Examen M1

J.-M Friedt, 5 avril 2018

Connexion internet interdite, téléphones portables interdits, communications interdites, réflexion autorisée.
Supports de cours à `172.20.133.70`

1 Assignation de broches (1 point/question)

Chaque broche peut avoir une multitude de fonctions sur un microcontrôleur. L'environnement matériel de l'une ou l'autre broche permettant une même fonction peut imposer un choix. À titre d'exemple, nous considérons la transmission asynchrone sur STM32F407 tel que cablé sur la carte STM32F407-Discovery, dont la documentation est fournie dans l'archive sous forme du fichier `en.DM00039084.pdf`

1. **Identifier les deux broches permettant de communiquer sur le port de communication asynchrone numéro 1 (en transmission depuis le microcontrôleur vers le PC)**
2. **Laquelle de ces broches peut poser problème telle que cablée sur la carte ?**
3. **Quel registre de quel port permet de tout de même communiquer sur le port 1 en choisissant la broche qui n'est pas impactée par le routage de la carte ?**

2 Programmation bas niveau sur STM32 (2 points/question)

Nous fournissons une archive d'un programme fonctionnel sur STM32 qui affiche un message. En partant de l'archive `baremetal_stm32`, compilez le programme et exécutez le sous `qemu` par
`qemu-system-arm -M stm32-p103 -serial stdio -serial stdio -kernel main.bin`
avec `qemu-system-arm` disponible dans `/home/jmfriedt/qemu_stm32/arm-softmmu` [1].

4. **Comment sélectionner les 4 bits de poids faible d'une variable ?**
5. **Écrire une fonction qui affiche sur le port de communication, en s'appuyant sur la fonction d'affichage d'une chaîne de caractères, le contenu de la variable `rayon` en hexadécimal. Il peut être judicieux de développer ce code source dans un fichier séparé afin de le réutiliser dans la section suivante. Afficher le rayon de la Terre en hexadécimal depuis le STM32 émulé par `qemu` ?**
6. **Connaissant le rayon de la Terre, calculer sa circonférence en kilomètres, en suivant les consignes de programmation vues en cours, avec deux décimales exactes. Afficher ce résultat en décimal au moyen de `qemu`. Il est envisageable qu'il faille modifier la fonction d'affichage pour arriver à ce but.**

3 Programmation sur STM32 sous FreeRTOS (2 points/question)

Nous nous intéressons maintenant à séquencer `N` itérations d'une même fonction pour afficher le résultat d'un calcul sous FreeRTOS. On s'inspirera pour cela de l'archive `FreeRTOS`. On commencera par démontrer la compilation et l'exécution du programme d'exemple dans `qemu`.

7. **Lancer $N = 5$ tâches identiques et observer leur comportement. Que constatez vous comme dysfonctionnement ?**
8. **Corriger ce dysfonctionnement en exploitant le mécanisme approprié.**
9. **Passer *trois arguments*, au lancement de chaque tâche, correspondant à 1000, 1333 et 2000 fois l'indice de la tâche, et compléter l'affichage de chaque tâche par ces valeurs.**

4 Questions de cours (1 point/question)

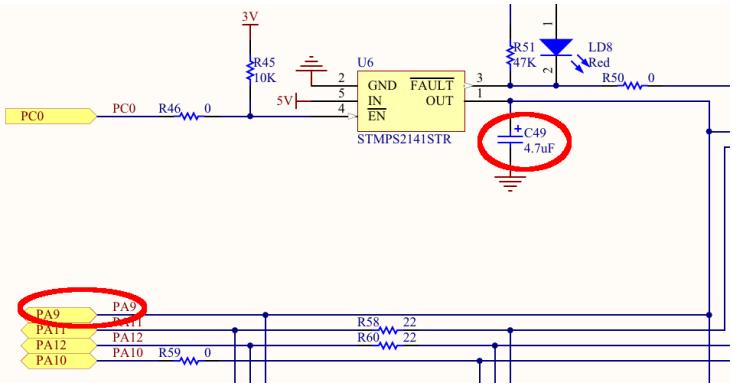
10. **Quel mécanisme intrinsèque à la majorité des processeurs supportant un système d'exploitation interfère avec l'accès aux ressources matérielles ?**
11. **En quel emplacement de la RAM se trouve classiquement la pile ?**
12. **Quel est la durée de vie d'une variable placée sur la pile ?**
13. **En quel emplacement de la RAM se trouve classiquement le tas ?**
14. **Quel est la durée de vie d'une variable placée sur le tas ?**
15. **Que vaut 0x234×4 ? (dans la base de votre choix). Justifier comment ce résultat est obtenu.**

1. Noter que `qemu` prend en argument le fichier `bin` issu de la conversion de l'exécutable au format ELF par `objcopy`

Corrections

1. Toute les informations sont dans la *datasheet* `dm00037051` spécifique aux implémentations des microcontrôleurs, ou le manuel de la carte STM32Discovery-F4 `en.DM00039084.pdf`, mais le *Reference Manual* est inutile. Nous trouvons, en recherchant USART1, pp.20/21 de la datasheet ou p.24 du descriptif de la carte d'évaluation, que les broches PA9 et PA10 sont associées à USART1_TX et USART1_RX respectivement. Alternativement, PB6 et PB7 peuvent aussi s'acquitter de cette fonction, mais ce n'est pas leur attribution principale (I2C).

2. Nous constatons sur le schéma de la carte `en.DM00039084.pdf` que PA9 sert de détection de niveau du port USB utilisé en mode OTG : de ce fait, un condensateur de 4,7μF (C49) y est connecté, lissant le signal de communication qui ne peut donc plus porter d'information vers le PC.



3. Nous passons de PA9/PA10 à PB6/PB7 en sélectionnant la fonction alternative (AF) appropriée à chaque broche. Dans la datasheet `dm00037051.pdf`, p.62, la fonction alternative AF7 de PA9 et PA10 devient la fonction alternative AF7 de PB6 et PB7 (p.63).

4. `variable & 0x0f`

5. la sortie du programme lors de son exécution est

```
Hello World
Hello World
...
```

   6400 en décimal s'écrit 0x1900, tel que nous le constatons avec

```
#include "common.h"
void affiche(short s)
{int k;char c;
 for (k=3;k>=0;k--)
   {c=(s>>k*4)&0x0f;
    if (c<10) uart_putc('0'+c); else uart_putc('A'+c-10);
   }
}

int main(void){
 short rayon=6400;
 Usart1_Init(); // inits clock as well
 while(1){affiche(rayon);uart_putc('\n');}
 return 0;
}
```

6. Le calcul d'incertiture est le suivant : sachant que nous désirons obtenir la circonférence C avec deux décimales exactes, l'incertitude $d\pi$ sur π est de

$$C = 2\pi R \Rightarrow dC = 2Rd\pi \Leftrightarrow d\pi = \frac{dC}{2R} = \frac{10^{-2}}{2 \times 6400} = 8 \times 10^{-7} \simeq 10^{-6}$$

   donc il faut 6 décimales de π. Les deux fractions rationnelles classiques – que l'on retrouve par la fonction `rats()` de GNU/Octave – de π sont $\pi \simeq 355/113$ et $\pi \simeq 22/7$. Dans les deux cas, la multiplication du numérateur par 6400 fait dépasser la capacité de stockage d'un `short` : il faut donc passer par une variable intermédiaire en long. Cependant, 22/7 ne présente pas suffisamment de décimales exactes et le résultat est erroné. Seule la fraciton 355/113 permet d'atteindre le résultat. Mis à part cette subtilité, l'affichage se réduit à

```
#include "common.h"

void affiche(long val)
{int k=10000000;
 while (k>10) {uart_putc('0'+val/k);val-=(val/k)*k;k/=10;}
 uart_putc('.'); // decimales
 while (k>=1) {uart_putc('0'+val/k);val-=(val/k)*k;k/=10;}

```

```
}

int main(void){
 short rayon=6400;
 long circonference;
 Usart1_Init(); // inits clock as well
 circonference=((long)rayon*35500*2)/113;
 while(1){affiche(circonference);uart_putc('\n');}
 return 0;
}
```

La solution de multiplier par 3141592 puis de diviser par 1000000 n'est pas viable car ce numérateur multiplié par 6400 dépasse 2^{32}.

7. la sortie du programme lors de son exécution est

```
BonjourBonjBonjour l la valeuour la vala valeur r de la meur de lade la mesesure est mesure eure est  st
BonjourBonjour lBonjour l la valeua valeur da valeur r de la me la mesude la meesure estre est su re est
Bonjour laBonjour lBonjour l valeur da valeur a valeur e la mesde la mesude la mesure est rure est e est
```

8. Protéger l'affichage par un mutex
9. Définir une structure donc les éléments sont les arguments passés en paramètre.

   `struct troisvaleurs {int i1;int i2;int i3;};`

   Penser à définir cette structure en `static` dans le main :

   `static struct troisvaleurs v[NBTASK];`

   et finalement caster le passage d'argument de `void*` en `struct*`

   ```
   struct troisvaleurs *argument;
   argument=(struct troisvaleurs*)dummy;
   ```

```
#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"
#include "common.h"

#define NBTASK 3
struct troisvaleurs {int i1;int i2;int i3;};
xSemaphoreHandle xMutex;

void vPrintUart(void* dummy)
{struct troisvaleurs *argument;
 argument=(struct troisvaleurs*)dummy;
 while(1){
          xSemaphoreTake( xMutex, portMAX_DELAY );
          uart_puts("Bonjour␣la␣valeur␣de␣la␣mesure␣est␣\0");
          affiche(argument->i1,c); uart_puts(c); uart_putc('␣');
          affiche(argument->i2,c); uart_puts(c); uart_putc('␣');
          affiche(argument->i3,c); uart_puts(c); uart_putc('␣');
          xSemaphoreGive( xMutex );
          vTaskDelay(100);
   }
}

int main(void){
 static struct troisvaleurs v[NBTASK];
 int k;
 Usart1_Init(); // inits clock as well
 xMutex=xSemaphoreCreateMutex();
 for (k=0;k<NBTASK;k++)
    {v[k].i1=1000*k;
     v[k].i2=1333*k;
     v[k].i3=2000*k;
     xTaskCreate( vPrintUart, (signed char*) "Uart", 128,(void*)&v[k],3,NULL );
    }
 vTaskStartScheduler();
 return 0;
}
```

10. Memory Management Unit – MMU

11. à l'adresse la plus élevée : le pointeur de pile est *décrémenté* quand on empile une variable
12. locale à la durée d'exécution de la fonction
13. à l'adresse la plus basse de la RAM
14. la durée d'exécution du programme
15. 0b001000110100 << 2 = 0b100011010000 =0x8D0