TP N° 1 - Liaison série RS-232

Vous rédigerez un compte rendu, sur lequel vous indiquerez la réponse à chaque question, vos explications et commentaires (interprétation du résultat), et le cas échéant la ou les commandes utilisées.

1 L'interface d'entrées/sorties séries asynchrones

L'interface entrées/sorties séries équipe tous les PC et permet l'échange d'informations à faible débit avec un périphérique comme un modem, ou avec un autre PC, sur des distances inférieures à quelques dizaines de mètres.

1.1 Pourquoi une transmission série ?

Sur des distances supérieures à quelques mètres, il est difficile de mettre en œuvre une transmission en parallèle : coût du cablage, mais surtout interférences électromagnétiques entre les fils provoquant des erreurs importantes. On utilise alors une liaison série, avec un seul fil portant l'information dans chaque sens.

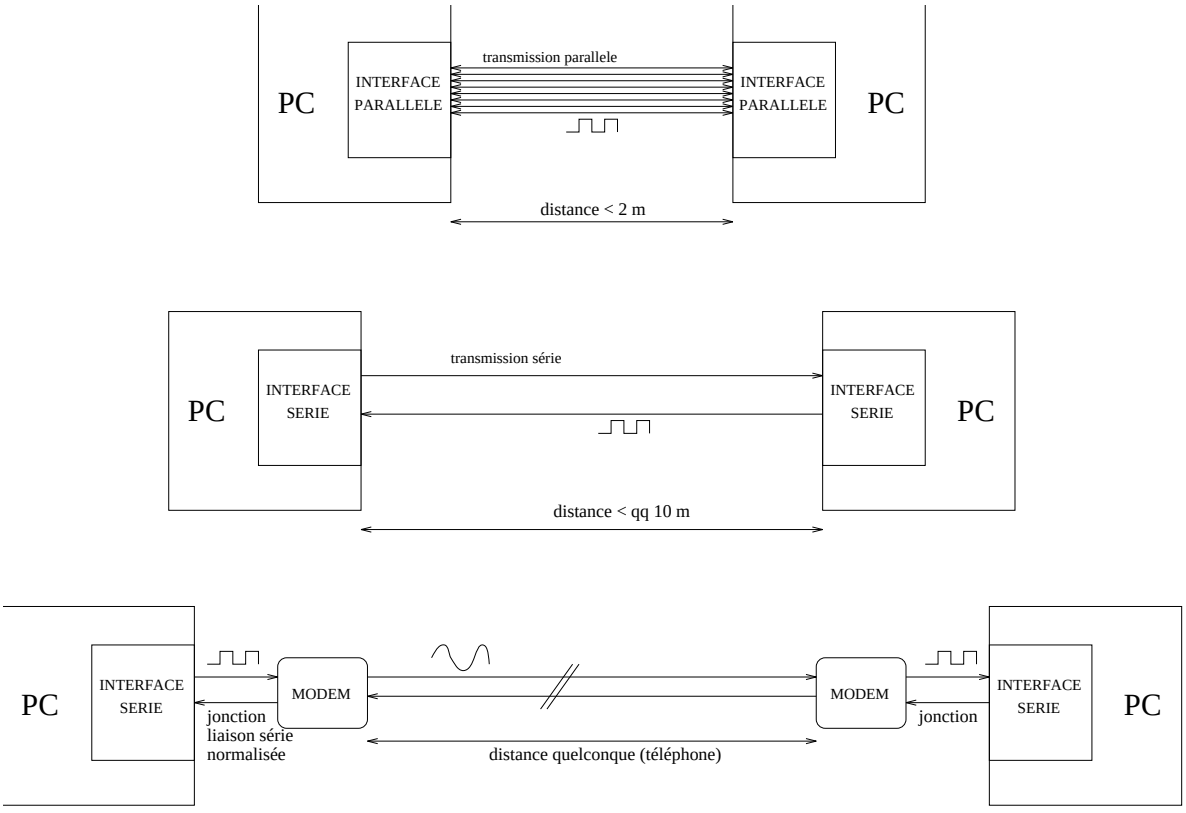

FIG. 1 – Différents types de transmissions pour relier simplement deux PC.

1.2 Principe de la transmission série asynchrone

En l'abscence de transmission, le niveau de la liaison est 1 (niveau de repos).

Les bits sont transmis les un après les autres, en commençant par le bit de poids faible b_0. Le premier bit est précédé d'un bit *start* (niveau 0). Après le dernier bit, on peut transmetre un bit de parité (voir cours de réseaux), puis un ou deux bits *stop* (niveau 1).

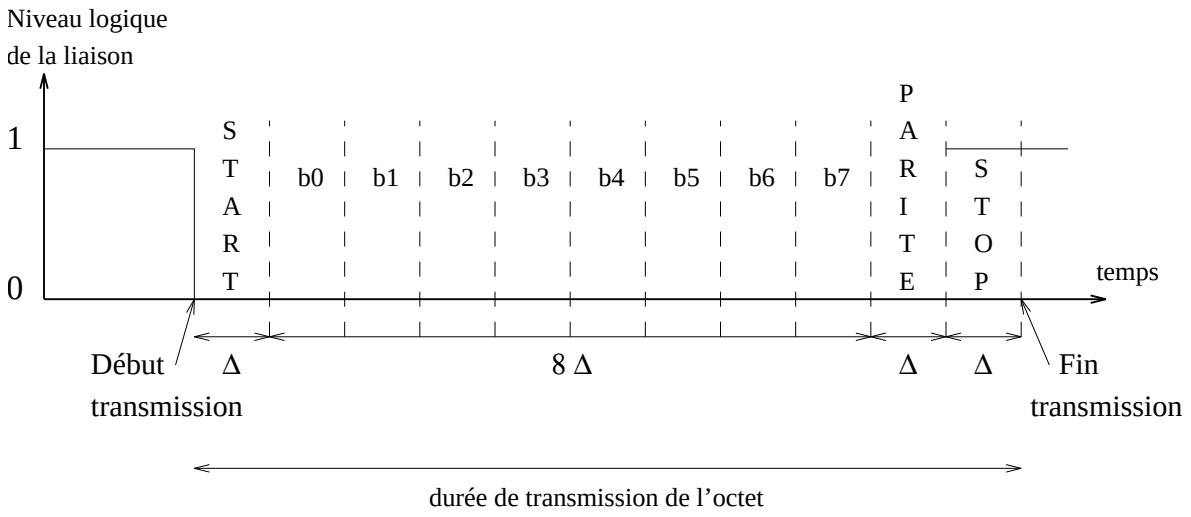

FIG. 2 – Transmission d'un octet $b_7b_6b_5b_4b_3b_2b_1b_0$ en série.

Chaque bit a une durée de Δ, qui fixe le débit transmission. Le nombre de changements de niveaux par seconde est appelé *rapidité de modulation* (RM), et s'exprime en Bauds (du nom de Baudot, l'inventeur du code TELEX). On a

$$RM = \frac{1}{\Delta}$$

et aussi

$$\text{débit binaire} = \frac{1}{\Delta}\text{bits/s}$$

Le récepteur détecte l'arrivée d'un octet par le changement de niveau correspondant au bit *start*. Il échantillonne ensuite chaque intervalle de temps Δ au rythme de son horloge.

Comme les débits binaires de transmission série de ce type sont faibles (< 19600 bits/s) et que les horloges de l'émetteur et du récepteurs sont suffisament stables (horloges à quartz), il n'est pas nécessaire de les synchroniser. C'est la raison pour laquelle ce type de transmission série est qualifié d'*asynchrone*.

Lorsque les débits sont plus importants, la dérive des horloges entrainerait des erreurs, et on doit mettre en œuvre une transmission *synchrone* (voir cours de réseaux).

1.3 L'interface d'E/S séries 8250

Le composant électronique chargé de la gestion des transmissions séries asynchrones dans les PC est appelé UART (*Universal Asynchronous Receiver Transmitter*). Nous décrivons ici le circuit Intel 8250.

1.3.1 Bornes de l'interface

Les bornes de l'interface UART 8250 sont présentées sur la figure 3. Seules les bornes essentielles à la compréhension du fonctionnement de l'interface sont représentées.

1.3.2 Les registres du 8250

L'interface 8250 possède un certain nombre de registres de 8 bits permettant de gérer la communication. Ces registres sont lus et modifiés par le processeur par le biais des instructions IN et OUT vues plus haut.

L'adresse de chaque registre est donnée par l'adresse de base de l'interface (fixée une fois pour toute) à laquelle on ajoute une adresse sur 3 bits $A_2A_1A_0$.

Une complication supplémentaire est introduite par le fait qu'il y a plus de 8 registres différents et que l'on ne dispose que de 3 bits d'adresse. La solution est d'utiliser un bit d'un registre spécial, DLAB.

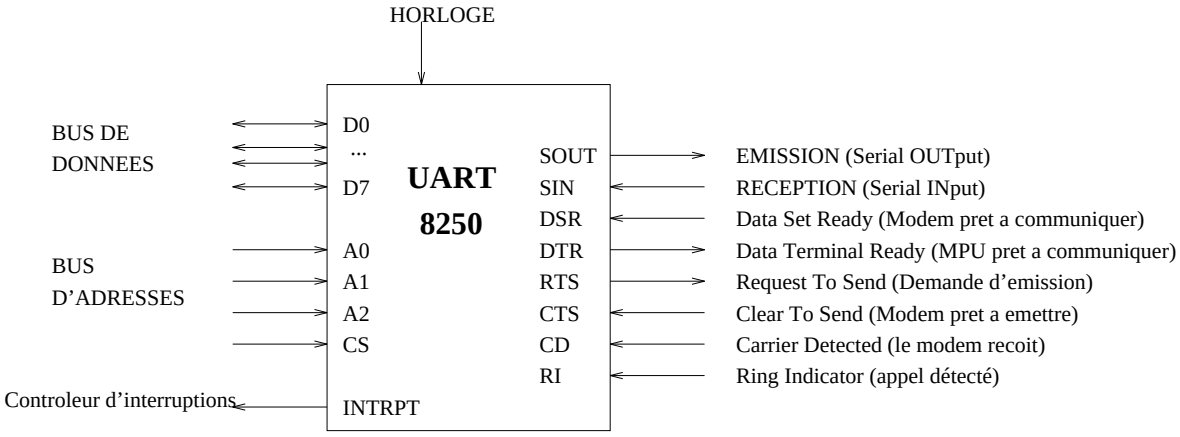

FIG. 3 – Bornes du circuit UART 8250.

Suivant l'état du bit DLAB, on va sélectionner tel ou tel jeux de registres.

La table suivante donne l'adresse et le nom de chaque registre du 8250 :

Adresse				REGISTRES
DLAB	A2	A1	A0	
0	0	0	0	RBR : Receiver Buffer (registre de réception)
0	0	0	0	THR : Transmitter Holding Register (registre d'émission)
1	0	0	0	DLL : Divisor Latch LSB (poids faible diviseur horloge)
1	0	0	1	DLM : Divisor Latch MSB (poids fort diviseur horloge)
0	0	0	1	IER : Interrupt Enable Register
x	0	1	0	IIR : Interrupt Identification Register
x	0	1	1	LCR : Line Control Register
x	1	0	0	MCR : Modem Control Register
x	1	0	1	LSR : Line Status Register
x	1	1	0	MSR : Modem Status Register
x	1	1	1	non utilisé

Exemple : si l'adresse de base de l'interface est par exemple `3F8H`, l'adresse du registre RBR sera simplement

```
3F8H + 000 = 3F8H
```

et celle de IIR

```
3F8H + 010b = 3F8H + 2H = 3FAH.
```

On voit, comme nous l'avons dit plus haut, que les registres DLM et IER (par exemple) possèdent la même adresse (`001b`). Si le bit DLAB est 0, cette adresse permet d'accéder à DLM, et si DLAB=1 à IER.

Le bit DLAB est le bit de poids fort du registre LCR.

Notons aussi que THR et RBR ont la même adresse, ce qui n'est pas gènant car on accède toujours en *écriture* à THR et en *lecture* à RBR.

Voyons maintenant comment utiliser ces différents registres pour programmer l'interface de transmission série.

1.3.3 Choix de la rapidité de modulation

L'horloge de référence du 8250 est un signal à 1,8432 MHz stabilisé par un quartz.

Une première division de cette fréquence par 16 est effectuée dans l'interface. La fréquence obtenue est ensuite divisée par un diviseur codé sur 16 bits et contenu dans la paire de registres DLM (poids fort), DLL (poids faible).

On modifie donc le débit de transmission en changeant les valeurs de DLM et DLL.

Le tableau suivant donne les valeurs à utiliser pour les rapidités de modulation courantes :

Modulation (bauds)	Diviseur (hexa)
50	0900H
75	0600H
110	0417H
300	0180H
600	00C0H

Modulation (bauds)	Diviseur (hexa)
1200	0060H
2400	0030H
4800	0018H
7200	0010H
9600	000CH

1.3.4 Registre THR

C'est le registre d'émission. Il est chargé par le MPU en exécutant l'instruction

```
OUT THR, AL
```

(où THR est une constant initialisée avec l'adresse du registre THR).

Le contenu de THR est automatiquement copié par l'interface dans le registre à décalage d'émission, qui permettra la sortie en série des bits sur la sortie SOUT.

1.3.5 Registre RBR

C'est le registre de réception. Les bits qui arrivent en série sur la borne SIN du circuit entrent dans un registre à décalage. Lorsque ce dernier est complet, il est transféré dans RBR.

Le MPU peut lire le contenu de RBR avec l'instruction

```
IN AL, RBR
```

Notons que si un deuxième octet arrive avant que RBR n'ait été lu par le MPU, il remplace RBR : on perd le premier arrivé, c'est l'erreur d'*écrasement*[1].

1.3.6 Registre LCR (Line Control Register)

Ce registre de commande permet de définir certains paramètres de la transmission, en fonction de sa configuration binaire.

Bits 0 et 1 : spécifient le nombre de bits de la donnée à transmettre (caractères de 5, 6, 7 ou 8 bits) :

Bit 1	Bit 0	Nb de bits/caractère
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

Bit 2 : spécifie le nombre de bits STOP ($0 \rightarrow 1$ stop, $1 \rightarrow 2$ stops).

Bit 3 : spécifie la présence (si 1) ou l'absence (si 0) d'un bit de contrôle d'erreur (type bit de parité).

Bit 4 : s'il y a un bit de contrôle d'erreur, le bit 4 spécifie s'il s'agit d'un bit de parité paire (si 1) ou impaire (si 0).

Bit 5 : normalement à 0.

Bit 6 : normalement à 0.

Bit 7 : bit DLAB, permettant l'accès aux registres DLL et DLM dans la phase d'initialisation de l'interface.

[1] Le même type d'erreur peut se produire en émission si le processeur écrit THR avant que le caractère précédent ait été transféré.

1.3.7 Registre IER

Ce registre est utilisé pour les entrées/sorties par *interruption*.

Bit 0 : interruption lorsque donnée reçue dans RBR ;

Bit 1 : interruption lorsque registre THR vide ;

Bit 2 : interruption lorsque l'un des 4 bits de poids faible de LSR passe à 1 ;

Bit 3 : interruption sur état du modem.

1.3.8 Registre LSR

Ce registre d'état rend compte du fonctionnement de la liaison en réception (bits 0 à 4) et en émission (bits 5 et 6).

Bit 0 : passe à 1 lorsqu'une donnée a été reçue et chargée dans RBR.

Bit 1 : signale erreur d'écrasement. 1 si donnée reçue et chargée dans RBR alors que la précédente n'avait pas été lue. Remis automatiquement à 0 par la lecture du registre LSR.

Bit 2 : passe à 1 à la suite d'une erreur de parité. Remis à 0 par la lecture de LSR.

Bit 3 : passe à 1 lorsque le niveau du bit STOP n'est pas valide (erreur de format). Remis à 0 par la lecture de LSR.

Bit 4 : passe à 1 lorsque le niveau de la liaison est resté à 0 pendant la durée d'émission de la donnée (problème de l'emetteur). Remis à 0 par la lecture de LSR.

Bit 5 : passe à 1 lorsqu'une donnée est transférée de THR vers le registre à décalage d'émission (THR est alors libre pour le caractère suivant).

Bit 6 : passe à 1 lorsque le registre à décalage d'émission est vide.

1.3.9 Registre IIR

IIR est utilisé pour les E/S par interruption. Son contenu permet d'identifier la cause de l'interruption émise par l'interface 8250.

1.3.10 Registre MCR

MCR est le registre de commande du modem.

Bit 0 : commande le niveau de la borne DTR qui informe le modem que le MPU est prêt à communiquer ;

Bit 1 : commande la borne RTS qui demande au modem d'émettre.

1.3.11 Registre MSR

MSR est le registre d'état du fonctionnement du modem.

1.4 Normes RS-232 et V24

Ces normes spécifient les caractéristiques *mécaniques* (les connecteurs), *fonctionnelles* (nature des signaux) et *électriques* (niveaux des signaux) d'une liaison série asynchrone avec un longueur maximale de 15m et une rapidité de modulation maximum de 20kbauds.

L'EIA (*Electrical Industry Association*) a été à l'origine aux USA de la norme RS-232, dont la dernière version est RS-232C. Le CCITT (Comité Consultatif International pour la Téléphonie et la Télégraphie) a repris cette norme qu'il a baptisé V24.

Deux autres normes permettent des débits plus élevés et des distances plus importantes : RS-423 (666m, 300kbauds), et RS-422 (1333m, 10Mbauds).

La norme V24 utilise le connecteur DB25, de forme trapézoïdale à 25 broches, représenté figure 4.

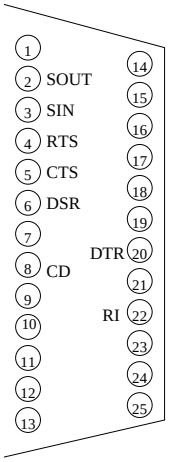

FIG. 4 – Connecteur DB25, avec les bornes correspondantes du circuit UART 8250.

Les niveaux électriques des bornes 2 et 3 (signaux d'information) sont compris entre +3V et +25V pour le niveau logique 0, et -3V et -25V pour le niveau logique 1 (niveau de repos).

1.4.1 Cable NULL-MODEM

On peut connecter deux PC par leur interface série. Si la distance est courte (< quelques dizaines de mètres), il n'est pas nécessaire d'utiliser un modem. On utilise alors un cable *Null-Modem*, qui croise certains signaux comme le montre la figure 5.

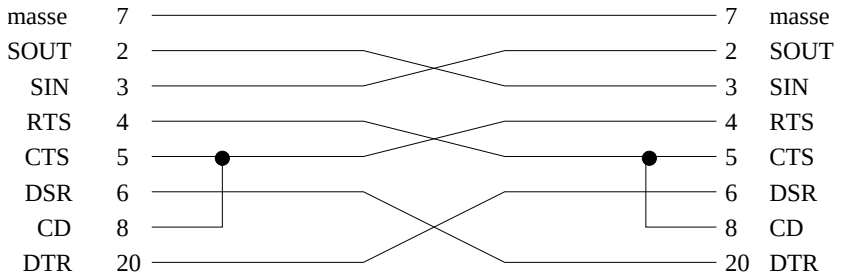

FIG. 5 – Cable Null Modem complet.

Lorsque les signaux de dialogues ne sont pas nécessaires, il suffit de croiser les signaux SIN et SOUT, ce qui donne le cable Null Modem simplifié (3 fils) représenté sur la figure 6.

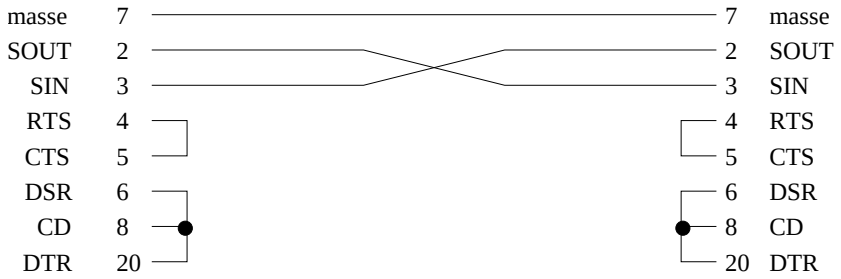

FIG. 6 – Cable Null Modem simplifié.

1.5 Programmation de l'interface en langage C

Nous avons vu que les deux instructions assembleur permettant au processeur de communiquer avec les interfaces d'entrées/sorties étaient IN et OUT.

Si l'on utilise un système d'exploitation multi-tâche (par ex. Windows XP, NT etc ou les variantes d'UNIX), ces instructions ne sont en général pas utilisables par les programmes utilisateurs : le système d'exploitation protège les accès au matériel. Par contre, le système offre des moyens d'accéder aux ports séries via des *appels systèmes* standardisés.

1.5.1 Accès sous DOS

Le système DOS est un ancien système d'exploitation très primitif, mais qui présente l'avantage de permettre l'accès direct au matériel (en effet, DOS n'est ni multi-tâche, ni multi-utilisateur).

Sous DOS, il est possible d'accéder simplement à ces instructions en langage C, grâce aux fonctions `inportb( adr )` et `outportb( adr )`.

- `unsigned char inportb( int address )`
  lit un octet à l'adresse (de l'espace d'entrées/sorties) indiquée et le retourne.
- `void outportb( int address, unsigned char *data )`
  écrit l'octet (argument `data`) à l'adresse (E/S) indiquée.

Voici un exemple de configuration de l'interface 8250 en langage C sous DOS. On configure ici l'interface pour un débit de 300 bauds, en mode scrutation, parité paire, 1 bit stop, 8 bits de données :

```
#include <dos.h>

/* Quelques constantes pour ameliorer la lisibilite:
 */
#define PORT (0x3F8) /* adresse de l'interface */
#define RBR  PORT
#define THR  PORT
#define LSR  (PORT+5)
#define IIR  (PORT+2)
#define LCR  (PORT+3) /* DLAB ... */
#define DLL  PORT     /* DLAB = 1 */
#define DLM  (PORT+1) /* DLAB = 1 */
#define IER  (PORT+1)
#define MCR  (PORT+4)
#define MSR  (PORT+6)

/* Initialise l'interface 8250
 */
void init_8250(void) {
  /* 1- Rapidite de modulation */
  outportb( LCR, 0x80 ); /* DLAB = 1 */
  outportb( DLM, 0x01 );
  outportb( DLL, 0x80 ); /* 300 bauds */

  /* 2- Format des donnees
   *  DLAB = 0, parite paire, 1 stop, 8 bits
   *   LCR = 00011011  = 1BH
   */
  outportb( LCR, 0x1B );

  /* 3- Mode de transfert: scrutation */
  outportb( IER,  0 );
}
```

1.5.2 Accès sous Linux

Sous Linux, l'interface d'entrées/sorties séries est gérée par un pilote (module) spécialisé, accessible via le *device* `/dev/ttyS`n, où n est le numéro de l'interface (la plupart des PC ont deux interfaces, `/dev/ttyS0` et `/dev/ttyS1`, qui correspondent aux ports `COM1` et `COM2` sous DOS).

La fonction importante est `tcsetattr`, qui permet de fixer les caractéristiques de la liaison : rapidité de modulation, nombre de bits, etc.

La communication s'effectue via les appels systèmes `read` (réception) et `write` (émission).

2 Travaux pratiques

Vous travaillerez sous Linux en salle de TP réseaux (image GTR-4). Des cables NULL-modem seront fournis.

1- Récupérer, compiler et tester les programmes "recepteur" et "emetteur" fournis par votre enseignant.

2- Modifier les programmes émetteur et récepteur pour envoyer et afficher une chaîne de caractère (constante ou saisie au clavier par l'émetteur). Comment repérer la fin de la chaîne ?

3- Faire envoyer 10 Ko ou 100 Ko d'une machine à l'autre, et mesurer le temps de transfert. Répéter la mesure avec les différents débits de la liaison série et conclure (tracer une courbe temps = f(débit nominal)).

2.1 Les programmes C / Linux

```
/***************************************************************************\

file name......: interface.c
description....: Paramétrage port série

authors........: Emmanuel Viennet pour GEII

remarks........:
history........: 17 nov 2008 creation
\***************************************************************************/

#include <termios.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/signal.h>
#include <sys/types.h>

#define PORTDEVICE "/dev/ttyS0"



int open_interface( long Baud_Rate, int Data_Bits, int Stop_Bits, int Parity) {
  long BAUD;
  long DATABITS;
  long STOPBITS;
  long PARITYON;
  long PARITY;
  int fd; /* descripteur de fichier */
  struct termios tio; /* parametres de la liaison, voir man tcsetattr */
  fprintf(stderr, "Ouverture interface\n");
  fprintf(stderr, "%ld bauds, %d bits/caractere, %d stops, parite %d\n",
  Baud_Rate, Data_Bits, Stop_Bits, Parity);

  switch (Baud_Rate) {
  case 38400:
  default:
```

```
  BAUD = B38400; /* constantes definies dans termios.h */
  break;
case 19200:
  BAUD  = B19200;
  break;
case 9600:
  BAUD  = B9600;
  break;
case 4800:
  BAUD  = B4800;
  break;
case 2400:
  BAUD  = B2400;
  break;
case 1800:
  BAUD  = B1800;
  break;
case 1200:
  BAUD  = B1200;
  break;
case 600:
  BAUD  = B600;
  break;
case 300:
  BAUD  = B300;
  break;
case 200:
  BAUD  = B200;
  break;
case 150:
  BAUD  = B150;
  break;
case 134:
  BAUD  = B134;
  break;
case 110:
  BAUD  = B110;
  break;
case 75:
  BAUD  = B75;
  break;
case 50:
  BAUD  = B50;
  break;
}  /* end of switch baud_rate */

switch (Data_Bits) {
  case 8:
  default:
    DATABITS = CS8;
    break;
  case 7:
    DATABITS = CS7;
    break;
  case 6:
    DATABITS = CS6;
    break;
  case 5:
    DATABITS = CS5;
    break;
  }  /* end of switch data_bits */

switch (Stop_Bits) {
  case 1:
  default:
    STOPBITS = 0;
    break;
  case 2:
    STOPBITS = CSTOPB;
    break;
  }  /* end of switch stop bits */

switch (Parity) {
  case 0:
  default:                        /* aucun */
    PARITYON = 0;
    PARITY = 0;
```

```
      break;
    case 1:                         /* impair (odd) */
      PARITYON = PARENB;
      PARITY = PARODD;
      break;
    case 2:                         /* pair (even) */
      PARITYON = PARENB;
      PARITY = 0;
      break;
  }  /* end of switch parity */

  /* Ouvre le port, I/O bloquantes  */
  fd = open(PORTDEVICE, O_RDWR | O_NOCTTY );
  if (fd < 0) {
    perror(PORTDEVICE);
    exit(-1);
  }
  fcntl(fd, F_SETFL, 0); /* verifier utilite */

  /* Parametres de la liaison serie: */
  tio.c_cflag = BAUD | CRTSCTS | DATABITS | STOPBITS | PARITYON | PARITY | CLOCAL | CREAD;
  tio.c_iflag = IGNPAR;
  tio.c_oflag = 0;
  tio.c_lflag = 0;
  tio.c_cc[VMIN]=1;
  tio.c_cc[VTIME]=0;
  tcflush(fd, TCIFLUSH); /* purge donnees non lues ou non envoyees */

  tcsetattr(fd, TCSANOW, &tio); /* fixe parametres liaison */

  return fd;
}

void close_interface(int fd) {
  fprintf(stderr, "fermeture interface\n");
  close(fd);
}
```

```
/*************************************************************************\

file name......: emetteur.c
description....: Paramétrage port série en envoi d'une chaine de caractères

authors........: Emmanuel Viennet pour GEII

remarks........:
history........: 17 nov 2008 creation
\*************************************************************************/

#include <termios.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/signal.h>
#include <sys/types.h>

#include "interface.h"

int main(void) {
  char car = 'X';
  int port = open_interface( 9600, 8, 1, 1 );
  /* port est un descripteur de fichier,
     a utiliser avec les appels read() et write()
     Voir man 2 write
  */

  write(port, &car, 1); /* ecriture d'un caractere sur le port */

  close_interface(port);

  fprintf(stderr, "fin emetteur\n");
  return 0; /* The end */
}
```

```
/*****************************************************************************\

file name......: recepteur.c
description....: Parametrage port série et reception d'un caractere

authors........: Emmanuel Viennet pour GEII

remarks........:
history........: 17 nov 2008 creation
\*****************************************************************************/

#include <termios.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/signal.h>
#include <sys/types.h>

#include "interface.h"

int main(void) {
  char car;
  int port = open_interface( 9600, 8, 1, 1 );
  int n=0;
  /* port est un descripteur de fichier,
     voir man 2 read
  */

  sleep(2); /* temporisation pour faciliter les manips */

  fprintf(stderr, "attente reception");

  n = read(port, &car, 1); /* lecture (avec attente) d'un caractere sur le port */
  if (n < 0) {
    perror("read error"); /* affiche le message d'erreur */
    exit(1); /* termine */
  }

  printf("Lu caractere '%c'\n", car );

  close_interface(port);

  fprintf(stderr, "fin recepteur\n");
  return 0; /* The end */
}
```