Examen L3 EEA sujet

É. Carry, J.-M Friedt, 20 mai 2015

Pour chaque question en rouge, vous ferez valider votre réponse par une démonstration rapide à un enseignant, et transmettrez les codes sources associés par courrier électronique à `emile.carry@femto-st.fr` et `jmfriedt@femto-st.fr`. Vous rédigerez pour toutes les questions un texte donnant les réponses et les calculs sur une feuille.
Tous les documents papier et électroniques autorisés, accès aux téléphones portables proscrit.
Les questions en rouge valent 4 points distribués entre l'analyse du code source, la compilation et l'exécution du programme sur microcontrôleur, et la pertinence du résultat. Les autres questions valent 1 point chacune.

Une carte SD est connectée sur bus SPI à un microcontrôleur Atmega32U4. On s'aidera de la documentation disponible à `http://www.atmel.com/images/doc7766.pdf` pour répondre aux questions.

1. Combien de signaux caractérisent une liaison SPI (*i.e.* combient faut-il souder de fils entre le microcontrôleur et la carte SD, à l'exclusion de la masse et de l'alimentation) ?
2. Lors de la configuration du bus SPI, le débit de données se configure au travers d'un horloge. Quel est le débit maximal accessible ?
3. Quel(s) bit(s) définir dans quel registre pour atteindre ce résultat ?
4. Nous désirons stocker un volume de 1 MB (soit 2^{20} octets) de données sur la carte SD. Quelle est la durée de la transaction, en négligeant les transactions définissant l'emplacement du stockage sur la carte SD ?
5. Si cette transaction se faisait par une liaison asynchrone à 115200 bauds, quelle serait la durée de la transaction ?
6. Les transactions avec la carte SD se font par paquets de 512 octets. Combien de tels paquets pouvons nous stocker dans la RAM d'un Atmega32U4 ?

Une fois les données stockées sur carte SD, il est souhaitable de pouvoir les restituer à l'utilisateur, et ce en affichant leur valeur sur un terminal.

7. Proposer un programme qui décompose le contenu d'une variable codée sur 16 bits en quartets en vue d'afficher le contenu de cette variable.
8. Remplir un tableau avec le code ASCII correspondant à chacun des quartets de la question précédente. Combien ce tableau comporte-t-il d'éléments ?
9. Proposer un programme qui affiche une valeur sur le port série du microcontrôleur, accessible au travers de `/dev/ttyACM0` sur le PC. On affichera le nombre (contenu dans une variable de type `short`) 0x342 en hexadécimal.
10. Même question mais en décimal. Le résultat est-il cohérent avec vos attentes ? (*i.e.* que vaut 0x342 en décimal ?).
11. afficher le nombre d'octets contenus dans une variable de type `long`.

Solutions

1. MISO, MOSI, SS#, CK donc 4 signaux (page 179 de la documentation).
2. Le chapitre 17 de la datasheet décrit le bus SPI : la table 17-4 donne la relation entre SCK et la fréquence de l'oscillateur qui cadence le microcontrôleur. La vitesse la plus rapide accessible est $F_{osc}/2$ soit 8 Mb/s.
3. SPR0=SPR1=0 (bits 0 et 1) dans SPCR et SPR2X=1 (bit 0) dans SPSR.
4. 2^{20} octets donc $2^{20} \times 8$ bits à transmettre au rythme de 8 Mb/s, donc $2^{20} \times 8/(8 \cdot 10^6) = 1{,}0486$ s
5. 115200 bauds permettent de transmettre 11520 octets/seconde dans un protocole tel que 8N1 (1 start bit, 8 data bits, 1 stop bit, donc 10 bits/octet transmis). Dans ces conditions, transmettre 1 MB nécessite environ 91 secondes.
6. L'Atmega32U4 propose 2,5 KB (0xAFF-0x100=2560 octets) de RAM (Table 5-1 de la datasheet). Nous pouvons y placer au mieux 5 tableaux de 512 octets.
7. `int k; short v; char c[4];`
   `for (k=0;k<4;k++) c[k]=(v&(0xf<<(k*4)))>>(k*4); // ou c[k]=((v>>(4*k)) & 0x0f);`
   dont on valide le bon fonctionnement sur PC par `for (k=0;k<4;k++) printf("%x ",c[k]);`.
8. `int k; short v; char c[4];`
   `for (k=0;k<4;k++) {c[k]=(v&(0xf<<(k*4)))>>(k*4);if (c[k]<10) c[k]=c[k]+'0'; else c[k]=c[k]+'A'-10;}`
   dont on valide le bon fonctionnement sur PC par `for (k=0;k<4;k++) printf("%c ",c[k]);`.
   Une valeur sur 16 bits contient 4 quartets : le tableau (ici `c`) contenant les éléments à afficher doit fournir 4 emplacements.
9. ```
   #include <avr/io.h>       //E/S ex PORTB
   #define F_CPU 16000000UL  //T=62.5ns
   #include <util/delay.h>   // _delay_ms
   #include "libttycom.h"    // init_olimexIno()
   #include "USBAPI.h"       // USB_writestr()

   int main(void){
     int k; short v=0x342; char c[7]="....\r\n0";
     for (k=0;k<4;k++)
         {c[k]=(v&(0xf<<(k*4)))>>(k*4);if (c[k]<10) c[k]=c[k]+'0'; else c[k]=c[k]+'A'-10;}
     init_olimexIno();
     sei();
     USB_init();  //Attend la connexion avec le PC(minicom)
     while(1) {USB_writestr(c);_delay_ms(1000);}
   }
   ```
10. Un nombre sur 16 bits est compris entre 0 et 65535 (ou -32767 et 32767). Dans tous les cas l'expression décimale contient au plus 5 chiffres (dizaine de milliers, milliers, centaines, dizaines et unité). Dans l'exemple précédent, l'initialisation du contenu du tableau de caractères `c` devient

    ```
    int k;
    short l=10000;
    short v=0x342;
    char c[5];
    for (k=0;k<5;k++)
       {c[k]=(v/l);v=v-(v/l)*l;l=l/10;}
    ```

    dont on vérifie le bon fonctionnement sur PC par `for (k=0;k<5;k++) printf("%d ",c[k]);` : l'affichage est `0 0 8 3 4`. 0x342=834 en décimal : le résultat est correct.
11. reprendre le programme précédent avec `v=sizeof(long);`
    ```