer le 2eme buffer. La fonction renvoie 0 si le 2eme buffer n'est pas dans le 1er buffer. Si le 2eme buffer est dans le 1er buffer, la fonction retourne la position de fin ; ex :

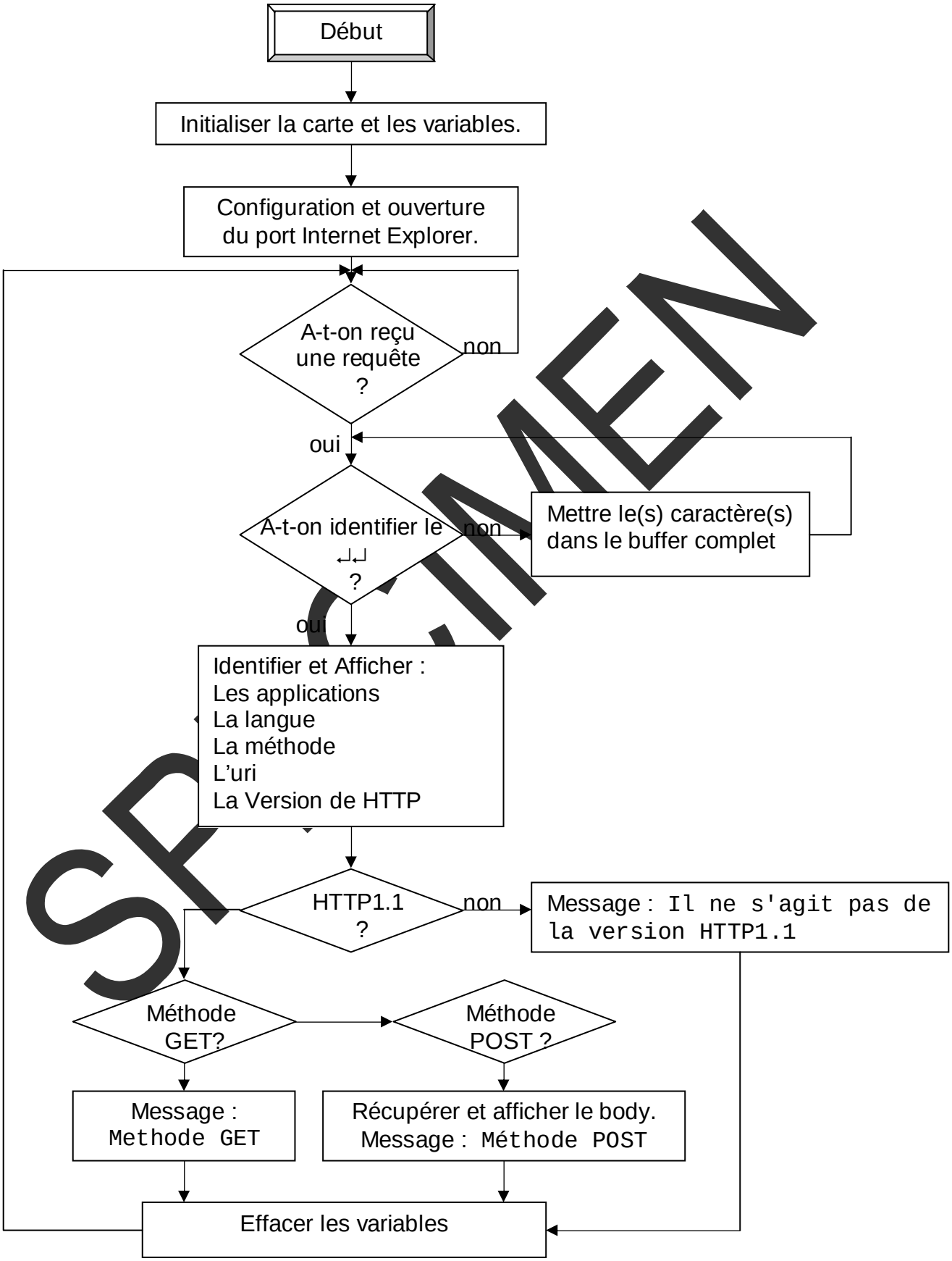
```
result=Search_String("azerty","er");
```

Dans ce cas result=5.
Cette fonction nous permettra de récupérer la valeur du Content-Length modifier un buffer.

Il faut intégrer les bibliothèques stdio.h et string.h pour se servir des fonction de traitement de chaînes de caractères (ex :strstr, strcmp,…).

Une foi le programme exécuté, il faut lancer l'application Internet Explorer de Windows, avec l'adresse http://10.0.0.24/index.html . (10.0.0.24 étant l'adresse par défaut de la carte)

3.3 Organigramme :



3.4 Programme en C

```
/*********************************************************/
/*    Création d'un serveur HTLM                          */
/*    But du Programme:                                  */
/*          Réalisation d'un serveur HTTP1.1              */
/*          Analyse des trames provenant d'Internet Explorer     */
/*                                                                */
/*********************************************************/

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

#include "eid210.h"
#include "eid003.h"

#include "eid003_HTML.c"
/*********************************/
/*    Programme principal          */
/*********************************/
main()
{
clsscr();
//Initialisation des temps d'accès pour la carte EID210
Init_pc104_eid003();
//Configuration + ouverture du port TPC/IP Internet Explorer
SetPort(80);
EID003_CONFIG=0x01;
//Initialisation des variables
Init();

//Boucle principale
clsscr();
while(CRTL==1)
      {
      while(EID003_RX_COUNT==0);//Si le buffer est vide, on attend
      clsscr();

      while(0==strstr(Buffer_reception.complet,standar[1])) //On receptionne
tout le buffer
      VideBufferRX();                    //jusqu'à \r\n\r\n

      code_erreur=0;

//Vérification des applications et de la langue supportées par le client
      IndentifieLigneAccept();           //Formats supportés
      IndentifieLigneLanguage();    //Quelle est la langue
      InRs232();                    //Attente un appuie touche
      clsscr();
      //On identifie tout les éléments provenant de IE
      IndentifieLigneRequete();          //On identifie la ligne de requête
Méthode, URI_PAGE, VERSION_HTTP
      //On les affiche
      AfficheBuffer(Buffer_reception.methode,2);
      AfficheBuffer(Buffer_reception.uri,4);
      AfficheBuffer(Buffer_reception.version,6);
      AfficheBuffer(Buffer_reception.complet,10);
```

```
      //Identifie la version HTTP1.1
      if(0!=strstr(Buffer_reception.version,version[0]))
           {
            code_erreur=code_erreur|0x0002; //code HTTP1.1
            printf("Version HTTP 1.1\n"); //Message HTTP1.1
          //Methode GET
            if(0!=strstr(Buffer_reception.methode,methode[0]))
                 {
                  code_erreur=code_erreur|0x0020; //code GET
                  printf("Methode GET\n"); //code GET
                 }
          //Methode POST
            if(0!=strstr(Buffer_reception.methode,methode[1]))
                 {
                  code_erreur=code_erreur|0x0040; //code POST
                  printf("Methode POST\n");  //Message POST

               //Si le Content-Length est présent
                  if(0!=strstr(Buffer_reception.complet,"Content-Length: "))
                        {
                   //=> il faut récupérer le body venant de IE
                         while(EID003_RX_COUNT!=0) VideBufferRX();
                  //Identifier le body et le mettre dans son buffer
                         IndentifieLigneBody(Buffer_reception.body);
                  //puis l'afficher
                        AfficheBuffer(Buffer_reception.body,8);
                         }
                   }
            }
            else  //Ce n'est pas la version HTTP1.1 on en informe l'utilisateur
            {//et faire code_erreur=code_erreur|0x0004ou8.
             //Sinon on veut on peut traiter d'autres version ici
            printf("Il ne s'agit pas de la version HTTP1.1\n");
            }
      Init(); //Efface les variables
      }
}
```

EID003_TP 4 RECEPTION ET REPONSE SUR INTERNET EXPLORER

4.1 Enoncé du sujet

Objectifs:	Etre capable d'utiliser les utilitaires rangés en bibliothèque, permettant d'interpréter et de répondre à une requête Internet.
Cahier des charges :	**Sujet** Ecrire un programme en assembleur et en C qui analysera la requête, et répondre en conséquence. Il faudra également gérer les erreurs.

Matériel nécessaire :

Micro ordinateur de type PC sous Windows 95 ou ultérieur,
Carte mère 16/32 bits à microcontrôleur 68332, Réf : EID 210 000
Carte Web : EID00300
Câble de liaison Réseau, et câble RS232, Réf : EGD 000 003
Alimentation AC/AC 8V, 1 A Réf : EGD000001,

Documentations nécessaires :

Document : DMS Web: EID00300

Durée : 1 séance de 3 heures

4.2 Eléments de solution

4.2.1 Réponse standard du protocole HTTP1.1

La trame se présente de la manière suivante :
HTTP/1.1
Espace
Status-Code espace Reason-Phrase
↵↵
Body
↵↵

Le but va etre de créer un buffer contenant tous ces éléments.
Rem : pour le HTTP0.9, on peut se permettre d'envoyer que : Body↵↵

4.2.1.1 Constitution du buffer

Pour ce faire il existe une structure, crée dans le fichier eid003_HTML.c :

```
Struct
{
char version[8];
char space;
char reason [50];
char crlf[2];
char doublecrlf[4];
char body[5000];
}Buffer_emission;
```

Sont fonctionnement est similaire à Buffer_reception.
Lors de l'initialisation (sous programme Init();), Buffer_emission.version, Buffer_emission.reason, Buffer_emission.body sont remplis par le caractère NULL.
Buffer_emission.space est un espace, Buffer_emission.crlf est un retour chariot, Buffer_emission.doublecrlf est deux retours chariots.
Avant d'envoyer ce buffer, il faudra remplir la version, la reason et le body.

Un autre buffer `char Buff_tmp[15000];` est mis a disposition de l'utilisateur.

4.2.1.2 Buffer_emission.version

Il faut y placer la version.
On pourra se servir de Buffer_reception.version, et en faire un copier coller dans Buffer_emission.version
Rem : le protocole HTTP1.1 et HTTP1.0 sont identiques.

4.2.1.3 Buffer_emission.reason

La reason est constitué par le Status-Code, un espace, et la Reason-Phrase.
Cet deux élément sont définies par le protocole.
Ce que l'on retrouve le plus fréquemment sont définis dans le fichier eid003_pageHTML.c.
Cette fois encore un copier coller permettra de le placer dans le Buffer.

`//raison évoqué lors d'un problème`
`char Reason_200[]="200 OK";`
`char Reason_400[]="400 Bad Request";`
`char Reason_404[]="404 Not Found";`
`char Reason_414[]="414 Request-URI Too Long";`
`char Reason_501[]="501 Not Implemented";`
`char Reason_505[]="505 HTTP Version Not Supported";`
	Requête Comprise.
	Erreur lors de la requête.
	URI inexistante.
	URI trop longue.
	Fonction non traité par le serveur.
	Version HTTP différente.

4.2.1.4 Buffer_emission.body

Il comprend la page HTML.
Attention cette foi il est obligatoire.
Dans le cas ou la reason est `"200 OK"` ,on y place la page HTML.
Dans le cas d'une erreur, on y placera le buffer `Erreur_HTML[], qui est constitué de "<html></html>"` .

4.2.1.5 Tableau de buffer comparatif

Pour répertorier les différentes méthodes, uri, versions d'HTTP, des tableau de buffer sont crées sous cette forme :

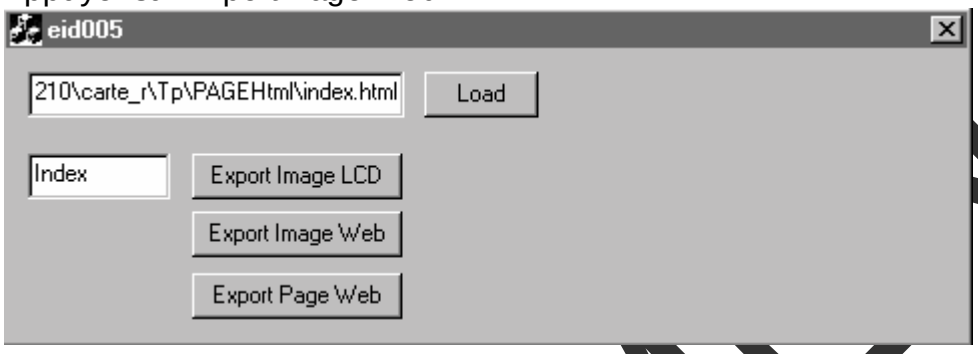
`char *methode[]={`
`"GET",`
`"POST",`
`"HEAD",`
`"DELAY"`
`};`
	`char *pages[]={`
	`"/index.html",`
	`"/simes.html",`
	`"/favicon.ico"`
	`};`
		`char *version[]={`
		`"HTTP/1.1",`
		`"HTTP/1.0",`
		`"HTTP/0.9"`
		`};`

Rem : défini de cette manière, la fonction strstr peut-être utilisée directement.

4.2.2 Constitution d'une page HTML

Il faut créer une page HTML avec l'éditeur de son choix.
Après il faut la transformer en un buffer avec le logiciel fourni.

Appuyer sur load pour sélectionner la page HTML à convertir.
Dans la fenêtre en dessous donner le nom du buffer.
Appuyer sur Export Page Web



La page index.html sera retranscrite en index.c et le buffer sera nommé en Index_html.Le fichier .c sera crée dans le répertoire du fichier à retranscrire.
Dans le TP toutes les page HTML seront regroupées dans un seul fichier : eid003_pageHTML.c .

4.2.3 Gestion du buffer circulaire de l'EID003

Fonctionnement:
EID003_BUFFER est un buffer circulaire:
Chaque caractère écrit dans ce buffer y est stocké.
Lorsque le buffer de l'eid003 est plein(soit 32 caractères), il sera envoyé dans le buffer de l'IP2222.
Lorsque le buffer de l'IP2222 est plein (soit tout les 128 caractères), il est envoyé automatiquement sur le réseau.
Si le buffer de l'IP2222 n'est pas plein, et si il reste des caractères a envoyer, il faudra forcer un envoie.

4.2.4 Gestion des erreurs

Une variable code erreur permet de vérifier le bon fonctionnement.
Elle est définie dans le fichier eid003_HTML, et se compose de la manière suivante :

```
int code_erreur;  //permet de connaître d'ou vient une potentielle erreur
                  //le  1er  octet :version HTTP:     0=> non identifier
                  //                                  2=> version 1.1
                  //                                  4=> version 1.0
                  //                                  8=> version 0.9
                  //le 2eme octet : méthode     :     0=> non traité
                  //                                  2=> GET ou HEAD
                  //                                  4=> POST
                  //                                  8=> libre
                  //le 3eme octet : page        :     0=> page non répertorié
                  //                                  2=> page répertorié & identifiée
                  //                                  4=> image répertorié & identifiée
```

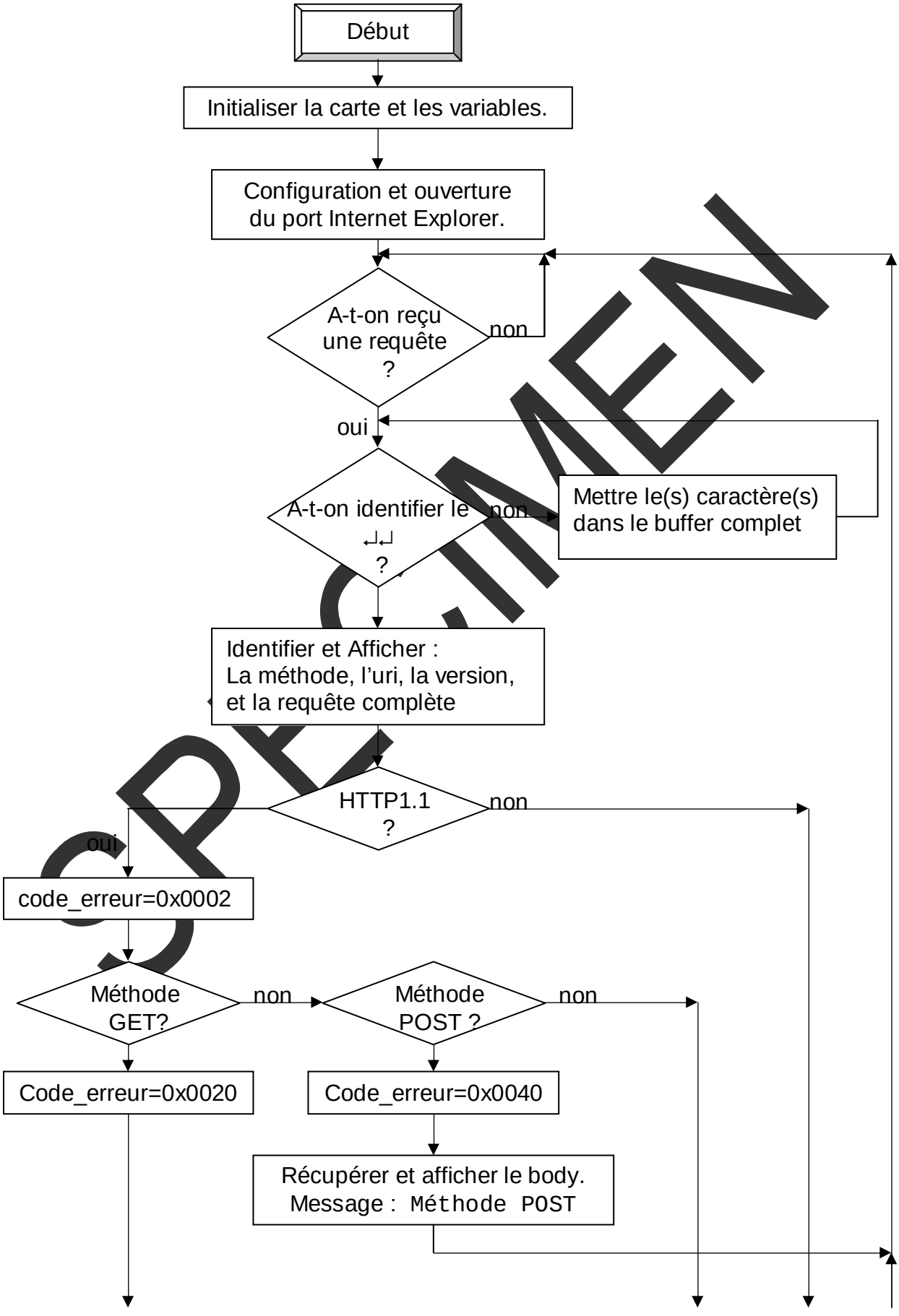
4.2.5 Bibliothèques utiles

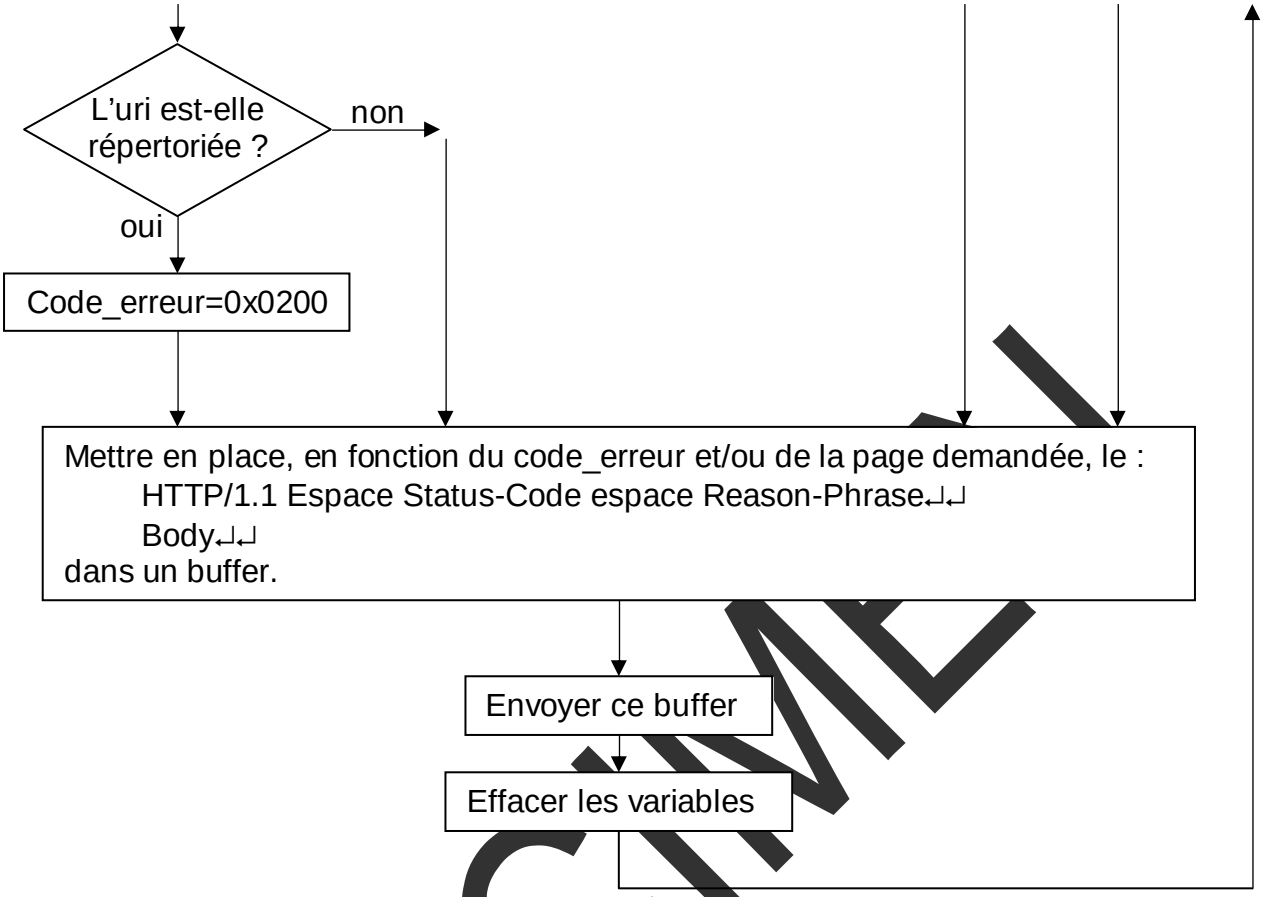
`Reponse_Requete(char Page_HTML[],char Reason[]);` Permet de créer le buffer complet avec la reason et la page adéquate.

`EnvoiePageHTML(char HTML[]);` Permet d'envoyer un buffer.

`Analyse_code_erreur();` Permet de gérer les erreurs : méthode non reconnu, version HTTP différente, page inconnu,….

4.3 Organigramme




L'uri est-elle répertoriée ?
non
oui
Code_erreur=0x0200
Mettre en place, en fonction du code_erreur et/ou de la page demandée, le :
HTTP/1.1 Espace Status-Code espace Reason-Phrase↵↵
Body↵↵
dans un buffer.
Envoyer ce buffer
Effacer les variables

4.4 Programme en C

```
/*********************************************************/
/*    Création d'un serveur HTLM                         */
/*    But du Programme:                                  */
/*          Réalisation d'un serveur HTTP1.1             */
/*          Analyse des trames provenant d'Internet Explorer */
/*          Réponse en respectant la norme HTTP1.1       */
/*********************************************************/

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

#include "eid210.h"
#include "eid003.h"

#include "eid003_pageHTML.c"
#include "eid003_HTML.c"


/***********************************/
/*    Programme principal          */
/***********************************/
main()
{
clsscr();
//Initialisation des temps d'accès pour la carte EID210
Init_pc104_eid003();

//Configuration + ouverture du port TPC/IP Internet Explorer
SetPort(80);
EID003_CONFIG=0x01;

//Initialisation des variables
Init();

//principale
clsscr();
while(CRTL==1)
      {
      while(EID003_RX_COUNT==0);    //Si le buffer est vide, on attend
      clsscr();

      while(0==strstr(Buffer_reception.complet,standar[1])) //On receptionne
tout le buffer
      VideBufferRX();                                  //jusqu'à \r\n\r\n

      code_erreur=0;

      //On identifie la ligne de requete Methode, URI_PAGE, VERSION_HTTP
      IndentifieLigneRequete();
      //On les affiche
      AfficheBuffer(Buffer_reception.methode,2);
      AfficheBuffer(Buffer_reception.uri,4);
      AfficheBuffer(Buffer_reception.version,6);
      AfficheBuffer(Buffer_reception.complet,10);
```

```
      //Identifie la version HTTP1.1
      if(0!=strstr(Buffer_reception.version,version[0]))
            {
            code_erreur=code_erreur|0x0002; //code HTTP1.1
            if(0!=strstr(Buffer_reception.methode,methode[0]))//Methode GET
                  {
                  code_erreur=code_erreur|0x0020; //code GET
                  //Recherche la page qui a été demané
                  if(0!=strstr(Buffer_reception.uri,pages[0]))//Page index
                        Reponse_Requete(Index_HTML,Reason_200);
                  if(0!=strstr(Buffer_reception.uri,pages[1]))//Page simes
                        Reponse_Requete(Simes_HTML,Reason_200);
                  //demande d'ajout dans les favoris
                  if(0!=strstr(Buffer_reception.uri,pages[2]))
                        Reponse_Requete(Erreur_HTML,Reason_200);
                  }
            if(0!=strstr(Buffer_reception.methode,methode[1]))//Method POST
                  {
                  code_erreur=code_erreur|0x0040; //code POST
                  printf("Methode POST\n");
                  //Si le Content-Length est present
                  if(0!=strstr(Buffer_reception.complet,"Content-Length: "))
                        {
                        //=> il faut recuperer le body venant de IE
                        while(EID003_RX_COUNT!=0) VideBufferRX();
                        //Identifier le body et le mettre dans son buffer
                        IndentifieLigneBody(Buffer_reception.body);
                        AfficheBuffer(Buffer_reception.body,8);
                        }
                  }
            }
            else  //Ce n'est pas la version HTTP1.1 on en informe
l'utilisateur
            {     //et faire code_erreur=code_erreur|0x0004ou8.
                  //Sinon on veut on peut traiter d'autres version ici
            printf("Il ne s'agit pas de la version HTTP1.1\n");
            }

      Analyse_code_erreur(code_erreur); //Analyse les erreurs
      //Efface les variables de réception, d'identification et de réponses
      Init();
      }
}
```

EID003_TP 5 INTEGRATION DU CGI.

5.1 Enoncé du sujet

Objectifs:	Etre capable d'utiliser les utilitaires rangés en bibliothèque, permettant d'intégrer des images et des sons. Rendre une page interactive.
Cahier des charges :	**Sujet** Etre capable d' intégrer des son et des images au sein d'une page HTML. Piloter et visualiser le simulateur d'entrées / sorties via une page HTML.

Matériel nécessaire :

Micro ordinateur de type PC sous Windows 95 ou ultérieur,
Carte mère 16/32 bits à microcontrôleur 68332, Réf : EID 210 000
Carte Web : EID003000
Câble de liaison Réseau, et câble RS232, Réf : EGD 000 003
Alimentation AC/AC 8V, 1 A Réf : EGD000001,

Documentations nécessaires :

Document : DMS Web: EID00300
Document carte d'entrées / sorties : EID001000

Durée : 1 séance de 4 heures

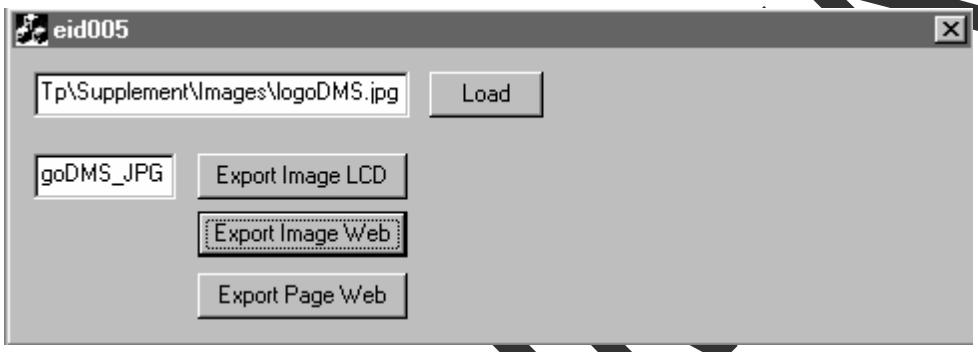
5.2 Eléments de solution

5.2.1 Intégration d'une image et d'un son :

5.2.1.1 Conversion des images et des sons

Le but est de retranscrire le fichier .jpg ou .mp3 en un tableau de valeurs.
Le logiciel fourni permet de le faire.

Appuyer sur Load pour sélectionner le fichier à convertir.
Dans la fenêtre en dessous donner le nom du buffer.
Appuyer sur Export Image Web.



Le fichier logosDMS.jpg sera retranscrite en logoDMS.c et le tableau sera nommé en logoDMS_JPG.
De plus, dans le fichier logoDMS.c, une variable nommé logoDMS_JPG_taille donnera le nombre d'elements au sein du tableau
Le fichier .c sera crée dans le répertoire du fichier à retranscrire.
Dans le TP toutes les page HTML seront regroupées dans un seul fichier : eid003_pageHTML.c . Il inclura les fichiers image.c et son son.c dans le quels se trouveront respectivement les images et les sons.

Ce ci est valable pour tout type d'images et de sons.
! ! ! Attention, la mémoire de l'eid210 n'est pas infinie, il faut donc compresser au maximum les fichiers que l'on veut inclure.
Certaine violation de privilège pourrait apparaître ! ! !

5.2.1.2 Envoyer les images ou les sons

Le principe est le même que pour une page HTML.
Cependant une autre fonction à été crée afin de pourvoir passer en paramètre la taille du tableau de valeurs :

```
Reponse_Image(char Image[],char Reason[],int Taille)
EnvoieImage(char Img[],int Taille)
```

5.2.2 Pilotage du simulateur d'entrées / sorties

Pilotage du simulateur d'entrées / sorties
Pour le fonctionnement du simulateur d'entrées / sorties , il faut ce référer au document carte d'entrées / sorties : EID001000.
Pour piloter le simulateur d'e/s, une bibliothèque et des fonction ont été écrite dans les fichiers eid001.h et eid003.c .
Il faudra inclure le fichier objet eid001.o dans le linkeur, et initialiser les ports A, B, et C.

5.2.3 Rendre une page interactive

Lors du la construction d'une page HTML, il est possible d'inclure des bouton, des Checkbox, des Select,… et de demander une actualisation de cette page. Dans ce cas, lorsqu'une Checkbox est activé, on retrouve dans la requête nomdelacheckbox=on. Si elle ne l'est pas, rien n'apparaît dans la requête.

Le but va donc être de retrouver dans la requête les leds a allumer, et la valeur que le client veut afficher sur le 7 segments.
Il faudra également ; lire la valeur des switchs et du potentiomètre sur la carte eid001, actualiser le code de la page HTML.
Enfin nous renverrons la page modifier au client.

En fonction de la méthode utilisée, le traitement, bien qu'identique, ne se fera pas au même endroit

5.2.3.1 Actualiser une page avec la méthode GET

Dans ce cas la requête apparaît comme cela :

```
GET

/actes.html?Potar=f7e&Switchs=00&Leds=bb&7Seg=7

HTTP/1.1
```

```
GET /actes.html?Potar=f7e&Switchs=00&Leds=bb&7Seg=7 HTTP/1.1
Accept: application/vnd.ms-powerpoint, image/gif, image/x-xbitmap, image/jpeg,
image/pjpeg, application/msword, application/vnd.ms-excel, application/x-shockw
 ve-flash, */*
Referer: http://10.0.0.25/actes.html?Potar=f8a&Switchs=00&Leds=aa&7Seg=a
Accept-Language: fr
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)
Host: 10.0.0.25
Connection: Keep-Alive
```

On s'aperçoit que l'uri comporte les nouvelles valeurs du potentiomètre, des swtichs, et les nouvelles demandes pour les Leds et le 7 segments.
Dans les informations la ligne Referer : nous donne la page précédente, ou l'on trouve les anciennes valeurs du potentiomètre, des swtichs, et les anciennes demandes pour les Leds et le 7 segments.

Lors d'une méthode GET les modifications apparaissent dans l'uri, sous la forme /nomdelapage.html ?modifications .
Il faudra donc faire une distinction entre /nomdelapage.html et /nomdelapage.html? pour se rendre compte si des modification de code doivent être faites.

5.2.3.2 Actualiser une page avec la méthode POST

Dans ce cas la requête apparaît comme cela :

```
POST

/s_es7.html

HTTP/1.1

7seg=8

_
POST /s_es7.html HTTP/1.1
Accept: application/vnd.ms-powerpoint, image/gif, image/x-xbitmap, image/jpeg,
image/pjpeg, application/msword, application/vnd.ms-excel, application/x-shockw
ave-flash, */*
 eferer: http://10.0.0.25/s_es7.html
Accept-Language: fr
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)
Host: 10.0.0.25
Content-Length: 6
Connection: Keep-Alive

7seg=8
```

Cette fois, les valeurs à modifier sont présentent dans le body.
Les informations (la ligne Referer :) ne comprennent pas de modifications.

5.3.4 Mise en œuvre du CGI

Lorsqu 'une modification va être demandée, il faudra récupérer les informations, et les traiter.
Pour ce faire la bibliotheque `Search_String` est mise a disposition dans le fichier eid003_HTML.c :

```
int Search_String(char *ptr1,char *ptr2)
{
char *ptr;
int ret=0;
     (int) ptr=strstr(ptr1,ptr2);
          ret =(ptr-ptr1)+strlen(ptr2);
          if(ret<0) ret=0;
     return ret;
}
```

Cette fonction recherche la deuxième occurrence dans la première.
Si elle est trouvée, la fonction retourne une variable qui représente la position de la fin de chaîne.

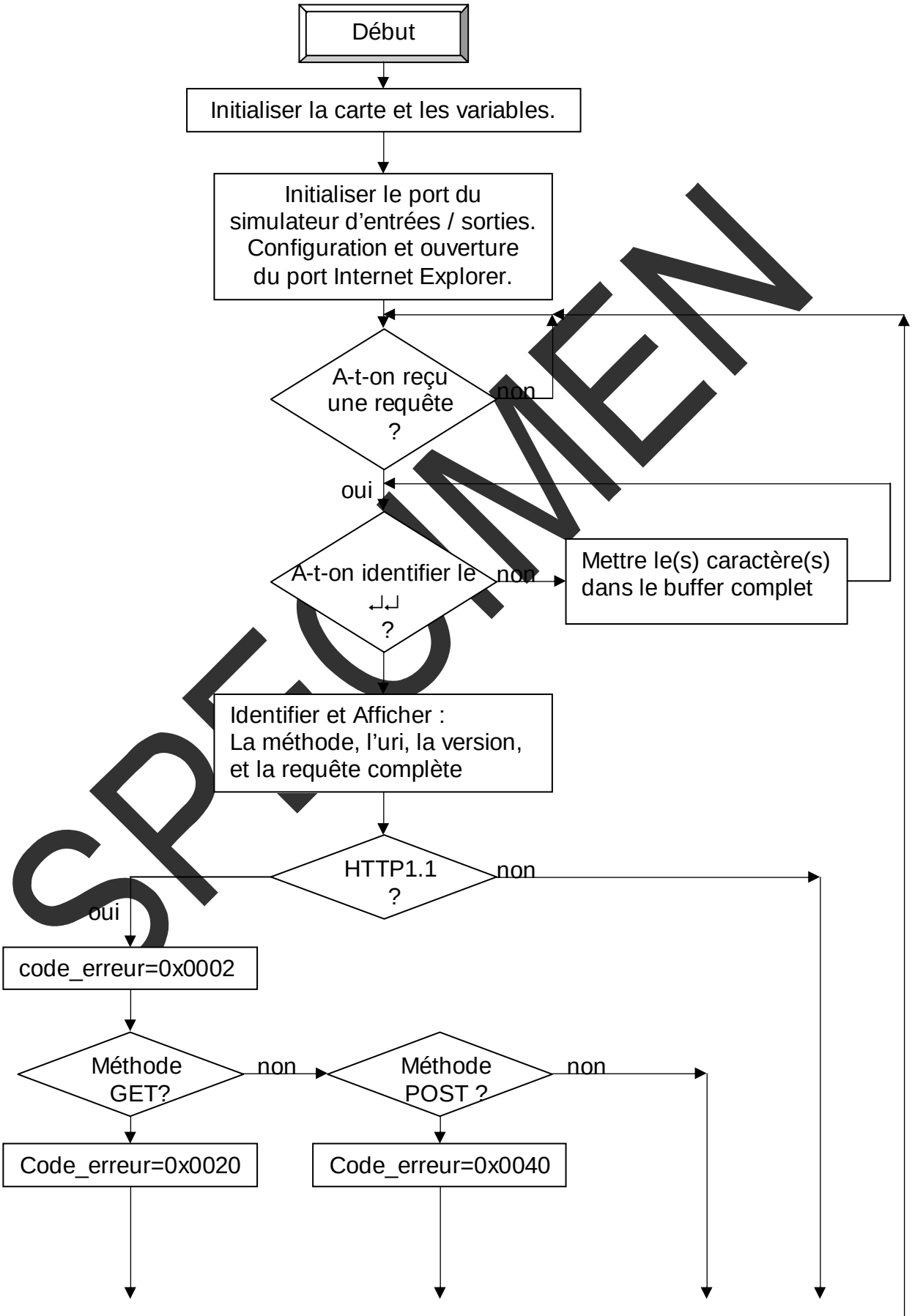
Exemple sur la page actes.html :
Soit le buffer uri :
`/actes.html?Potar=67e&Switchs=ff&Leds=55&7Seg=9`
`place=Search_String(Buffer_reception.complet,"&Leds=");`
La fonction va rechercher `&Leds=` dans le buffer uri.
`&Leds=` est retrouvé, la fonction retournera 39. Donc place=39.
On pourra alors faire `result=Buffer_reception.complet[place];`
On récupéra alors la valeur 9 dans result, et nous pourrons l'afficher sur le 7 segments.
Si `&Leds=` n'avait pas été trouvé, alors place aurait été NULL, et le traitement s'arrête.

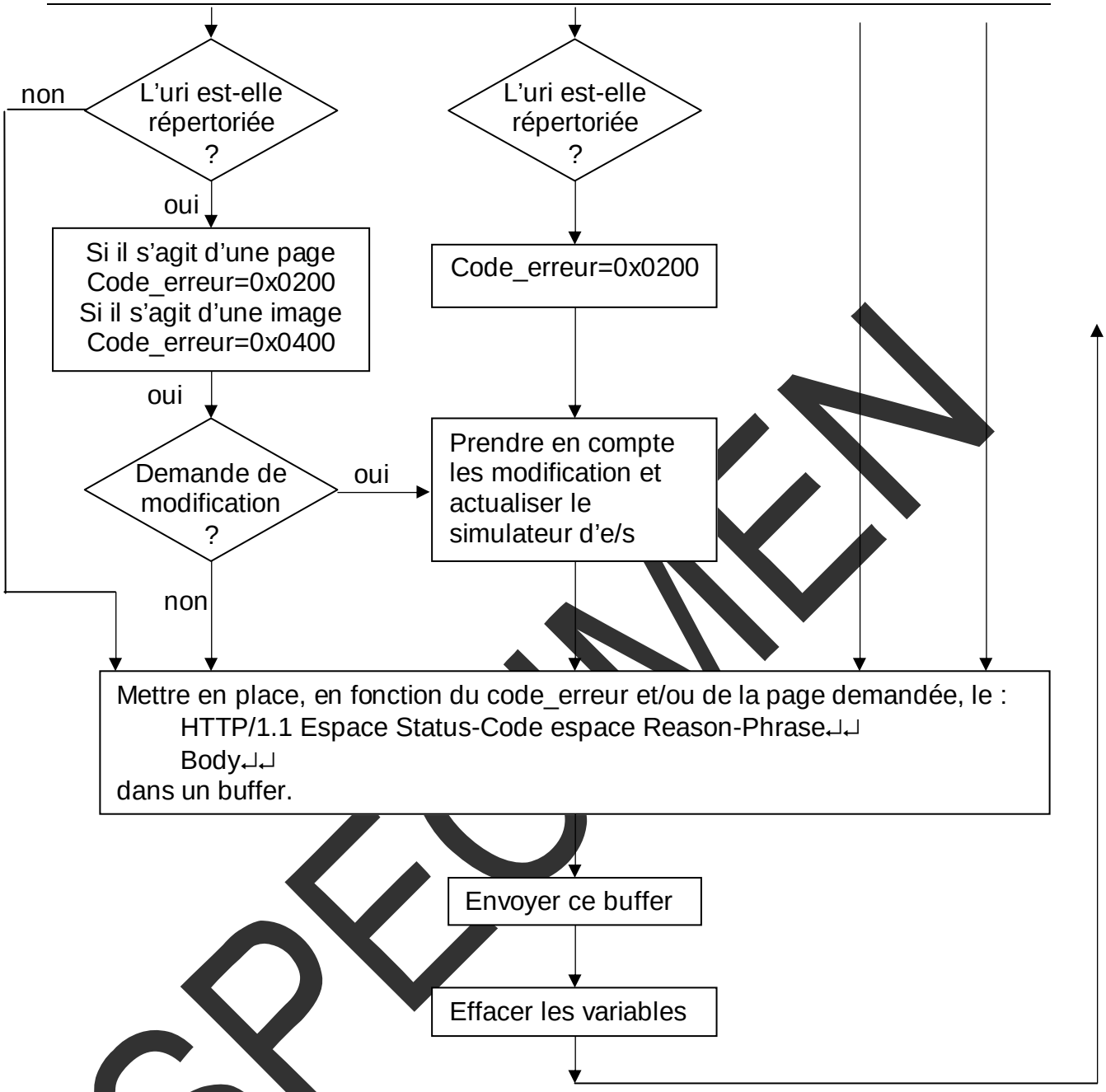
Pour actualiser la valeur des Swtichs, nous pouvons faire :

```
//Capture des switchs et traitement du resultat
result=CaptureSwitch();
j=result&0x0F;
result=(result>>4)&0x0F;
//On modifie le code de la page Actes_HTML
place=Search_String(Actes_HTML,"name=\"Switchs\" value=\"");
//place les switchs de poids fort dans le code de la page HTML
Actes_HTML[place]=tab_ascii[result];
//place les switchs de poids faible dans le code de la page HTML
Actes_HTML[place+1]=tab_ascii[j];
```

Tout ces traitements sont effectués dans le fichier eid003_CGI.c

5.3 Organigramme




L'uri est-elle répertoriée ?
non
oui
Si il s'agit d'une page Code_erreur=0x0200
Si il s'agit d'une image Code_erreur=0x0400
oui
Demande de modification ?
oui
non
L'uri est-elle répertoriée ?
Code_erreur=0x0200
Prendre en compte les modification et actualiser le simulateur d'e/s
Mettre en place, en fonction du code_erreur et/ou de la page demandée, le :
HTTP/1.1 Espace Status-Code espace Reason-Phrase↵↵
Body↵↵
dans un buffer.
Envoyer ce buffer
Effacer les variables

5.4 Programme en C

```
/*****************************************************************************/
/*    Création d'un serveur HTLM                                             */
/*    But du Programme:                                                      */
/*          Realisation d'un serveur HTTP1.1                                 */
/*          Analyse des trames provenant d'Internet Explorer (IE)            */
/*          Reponse en respectant la norme HTTP1.1                           */
/*          Rendre les pages interactives                                    */
/*****************************************************************************/

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

#include "eid210.h"
#include "eid003.h"

#include "eid003_pageHTML.c"

#include "eid003_HTML.c"
#include "eid003_CGI.c"

/***********************************/
/*    Programme principal          */
/***********************************/

main()
{
int tmp,i;
clsscr();

//InitSimES
InitLED();
InitSwitch();
Init7Seg();

//Initialisation des temps d'accès pour la carte EID210
Init_pc104_eid003();

//Configuration + ouverture du port TPC/IP Internet Explorer
SetPort(80);
EID003_CONFIG=0x01;

//Initialisation des varaibles
Init();

//principale
clsscr();
while(CRTL==1)
      {
      while(EID003_RX_COUNT==0);    //Si le buffer est vide, on attend
      clsscr();

      //On réceptionne tout le buffer
      while(0==strstr(Buffer_reception.complet,standar[1]))
      VideBufferRX();         //jusqu'à \r\n\r\n

      code_erreur=0;
```

```
//On identifie la ligne de requete Methode, URI_PAGE, VERSION_HTTP
IndentifieLigneRequete();
//On les affiche
AfficheBuffer(Buffer_reception.methode,2);
AfficheBuffer(Buffer_reception.uri,4);
AfficheBuffer(Buffer_reception.version,6);
AfficheBuffer(Buffer_reception.complet,10);

//Identifie la version HTTP1.1
if(0!=strstr(Buffer_reception.version,version[0]))
      {
      code_erreur=code_erreur|0x0002; //code HTTP1.1

      //Methode GET
      if(0!=strstr(Buffer_reception.methode,methode[0]))
            {
                  code_erreur=code_erreur|0x0020; //code GET
                  //Recherche la page qui a été demané
                  if(0!=strstr(Buffer_reception.uri,pages[0]))//Page index
                        Reponse_Requete(Index_HTML,Reason_200);

                  if(0!=strstr(Buffer_reception.uri,pages[1]))//Page simes
                        Reponse_Requete(Simes_HTML,Reason_200);

                  //Lors d'une demande de modification avec la méthode GET,
                  //la page apparaît sous la forme nompage.html?
                  //Il faut donc vérifier en premier si on a demande la page
                  //pour une modification
                  //Page Actes pour une modification
                  if(0!=strstr(Buffer_reception.uri,pages[9]))
                        {
                        CGI_PageActes();  //Modifier le code de la page
                        Reponse_Requete(Actes_HTML,Reason_200);   //Repondre
                        tmp=strlen(Buffer_reception.uri);//Effacer l'uri
                        for(i=0;i<tmp;i++) Buffer_reception.uri[i]='\0';
                        }
                  if(0!=strstr(Buffer_reception.uri,pages[2]))//Page Actes
                        Reponse_Requete(Actes_HTML,Reason_200);

                  if(0!=strstr(Buffer_reception.uri,pages[10])//Modif S_es2
                        {
                        CGI_PageS_es2();  //Modifier le code de la page
                        Reponse_Requete(S_es2_HTML,Reason_200);   //Repondre
                        tmp=strlen(Buffer_reception.uri);   //Effacer l'uri
                        for(i=0;i<tmp;i++) Buffer_reception.uri[i]='\0';
                        }
                  if(0!=strstr(Buffer_reception.uri,pages[3]))//Page S_es2
                        Reponse_Requete(S_es2_HTML,Reason_200);

                  //Frames, page principale
                  if(0!=strstr(Buffer_reception.uri,pages[4]))//Page S_es3
                        Reponse_Requete(S_es3_HTML,Reason_200);
                  //Frames, composant la page principale
                  if(0!=strstr(Buffer_reception.uri,pages[5]))//Page S_esS
                        Reponse_Requete(S_esS_HTML,Reason_200);
                  if(0!=strstr(Buffer_reception.uri,pages[6]))//Page S_esP
                        Reponse_Requete(S_esP_HTML,Reason_200);
                  if(0!=strstr(Buffer_reception.uri,pages[7]))//Page S_esL
                        Reponse_Requete(S_esL_HTML,Reason_200);
                  if(0!=strstr(Buffer_reception.uri,pages[8]))//Page S_es7
                        Reponse_Requete(S_es7_HTML,Reason_200);
```

```
                //La page s'actualise toutes les 2 secondes
                if(0!=strstr(Buffer_reception.uri,pages[11]))
                      {
                      CGI_PageAutoActu();
                      Reponse_Requete(Auto_Actu_HTML,Reason_200);
                      }
                //IMAGES
                if(0!=strstr(Buffer_reception.uri,images[0]))//image DMS
                Reponse_Image(logoDMS_JPG,Reason_200,logoDMS_JPG_taille);
                if(0!=strstr(Buffer_reception.uri,images[1]))//image
                Reponse_Image(dmsbat_JPG,Reason_200,dmsbat_JPG_taille);

                //Son
                if(0!=strstr(Buffer_reception.uri,son[0]))
                      Reponse_Image(demo_MP3,Reason_200,demo_MP3_taille);

                //demande d'ajout dans les favoris
                if(0!=strstr(Buffer_reception.uri,pages[12]))
                      Reponse_Requete(Erreur_HTML,Reason_200);
          }

     //Methode POST
     if(0!=strstr(Buffer_reception.methode,methode[1]))
          {
          code_erreur=code_erreur|0x0040; //code POST

          //Si le Content-Length est present
          if(0!=strstr(Buffer_reception.complet,"Content-Length: "))
                {
                //=> il faut recuperer le body venant de IE
                 while(EID003_RX_COUNT!=0) VideBufferRX();
                //Indentifier le body et le mettre dans son buffer
                 IndentifieLigneBody(Buffer_reception.body);
                 AfficheBuffer(Buffer_reception.body,8);
                }

                if(0!=strstr(Buffer_reception.uri,pages[5]))//Page S_esS
                      {
                      CGI_PageS_esS();
                      Reponse_Requete(S_esS_HTML,Reason_200);
                      }
                if(0!=strstr(Buffer_reception.uri,pages[6]))//Page S_esP
                      {
                      CGI_PageS_esP();
                      Reponse_Requete(S_esP_HTML,Reason_200);
                      }
                if(0!=strstr(Buffer_reception.uri,pages[7]))//Page S_esL
                      {
                      CGI_PageS_esL();
                      Reponse_Requete(S_esL_HTML,Reason_200);
                      }
                if(0!=strstr(Buffer_reception.uri,pages[8]))//Page S_es7
                      {
                      CGI_PageS_es7();
                      Reponse_Requete(S_es7_HTML,Reason_200);
                      }
          }
     }
```

```
            else  //Ce n'est pas la version HTTP1.1 on en informe l'utilisateur
            {     //Sinon on veut on peut traiter d'autres version ici
                  //et faire code_erreur=code_erreur|0x0004ou8.
            printf("Il ne s'agit pas de la version HTTP1.1\n");
            }

      Analyse_code_erreur(code_erreur); //Analyse les erreurs
      Init();//Efface les variables de reception,d'identification et de reponses
      }
}
```