Communication RS232 et mesure de température.

É. Carry, J.-M Friedt, 15 mars 2018

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

Notre objectif est de mesurer la température au moyen de la sonde interne au microcontrôleur Atmega32U4, et de transmettre cette mesure pour un affichage à l'écran, idéalement au format de degrés celsius avec une décimale de résolution.

Nous fournissons les fonctions d'initialisation du port de communication `void uart_init()`, ainsi que les fonctions d'initialisation `void setupADC()` et d'acquisition `int temperature(void)` de la sonde de température.

1. comment sélectionner les 4 bits de poids faibles d'une variable `m` ?
2. sachant que nous avons en plus à notre disposition dans `/home/jmfriedt/exam_EP2_2018.tar.gz` une procédure `void rs232_write(char*,int);` qui accepte une chaîne de caractères et le nombre de caractères à afficher (prototype habituel dans la fonction `write();`), afficher à l'écran "Hello World" grâce à un programme séparé de celui contenant les fonctions fournies pour ce projet. Démontrer la capacité à compiler les deux codes sources, bibliothèque fournie et votre programme, pour générer un exécutable fonctionnel capable d'afficher un message à l'écran par `minicom -D /dev/ttyUSB0` (le port de communication est `/dev/ttyUSB0`) .
3. Le signal transmis par le microcontrôleur vers le PC est nommé TXD. Ce signal est reproduit sur une des broches des deux borniers de part et d'autre de la carte Olimexino32U4. Identifier cette broche, observer son potentiel à l'oscilloscope, et en déduire la durée de chaque bit. En déduire un débit de communication probable, compte tenu des débits classiquement accessibles sur un périphérique de communication asynchrone (ces débits classiques se trouvent dans la documentation technique du microcontrôleur).
4. Étant capables d'afficher un message à l'écran, afficher en hexadécimal le contenu d'une variable définie sur 8 bits (ni plus, ni moins) qui s'incrémente chaque seconde. L'affichage sera raffraîchi chaque fois que la variable change de valeur.
5. Basé sur le programme précédent, afficher en hexadécimal le contenu d'une variable définie sur 16 bits (ni plus, ni moins) qui s'incrémente chaque seconde. L'affichage sera raffraîchi chaque fois que la variable change de valeur.
6. stocker dans la variable sus-créée le résultat de la mesure de température obtenue par l'appel à la fonction `int temperature(void);`. Vérifier qu'en échauffant la surface du microcontrôleur, la mesure de température évolue bien.
7. La sensibilité en température, si la "bonne" référence de tension est sélectionnée (quelle est-elle, d'après la documentation technique ?), est de 1 bit/K, tel que décrit dans la note d'application AVR122 "Calibration of the AVR's Internal Temperature Reference" dont un extrait est fourni ci-dessous

   > This application note describes how to calibrate and compensate the temperature measurements from the Atmel® ATtiny25/45/85. It can also be used on other AVR® microcontrollers with internal temperature sensors. The temperature measurement is based on an on-chip temperature sensor that is coupled to a single ended ADC channel. The sensor is a diode that produces a temperature dependent voltage. This voltage is measured with the ADC. The voltage has a linear relationship to temperature and the result has approximately a 1 LSB/°C correlation to temperature.
   >
   > The diode voltage is highly linear, but due to process variations the temperature sensor output voltage varies from one chip to another. Also, the internal voltage reference used as the ADC reference voltage varies with temperature. The typical accuracy of temperature measurements over the

   Afficher la température en degrés celsius. Cette valeur semble-t-elle réaliste ? Si non, justifier.
8. Avec une sensibilité de 1 bit/K, nous ne pouvons prétendre à une résolution meilleure que le Kelvin. Comment améliorer cette résolution, pour par exemple obtenir une résolution de 0,1 K ? Démontrer la capacité à afficher une température avec une décimale significative, et ce sans utiliser de calcul sur des nombres à virgule flottante (i.e. en se restreignant à calculer sur des entiers).
9. Sachant que la mesure de température par le convertisseur analogique numérique est idéalement un mot qui représente la température en Kelvin, quelle est la sensibilité (en V/K) de la diode servant de capteur de température ?
10. La documentation technique nous informe, dans la section des caractéristiques du convertisseur analogique-numérique (Tableau 29.5), que la référence de tension n'est pas aussi idéale que le laisserait penser son nom. Quelle est la gamme de valeurs que peut prendre la référence de tension ? Quelle est l'erreur résultante sur une mesure supposée à 100°C.

Solutions

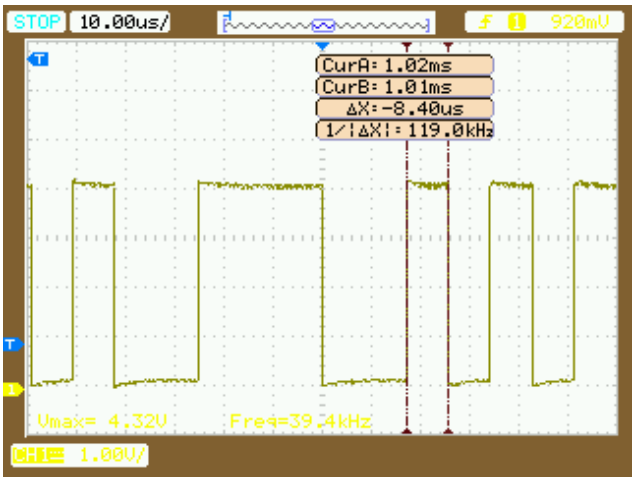
1. le programme ci-dessous affiche “Hello World”, affiche en hexadécimal un nombre sur 16 bits (`affiche_s()`) en exploitant l'affichage d'un entier sur 8 bits (`affiche_b()`).

```
#include "temperature.h"

void affiche_b(char v,char *c)
{unsigned char tmp;
 tmp=(v>>4);
 if (tmp<10) c[0]=(tmp+'0'); else c[0]=(tmp+'A'-10);
 tmp=(v&0x0f);
 if (tmp<10) c[1]=(tmp+'0'); else c[1]=(tmp+'A'-10);
}

void affiche_s(short s,char* c)
{affiche_b(s>>8,c);
 affiche_b(s&0x0ff,&c[2]);
}

void affiche_d(short s,char* c)
{c[0]=s/100;s=s-c[0]*100;c[0]=c[0]+'0';
 c[1]=s/10;s=s-c[1]*10;  c[1]=c[1]+'0';
 c[2]=s;                 c[2]=c[2]+'0';
}

int main()
{unsigned short s;
 char c[15];
 int n;
 setupADC();
 uart_init();

 rs232_write("Hello␣World\r\n",13);
 c[4]='␣';
 c[8]='␣';
 c[12]='\r';
 c[13]='\n';
 while (1)
 {s=temperature();
  affiche_s(s,c);
  affiche_d(s-273,&c[5]);
  s=0;
  for (n=0;n<100;n++) s+=(temperature()-273);
  affiche_d(s/10,&c[9]);
  rs232_write(c,15);
  _delay_ms(100);
 }
}
```

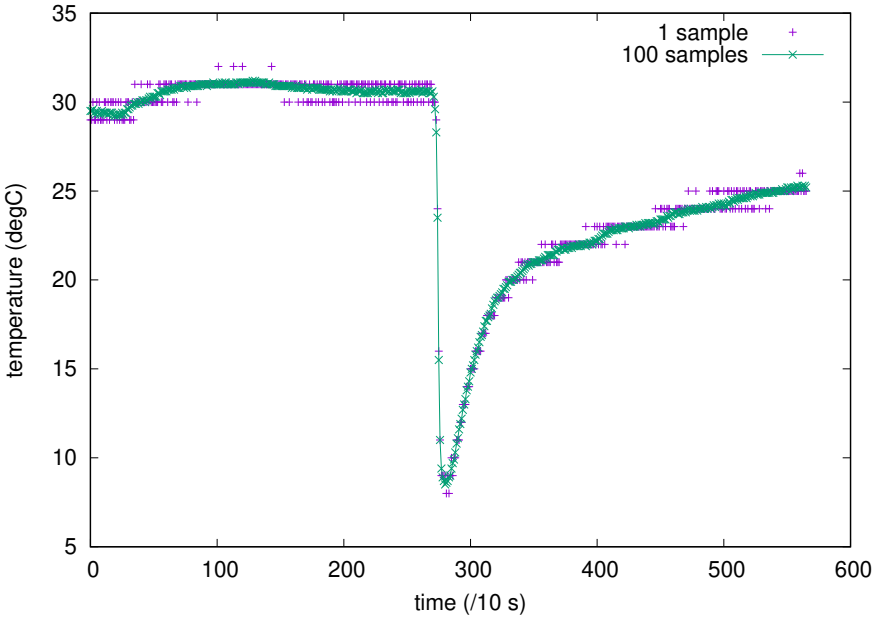
   Le Makefile correspondant, qui se charge de compiler séquentiellement les objets puis de les lier en un exécutable, est (noter l'utilisation de la variable `$<` qui définit l'argument de la condition de compilation)

```
all: affiche_temperature

CFLAGS=-mmcu=atmega32u4 -Os -W -Wall

affiche_temperature: temperature.o main.o
   avr-gcc -mmcu=atmega32u4 -o affiche_temperature temperature.o main.o

main.o: main.c
   avr-gcc $(CFLAGS) -c $<

temperature.o: temperature.c
   avr-gcc $(CFLAGS) -c $<
```

2. Le signal observable sur la broche D1 présente une transition la plus courte d'environ 8,4 μs, ou la période d'un signal proche de 115200 bauds,



3. la lecture de la température se traduit par la courbe violette ci-dessous lorsque nous refroidissons le microcontrôleur avec une bombe givrante



4. la fonction `affiche_d()` du listing ci-dessus affiche la température en décimal. Nous retranchons 273, l'écart entre les échelles de degrés Celsius et Kelvin. L'écart avec la température ressentie peut provenir de l'incertitude sur la sonde de température de $\pm 10°C$.
5. La résolution s'améliore en moyennant des mesures successives. 100 mesures sont nécessaires pour abaisser la variance d'un facteur 10, donc gagner une décimale sur la mesure de température.
6. l'affichage de la température moyennée 100 fois et divisée par 10 donne une mesure au dixième de degrés près.
7. 2,56V/1023=1 K donc le coefficient de sensibilité est de 2,5 mV/K
8. $T = \text{bit}/(2^{10}-1) \times V_{ref}$. 100°C=373K $\Rightarrow$ bit=373. Si V_{ref} est de 2,4 à 2,8V, alors $T = 373/2,56 \times \{2,4..2,8\} = 77..135°C$.