LP2S2 TP n°1
Microcontrôleur ATMEL ATMEGA16 Mise en œuvre des ports //

Objectifs du TP :

- Se familiariser avec les outils de développement CodeVision et de simulation *AVR Studio4* du microcontrôleur ATMEL ATMEGA16 (édition de fichiers et compilation).
- Débugger et tester un programme écrit en langage C sous CodeVision.
- Voir l'évolution des ressources internes du microcontrôleur étudié et les justifier.
- Evaluer un temps d'exécution.
- Mettre en œuvre les points d'arrêt, l'exécution en pas à pas, la visualisation des variables.
- Se familiariser avec la mise en œuvre des ports parallèles (PORTA, PORTB, PORTC, PORTD).

Matériel utilisé :

- Un PC équipé des logiciels CodeVision version évaluation et l'environnement de développement AVR Studio4 + 1 cordon liaison série.
- Une interface JTAG-ICE version RS232 ou USB.
- Une carte prototype équipée d'un micro contrôleur Atmega16 et diverses ressources.
- Documentation des logiciels et du micro contrôleur.

Expérimentation n°1:

Le TP s'effectue en 2 temps :

- une phase de simulation et de vérification du fonctionnement (seul le PC est utilisé).
- Une phase d'utilisation de la maquette via la sonde JTAG.

1) Dans CodeVision créer un projet appelé « recopie » (suivre les directives données par l'enseignant).
2) Ecrire un programme en C qui lit l'état du port A et le renvoit sur le port C (on peut utiliser le programme fourni). Compiler le programme.
3) En utilisant l'outil de recherche de Windows, rechercher le fichier « mega16.h » et l'ouvrir avec le bloc notes de Windows. Justifier la raison de sa présence dans le programme. Ce fichier est-il indispensable ?
4) commenter le programme fourni.

Simulation :

1) Compiler le programme (directive « make » dans « Project ») puis ouvrir AvrStudio4 (se reporter à la documentation de mise en œuvre de CodeVision).
2) Se mettre en mode « debug » et tester le programme en simulation. (on testera les différentes commandes du simulateur). On prendra soin d'observer comment évoluent les différentes ressources du microcontroleur.
3) Quel est le rôle de ce programme ?

Mise en œuvre de la sonde d'émulation JTAG-ICE :

1) Sélectionner l'émulateur JTAG-ICE et le µContrôleur ATMEGA16 (Debug => Select platform and Device …).
2) Exécuter le programme sur l'émulateur (Start Debug => Run).
3) Vérifier le fonctionnement sur la maquette.

Expérimentation n°2:

1) Créer un projet appelé « compteur ».
2) Charger le programme « compteur.c ». Compiler.
3) Exécuter le programme dans AvrStudio en mode pas à pas puis en mode normal. Quelle est l'utilité du sous-programme « delay_ms » ? justifier la présence de l'instruction :

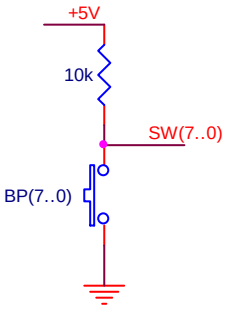
#include <delay.h>. Ouvrir avec le bloc notes la bibliothèque « delay.h ». Conclusions.

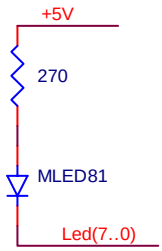
4) **Evaluer** l'ordre de grandeur du temps d'exécution du sous-programme « delay_ms).
5) Tester le programme sur la maquette.

Câblage des boutons poussoirs et des leds sur la maquette :

Un appui sur le bouton poussoir force un niveau logique 0.

Un 0 appliqué sur une entrée led allume celle-ci.





Expérimentation n°3:

6) Créer un projet appelé « chenillard ».
7) Réaliser et compiler le programme en C (on mettra un délai de 0.5 seconde).
8) Tester le programme dans AvrStudio.

Rappel : Fonctionnement d'un chenillard :

Led 0 allumée, toutes les autres éteintes => Led 1 allumée, toutes les autres éteintes => Led 2 allumée, toutes les autres éteintes => Led 3 allumée, toutes les autres éteintes, etc… puis retour au début et continuation indéfiniment.