François SCHNEIDER – Lycée Victor-Hugo BESANÇON.

STS SE

Développement de microcontrôleurs Microchip avec PICC
validation fonctionnelle PROTEUS

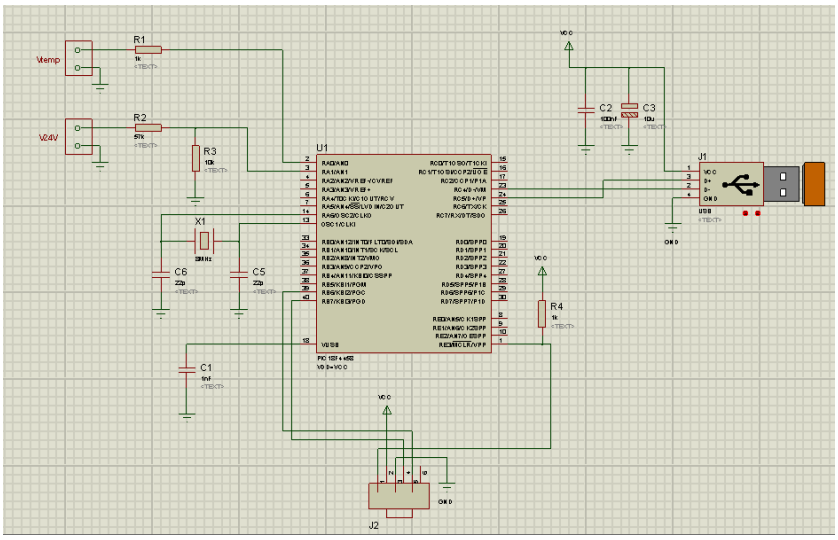
Entrées analogiques, USB

Utilisation Wizard et PROTEUS Simulation Validation

Prérequis : langage C, PROTEUS ISIS simulation d'un microprocesseur.

Le projet.

Le schéma du projet :



Nous trouvons :

- Le montage alimenté par l'interface USB.
- Une interface USB pour la liaison PC J1.
- Une interface ICD pour la programmation.
- Un quartz de 8MHz pour produire l'horloge de base du processeur. Le processeur sera configuré pour obtenir une fréquence de 48MHz pour le processeur et l'interface USB.
- Deux entrées analogiques :
    - Une en provenance d'une sonde de température, qui varie de 0 à 5V pour une température variant de 20 à 70°C avec une précision de 0,1°C.
    - L'autre en provenance d'une alimentation de 24V à contrôler. L'acquisition du 24V doit se faire avec une précision de 1/1000 sur la mesure nominale soit 24mV par rapport au 24V nominal.

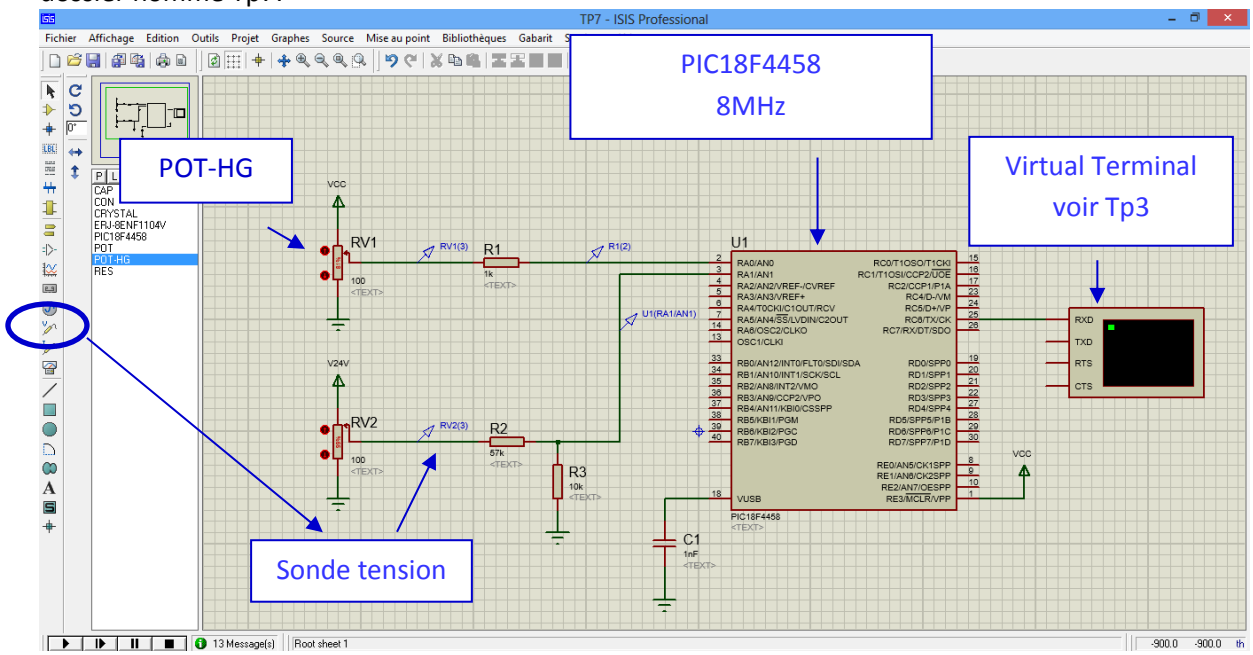
Le microcontrôleur 18F4458 possède 13 canaux analogiques de 12 bits. Il est assez simple de montrer qu'un convertisseur 12 bits est suffisant pour répondre au cahier des charges.

Le TP se déroule en plusieurs phases :

I. PICC : le wizard, production du squelette du programme
- II. Mise en œuvre de l'interface série : reprise des connaissances acquises au TP3.
- III. Acquisition d'une grandeur analogique et transmission vers l'interface série.
- IV. Acquisition des deux entrées analogiques.
- V. Test réel : transmission vers USB et transmission vers l'interface série.

Le schéma page suivante nommée Tp7.DSN est utilisé pour les parties II. à IV. La maquette correspondant au schéma du projet est utilisée pour la partie V.

Le schéma de simulation du montage est donné ci-dessous avec le nom des composants en bleu (librairie ISIS). Vous dessinez le schéma, vous le sauvegardez avec pour nom « Tp7.dsn » dans un dossier nommé Tp7.



Au niveau de la simulation les informations acquises sont envoyées à travers l'interface série vers virtual com. Sur la maquette elles sont envoyées à travers l'interface USB, en mode Virtual Com, vers un PC muni de Teraterm.

I. PICC : le wizard, production du squelette du programme.

Nous allons utiliser le logiciel PICC pour produire le programme. La programmation se fait en langage C. PICC nous permet, pour les microcontrôleurs 8 et 16 bits de marque microchip :

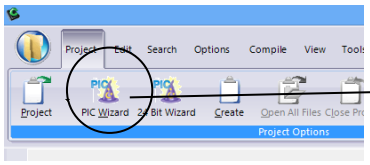
- De produire le squelette et la configuration de base du programme.
- D'éditer le programme en langage C.
- De compiler le programme source pour obtenir le programme en langage machine. Deux versions sont produites :
    - .HEX : programme binaire simple.
    - .COF : programme binaire contenant les éléments pour la simulation ou l'émulation en pas à pas.
- De programmer les microcontrôleurs.
- De tester à l'aide d'une sonde les programmes dans la cible.

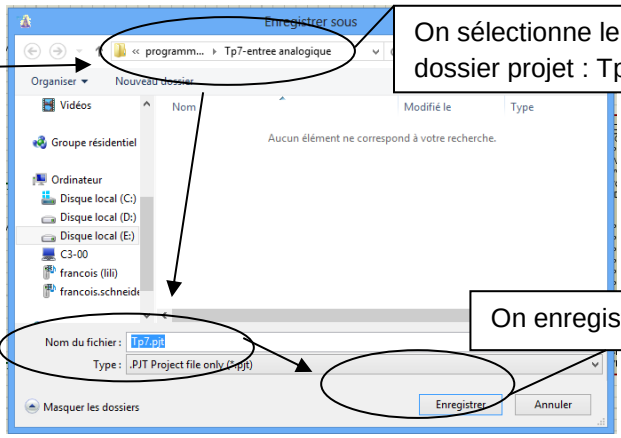
Dans ce document nous testerons les programmes par simulation avec Proteus. Vous devez avoir l'option pour la simulation des processeurs microchip, ici uniquement 8bits.

Nous allons compléter le programme en plusieurs phases. En dernière phase nous verrons comment utiliser un afficheur différent.

Création du squelette du programme en langage C avec PICC.

Vous lancez PICC. Nous allons utiliser le Wizard pour produire le squelette et la configuration.



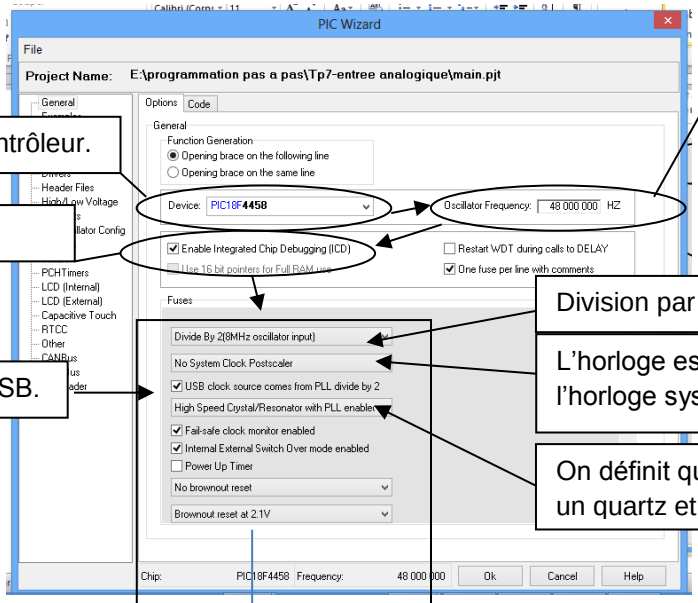


On sélectionne le dossier projet : Tp7

On donne un nom au projet : Tp7.pjt
Ici on donne le même nom au dossier et au projet, mais ce n'est pas obligatoire.

On enregistre le projet

Nous obtenons la première page du Wizard, il faut alors définir les valeurs des différentes ressources utilisées. Ici seul l'ICD et l'oscillateur sont utilisés.



Définir le microcontrôleur.

Définir la fréquence du quartz : 48MHz

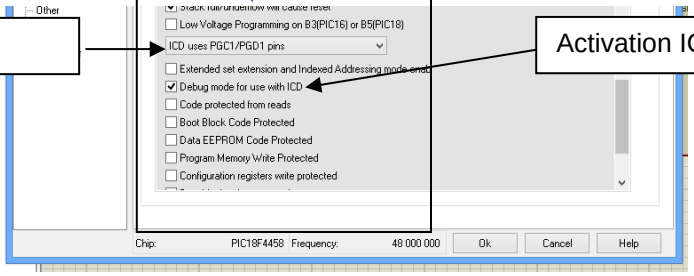
Activation de l'ICD.

Division par 2 de la fréquence du quartz.

L'horloge est produite en divisant l'horloge système par 1.

Sélection de l'horloge pour l'USB.

On définit que l'horloge est produite par un quartz et on active la PLL.

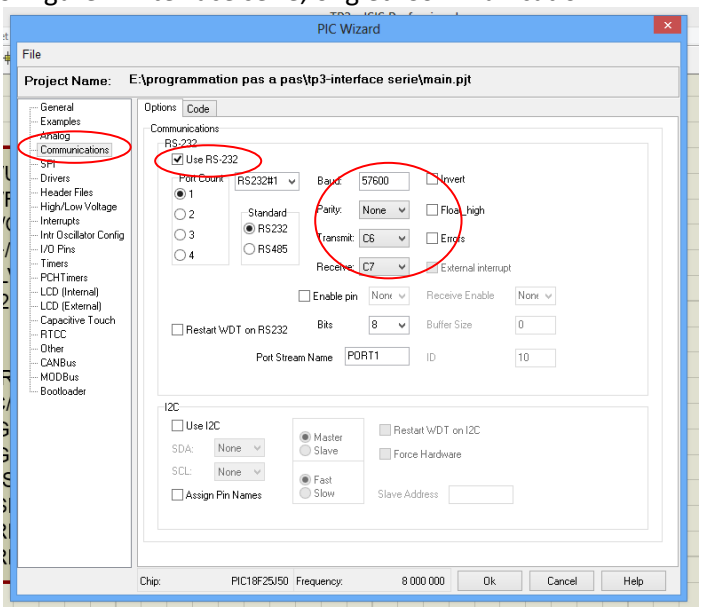


Pins PGC1/PGD1 pour ICD.

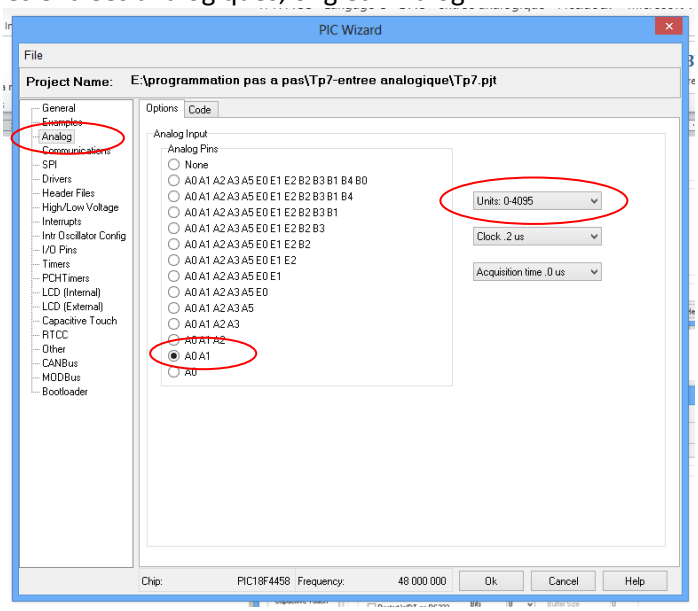
Activation ICD.

Remarque : pour avoir plus de détails sur la configuration de l'horloge, il faut se référer à la notice technique du circuit PI18F44J50. Ici la configuration est prête pour utiliser l'USB en mode haute vitesse.

Nous allons ensuite configurer l'interface série, onglet 'Communication'.



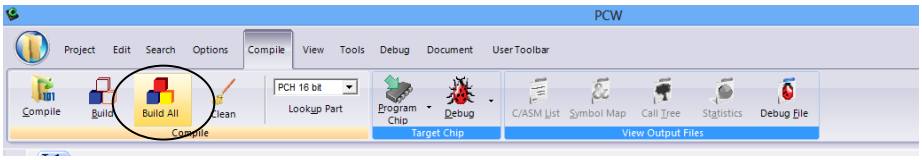
Nous terminons avec les entrées analogiques, onglet 'Analog'.



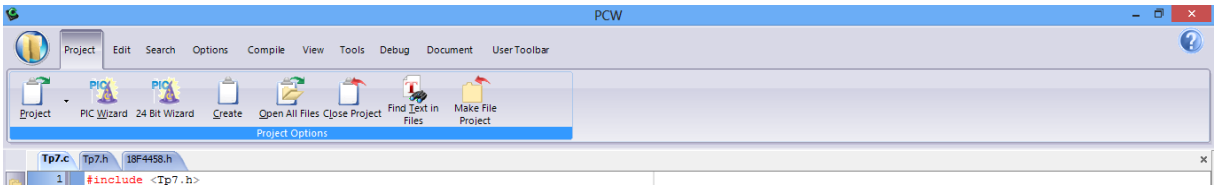
En activant le bouton Ok, le squelette du programme est réalisé : deux fichiers sont produits.

- TP7.c : fichier contenant les instructions dont la fonction principale 'void main()'.
- TP7.h : fichier contenant les déclarations.

Nous pouvons le compiler le projet en cliquant depuis l'onglet 'Compile' sur le bouton 'Build' :



Le résultat de la compilation est positif, nous pouvons poursuivre.
Il est possible de voir les différents fichiers utilisés par le projet, onglet Projet :



Trois fichiers sont utilisés par le projet :

- Tp7.c : fichier produit par le Wizard, contenant le squelette du programme.
- Tp7.h : fichier produit par le Wizard contenant les déclarations.
- 18F4458.h : fichier provenant de la librairie de PICC, contenant les déclarations particulières du composants 18F4458. En jetant un coup d'œil, nous voyons l'affectation des ports, des registres.

Les fichiers Tp7.c et Tp7.h.

Le fichier Tp7.C	Le fichier Tp7.H

#include <Tp7.h>
 	
void main()
{
  setup_adc_ports(AN0_TO_AN1);
  setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
  while(TRUE)
  {
  //TODO: User Code
  }
}
	#include <18F4458.h>
	#device ICD=TRUE
	#device adc=12
	
	#FUSES NOWDT //No Watch Dog Timer
	#FUSES WDT128 //Watch Dog Timer uses 1:128 Postscale
	#FUSES PLL2 //Divide By 2(8MHz oscillator input)
	#FUSES CPUDIV1 //No System Clock Postscaler
	#FUSES HSPLL //Crystal/Resonator with PLL enabled
	#FUSES NOBROWNOUT //No brownout reset
	#FUSES NOLVP //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
	#FUSES ICSP1 //ICD uses PGC1/PGD1 pins
	#FUSES NOXINST //Extended set extension and Indexed Addressing mode disabled (Legacy mode)
	#FUSES DEBUG //Debug mode for use with ICD
	
	#use delay(clock=48000000)
	
	#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,stream=PORT1)

La structure conditionnelle while(TRUE) { } :

En langage C, il existe différentes structures conditionnelles, qui permettent de faire des instructions sous conditions ainsi que des boucles sous conditions.

L'instruction *while* permet d'exécuter plusieurs fois la même série d'instructions.

La syntaxe de cette expression est la suivante :

```
while (condition réalisée) {
    liste d'instructions;

}
```

Cette instruction exécute la liste d'instructions tant que (while est un mot anglais qui signifie tant que) la condition réalisée est vraie. Les accolades permettent de délimiter le début et la fin de la liste d'instruction.

Pour avoir plus d'informations cliquez ici.

Nous revenons au programme « Tp7.c ». La boucle

```
while(TRUE)
{
  //TODO: User Code
}
```

Les instructions de configuration se place avant cette boucle.
La boucle s'effectue de façon perpétuelle car la condition est TRUE (vraie).
//TODO: User Code : indique ou il faut mettre la suite d'instruction du programme principal.

II. Nous allons ci-dessous valider le fonctionnement de l'interface série :

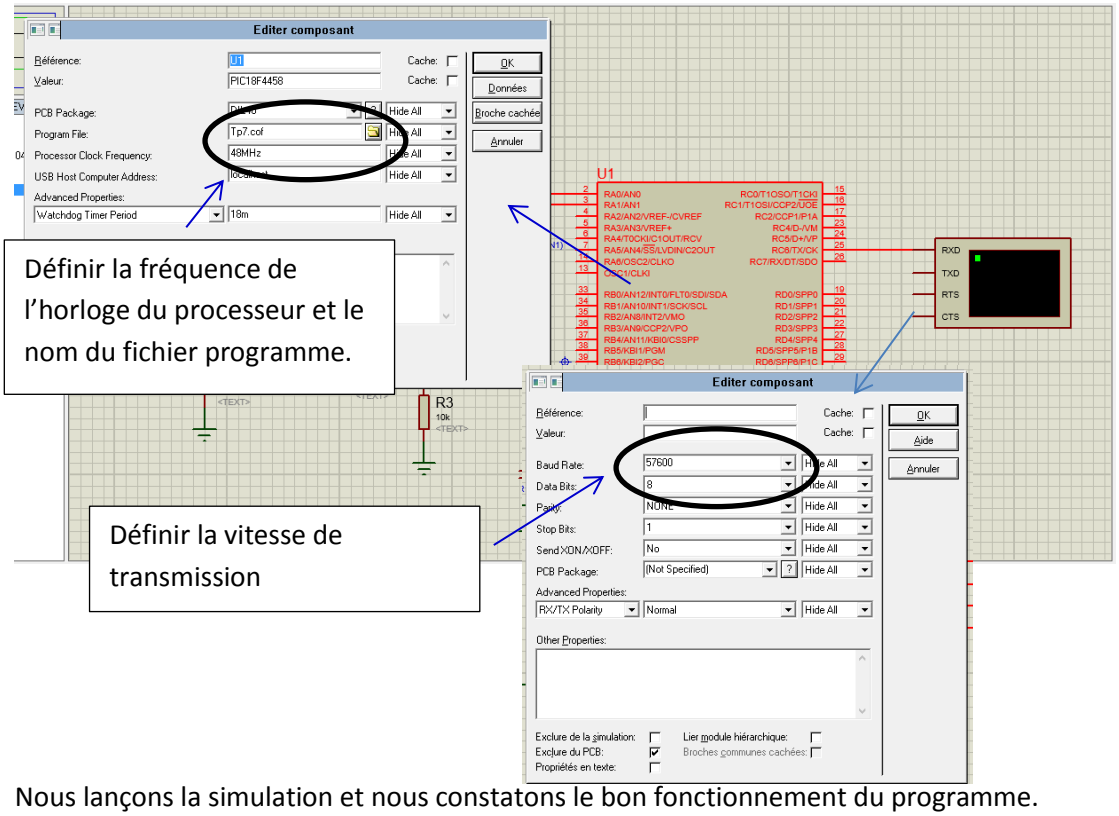
Nous allons afficher le message « Bonjour » au démarrage du système et ensuite attendre. Comment faire ?

Nous avons vu dans différents TP comment envoyer des données vers l'interface série. Pour cela la fonction main() devient :

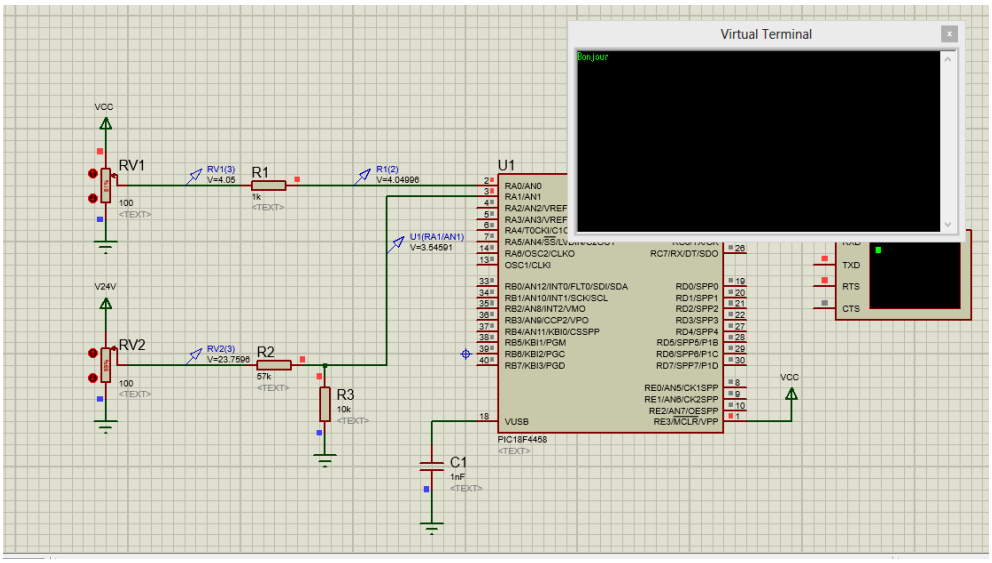
```
void main()
{
  setup_adc_ports(AN0_TO_AN1);
  setup_timer_3(T3_DISABLED | T3_DIV_BY_1);

  printf("Bonjour\r\n");
  while(TRUE)
  {
    //TODO: User Code
  }
}
```

➔ Ligne ajoutée en rouge.

Nous allons maintenant configurer Proteus. Il faut définir les paramètres du microcontroleur et du « Virtual terminal ».



Nous lançons la simulation et nous constatons le bon fonctionnement du programme.



Si nous déplaçons la position des curseurs des potentiomètres, nous constatons que les tensions mesurées varient.

Nous pouvons maintenant mettre en œuvre le convertisseur analogique.

III. Acquisition d'une grandeur analogique et transmission vers l'interface série.

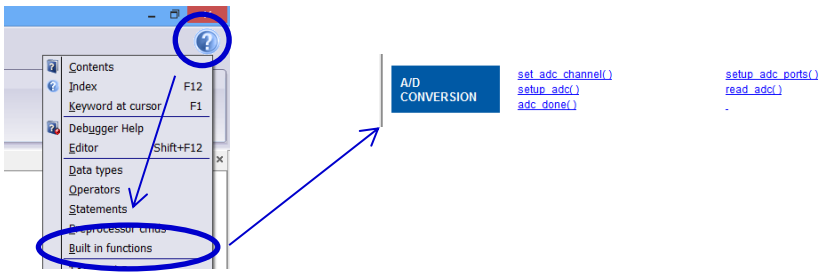
Le programme que nous souhaitons a un fonctionnement.

- Il affiche « Bonjour » au démarrage pendant 100ms.

Il répète toutes les secondes environ.

- Lecture de l'entrée analogique AN0.
- Transmission de la valeur lue vers la liaison série.

Nous trouvons les fonctions associées aux entrées analogiques dans l'aide :



Le programme Tp7.C devient :

```
#include <Tp7.h>

void main()
{
  setup_adc( ADC_CLOCK_INTERNAL );
  setup_adc_ports(AN0_TO_AN1);
  setup_timer_3(T3_DISABLED | T3_DIV_BY_1);

  printf("Bonjour\r\n");
  delay_ms(500);
  while(TRUE)
  {
    printf("AN0 %Lu\r\n",read_adc());
    delay_ms(1000);
  }

}
```

En rouge les lignes ajoutées.

La simulation nous permet de montrer le bon fonctionnement du programme.

- Affichage au démarrage 'Bonjour'.
- Affichage du message correspondant à la mesure de la tension sur l'entrée AN0.

La grandeur affichée au niveau de 'Virtual terminal' varie en faisant varier la position du curseur de RV1.

Envoi de la température en °C de la température.

Pour cela il suffit de faire un calcul à partir du tableau suivant :

Température mesurée	Tension	Nombre obtenu après conversion (12bits)	
Degré	Volt	Hexadécimal	Décimal
20°C	0V	0x000	0
70°C	+5V	0xFFF	4096

La relation liant la température au nombre N acquis est donc :

$$T = \frac{70 - 20}{4096} * N + 20$$

La relation ici est valable pour un capteur linéaire. Pour le cas où vous utilisez un capteur de type CTN, il faudra tenir compte de la non-linéarité dans la formule.

Le programme Tp7.C devient :

```
#include <Tp7.h>
void main()
{
  float temperature;
  setup_adc( ADC_CLOCK_INTERNAL );
  setup_adc_ports(AN0_TO_AN1);
  setup_timer_3(T3_DISABLED | T3_DIV_BY_1);

  printf("Bonjour\r\n");
  delay_ms(500);
  while(TRUE)
  {
    temperature = read_adc();
    temperature = temperature * 50 / 4096 + 20;
    printf("T = %f\r\n",temperature);
    delay_ms(1000);
  }
}
```

En rouge les lignes ajoutées.

La simulation montre le bon fonctionnement du programme. Nous pouvons passer à l'acquisition de deux entrées analogiques.

IV. Acquisition des deux entrées analogiques.

L'acquisition de deux entrées analogiques est simple, il suffit avant de lancer l'acquisition d'une entrée d'effectuer la fonction set_adc_channel (chan [,***neg***])). La variable indique le canal que nous voulons acquérir.

Le programme Tp7.C devient :

```
#include <Tp7.h>
void main()
{
  float temperature;
  setup_adc( ADC_CLOCK_INTERNAL );
  setup_adc_ports(AN0_TO_AN1);
  setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
  printf("Bonjour\r\n");
  delay_ms(500);
  while(TRUE)
  {
    set_adc_channel (0);
    temperature = read_adc();
    temperature = temperature * 50 / 4096 + 20;
    printf("T = %f, ",temperature);
    set_adc_channel (1);
    printf("AN1 = %lu\r\n",read_adc());
    delay_ms(1000);
  }
}
```

En rouge les lignes ajoutées.

La simulation montre le bon fonctionnement du programme. Vous pouvez ici adapter le programme pour afficher la tension à contrôler.

Nous pouvons passer à la situation réelle.

V. Test réel : transmission vers USB et transmission vers l'interface série.

Dans cette partie il s'agit simplement de remplacer la liaison RS232 par l'interface USB en mode 'Virtual Com'. Vous pouvez pour comprendre regarder le TP6, partie USB.

```
#include <Tp7.h>
#include <usb_cdc.h>

void main()
{
  float temperature;
  setup_adc(  ADC_CLOCK_INTERNAL  );
  setup_adc_ports(AN0_TO_AN1);
  setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
  usb_init_cs();
  printf("Bonjour\r\n");
  delay_ms(500);
  while(TRUE)
  {
    usb_task ();
    set_adc_channel (0);
    temperature = read_adc();
    temperature = temperature * 50 / 4096 + 20;
    printf(usb_cdc_putc,"T = %f, ",temperature);
    set_adc_channel (1);
    printf(usb_cdc_putc,"AN1 = %lu\r\n",read_adc());
    delay_ms(1000);
  }
}
```

En rouge les lignes ajoutées.

Vous reliez la carte au PC et utilisez Teraterm pour valider le fonctionnement.
Le seul problème, vous ne verrez jamais 'Bonjour' car Teraterm sera connecté trop tardivement.
Il existe une solution en ajoutant avant d'afficher la ligne printf l'instruction : while (usb_cdc_connected() == false); Le microcontrôleur attend qu'un programme se connecte. Attention cela ne marche pas avec tous les programmes.