UART implémenté de façon logicielle

Tous documents papiers autorisés. **Aucun** support électronique (téléphone, ordinateur portable ...) visible durant l'examen. Chaque question sur 1 point.

Il arrive que lors d'un développement de plateforme embarquée, le nombre de prériphériques croisse jusqu'à épuiser tous les ports de communication du microcontrôleur. La solution qui impose de changer de plateforme matérielle nécessite de cabler une nouvelle carte électronique : en attendant que le matériel soit livré, nous voulons continuer à développer notre prototype en implémentant de façon logicielle un USART permettant la communication asynchrone avec un périphérique. C'est dans ce contexte que se place l'examen.

1. Rappeler la différence entre un protocole synchrone et asynchrone. Pourquoi un protocole synchrone est-il simple à implémenter de façon logicielle ?
2. Quel périphérique d'un microcontrôleur permet d'assurer le cadencement des bits transmis dans un protocole asynchrone ?
3. Comment identifier la valeur du bit de poids faible d'un octet ? Fournir le code C associé.
4. Comment identifier successivement tous les bits, un à un, d'un octet ? Proposer un code C associé.
5. Nous désirons transmettre un octet par protocole compatible RS232 en encodage 8E1 [1] : proposer une procédure qui génère une séquence de signaux compatibles RS232, permettant l'envoi d'un octet donné en entrée, en supposant que nous avons à notre disposition deux fonctions pour manipuler une broche de GPIO en sortie nommées `gpio_up()` pour passer la broche au niveau haut et `gpio_down()` pour la passer au niveau bas. On rappelle que RS232 transmet le bit de poids faible en premier.
6. Proposer un cadencement de cette procédure par le périphérique mentionné à la question 2 pour respecter la contrainte de transmission de ce protocole.
7. Ayant résolu le problème de l'émission des messages, notre interlocuteur veut nous répondre et nous avons besoin d'écouter une transaction RS232. En supposant que nous ayons une fonction `gpio_in()` qui renvoie la valeur d'un autre GPIO utilisé en entrée, proposer la séquence d'opérations pour lire les bits de données et le bit de parité reçu de l'interlocuteur et placer le résultat dans une variable dont on prendra soin de dimensionner la taille convenablement.
8. Sur quelle condition de `gpio_in()` allons nous commencer à exécuter la fonction proposée ci-dessus ? Proposer au moins une méthode (deux sont possibles) pour amorcer la lecture.
9. Comment vérifier qu'il n'y a pas eu corruption d'un nombre impair de bits lors de la transaction ?
10. Pourquoi un programmeur, outre l'épuisement des ressources matérielles, se poserait-il la question d'implémenter de façon logicielle un protocole de communication ?

Questions de cours (10 points)

1. Quelle structure de données décrit une interface de communication compatible avec le protocole de communication sur internet (IP) ?
2. Comment afficher le contenu du répertoire `include` à la racine de l'arborescence dans un shell unix ?
3. Comment remonter d'un cran dans l'arborescence de répertoires dans un shell unix ?
4. Exprimer (160|0x20) en hexadécimal et en décimal.
5. Exprimer (160+0x20) en hexadécimal et en décimal.
6. Comment passer à 0 les 2^e et 4^e bits d'un registre sur 8 bits ?
7. Un convertisseur analogique-numérique échantillonne au rythme de 100 kéchantillons par seconde. Pendant quelle durée faut-il acquérir un signal pour que sa transformée de Fourier, sur l'ensemble des données acquises, présente une résolution de 100 Hz ? Justifier.
8. Quelle est la durée de transaction du message "0123456789" au format 8E2 sur une liaison RS232 en 2400 bauds ? Justifier.
9. Un convertisseur analogique-numérique de 10 bits de résolution est référencé sur une tension de 2,5 V. Comment afficher la tension en millivolts, sachant que nous avons le mot résultant de la conversion `x` codé sur 16 bits, sans effectuer de calcul sur des nombres à virgule flottante ?
10. Qu'affiche la procédure

```
volatile unsigned char c=42;
char *cc,tableau[40];
int i;
for (i=0;i<5;i++) tableau[i]=i*20;
i=25;
cc=tableau;
for (i=0;i<5;i++) c=tableau[i]*20;printf("%d\n",c);
```

? Justifier.

[1] nous rappelons que la parité E (*Even*) – paire – implique d'avoir un nombre pair de bits à 1 dans le mot transmis

1. L'horloge est partagée entre les interlocuteurs dans un protocole synchrone, elle est implicite (convention) dans un protocole asynchrone. Dans le premier cas, le maî tre impose la cadence et l'état du bus de données change sur une transition du signal d'horloge, il n'y a pas de notion de sycnchronisation ou de base de temps.
2. un timer est conçu pour cadencer les évènements.
3. masque sur le premier bit (0x01) : `char c; if (c&0x01) printf("1"); else printf("0");`
4. Balayer les bits par décallage : `int k;char c;for (k=0;k<8;k++) if ((c>>k)&0x01) printf("1"); else printf("0");`
5.
```
void gpio_up()   {printf("1");}
void gpio_down() {printf("0");}
int main() {
char parite=0;
int k;char v,c;
gpio_down();                        // start bit
for (k=0;k<8;k++)
 {v=((c>>k)&0x01);
  parite+=v;
  if (v) gpio_up(); else gpio_down();
 }
if (v%2) gpio_up(); else gpio_down(); // parity bit
gpio_up();                          // stop_bit
}
```
6. La fonction est placée dans un gestionnaire d'interruption déclenchée à intervalle de temps régulier par un timer configuré pour se déclencher avec l'intervalle inverse du baudrate. Il faut mémoriser l'état des variables entre deux appels du gestionnaire d'interruption donc `k` devient l'indice d'une machine à état qui nous informe de l'avancement du traitement.

```
#include <stdio.h>
#include <signal.h>
void gpio_up()   {printf("1");fflush(stdout);}
void gpio_down() {printf("0");fflush(stdout);}
volatile char k=0,c,parite,transmet=0;
void isr()
{char v;
 if (transmet==1)
   {switch (k)
    {case 0:gpio_down();parite=0;break;
     case 1:
     case 2:
     case 3:
     case 4:
     case 5:
     case 6:
     case 7:    //  vvv -1 car k=0 pour le start bit
     case 8:v=((c>>(k-1))&0x01);parite+=v;if (v) gpio_up(); else gpio_down();break;
     case 9:if (v%2) gpio_up(); else gpio_down();break;
     case 10:gpio_up();transmet=0;break;
    }
    k++;
   }
   alarm(1);    // c'est reparti pour le bit suivant
}

int main()
{signal(SIGALRM,isr);alarm(1);
 c=0x55;transmet=1;
 while(1) {};   // uart est dans l'interruption
}
```

7. 8 bits de données + parité = 9 bits : il faut un short ou un char et une variable de plus

```
char c=0,p;
while (gpio_in()) {} // attend start bit
for (k=0;k<8;k++)
  {if (gpio_in()) c+=0x80;
   c=c>>1;
```

```
  }
p=gpio_in();
```

8. la fonction commence à se dérouler lorsque le start bit est transmis, donc lorsque l'entrée est nulle. Nous pouvons soit sonder continuellement l'état de la broche (mode *polling*) on déclencher une interruption matérielle (GPIO) sur le changement d'état de cette broche.
9. faire la somme des bits de `c` et lui ajouter `p` : si le résultat est impair (`%2` ou `&0x01`), il y a eu corruption des données.
10. implémentation d'un protocole qui n'est pas supporté par le matériel, par exemple un nombre de bits exotique.

Questions de cours (10 points)

1. `socket`
2. `ls /include`
3. `cd ..`
4. 160=0xA0 car le bit `0x20` est inchangé.
5. 160+0x20=160+32=192=0xC0. Cet exemple et le précédent illustrent le cas où masquer et additionner ne donnent pas le même résultat.
6. `char c;c=c&(~0x0a);`
7. la transformée de Fourier discrète est bijective : une transformée de Fourier sur N points dans le domaine temporel donne N points dans le domaine spectral. L'axe des abscisses va de $-f_e/2$ à $+f_e/2$ sur ces N points avec f_e la fréquence d'échantillonnage, ou une résolution spectrale $\Delta f = f_e/N$. Ainsi $N = f_e/\Delta f$ donc il faut 1000 points ou, au rythme de 100 kpoints/s, 10 ms de mesures. Une autre façon de trouver ce résultat est que la résolution spectale est l'inverse du temps de mesure d'après la relation d'incertitude de Heisenberg.
8. 10 caractères au rythme de 12 symboles/seconde (8 bits de données, un bit de parité, un bit de départ et deux bits de fin) donc 12/2400=1/200 seconde par caractère et 1/20ème de seconde pour les 10 caractères.
9. $V = x/1024 \times 2500 = x/1024 \times (10000/4) = (10000 \times x)/(4096)$ et $(625 \times x)/(256)$ et `x` sur 10 bits signifie que $625 \times x$ est sur 20 bits et nous devons passer par une variable intermédiaire sur 32 bits.
10. 64
    L'affichage est en dehors de la boucle for, donc `c` ne contient que la dernière valeur c'est à dire $(20 \times 4) \times 20$ avec un dépassement de capacité sur le char donc `1600%256=64`.