# TD ASR2 Réseau

## Transmission asynchrone ~ Contrôle de flux

On s'intéresse à la transmission asynchrone entre deux ordinateurs munis chacun d'un port série. Les caractéristiques de la liaison seront les suivantes :

- données codées sur 8 bits,
- parité paire,
- 1 bit START, 2 bits STOP,
- 9600 bps.

### 1. Câblage

On désire utiliser un câble avec 3 fils. Donner le câblage de la jonction entre les 2 ordinateurs.

### 2. Signal

Quel est le nombre de bits transférés pour une donnée de 8 bits ? Donnez le signal envoyé pour l'envoi de l'octet 10101010.

### 3. Asynchronisme

Quelle est la durée qui sépare l'émission de deux caractères ?

### 4. Bibliothèque « Transmission Asynchrone »

Un module de transmission asynchrone contient les primitives suivantes :

```
Action Initialise(Entrée Config : Octet
                         NuméroPort : Octet);
```

Cette action permet d'initialiser le port série *NuméroPort* (0,1,...) avec la configuration *Config*. Les bits de l'octet *Config* ont la signification suivante :

| Bits | Description | Valeur | | | |
|---|---|---|---|---|---|
| 0..1 | Nombre de bits de données | 10 | : 7 bits | 11 | : 8 bits |
| 2 | Nombre de bits STOP | 0 | : 1 bit | 1 | : 2 bits |
| 3..4 | Parité | 00 | : pas de parité | 01 | : impaire |
| | | 10 | : pas de parité | 11 | : paire |
| 5..7 | Vitesse | 000 | : 111 bps | 001 | : 150 bps |
| | | 010 | : 300 bps | 011 | : 600 bps |
| | | 100 | : 1200 bps | 101 | : 2400 bps |
| | | 110 | : 4800 bps | 111 | : 9600 bps |

```
Action EmissionCaractère (Entrée Caract : Octet
                                 NuméroPort : Octet);
```

Cette action envoie le caractère *Caract* sur le port série *NuméroPort*.

```
Action RéceptionCaractère (Entrée  NuméroPort : Octet
                           Sortie  Caract : Octet);
```

Cette action permet de recevoir un caractère sur le port série *Numéroport*.

#### *4.1. Action Emetteur*

Ecrire une action *Emetteur* qui configure le port série 1 et qui envoie, sur ce même port, une suite de *P* caractères. (*P* constante prédéfinie)

#### *4.2. Action Récepteur*

Ecrire une action *Récepteur* qui configure le port série 1 et qui lit, sur ce même port, *P* caractères. Les caractères seront affichés sur l'écran, au fur et à mesure de leur arrivée.
[Dans les 2 questions précédentes, on effectue seulement la transmission de données, sans se préoccuper des phases d'ouverture et de fermeture de la liaison…]

### 5. Transfert de fichier – Temps d'émission

Donner une solution permettant le transfert d'un fichier. Quel est le temps minimal nécessaire pour le transfert d'un fichier de 120K ?

### 6. Situation d'erreurs

Expliquez pourquoi les situations critiques suivantes peuvent avoir lieu lorsqu'on effectue un transfert en utilisant les actions *Emetteur* et *Récepteur* :

a) perte de caractères lors de la réception,
b) perte de caractères lors de l'émission.

### 7. Contrôle de flux

On se propose d'éviter les situations critiques de "type a" en utilisant une file circulaire de type FIFO (First In First Out) de la façon suivante : sur le site récepteur, l'arrivée d'un caractère sur le port série va déclencher la routine d'interruption *InterruptSérie* qui déposera le caractère dans une file circulaire. On dispose alors d'un module *Async_Bufferisée* avec les primitives suivantes :

```
Action Initialise_Bufférisée(Entrée  Config : Octet
                                     NuméroPort : Octet);
```

(Idem à *Initialise*)

```
Action EmissionCaractère_Bufferisée(Entrée Caract : Octet
                                           NuméroPort : Octet);
```

(Idem à *EmissionCaractère*)

```
Fonction NombreCaractères(Entrée  NuméroPort : Octet) : entier;
```

Cette fonction retourne le nombre de caractères contenus dans la file circulaire associée au port *NuméroPort*.

```
Action RéceptionCaractère_Bufferisée(Entrée   NuméroPort : Octet
                                     Sortie   Caract : Octet);
```

Cette action fournit dans *Caract* le caractère en tête de la file circulaire associée au port *Numéroport* (et la file circulaire est mise à jour). A utiliser lorsque la file est non vide.

Donner les nouvelles versions des actions *Emetteur* et *Récepteur*. On ne demande pas d'écrire :

- la routine *InterruptSérie* de traitement de l'interruption série,
- les primitives du module *Async_Bufferisée*.

### 8. Gestion d'une transmission asynchrone

Cette solution n'est pas suffisante pour éviter toutes les situations de "types a". Expliquez pourquoi ? Donnez brièvement les idées d'une solution correcte.