Microprocesseur

TP1 : découverte

Cette séance vous permettra de prendre en main le système de façon rapide.

Les étapes de cette séance seront :
- découverte de l'environnement de travail
- écriture de séquences animées sur les sorties (LEDs)
- gestion simple d'entrées/sorties

1) Environnement de travail

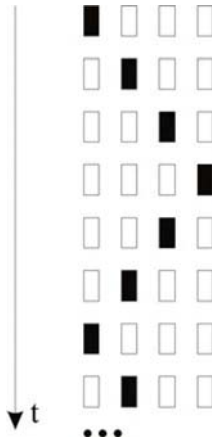
Dans cette partie, nous nous contenterons de mettre en oeuvre un projet fonctionnel et de l'exécuter sur le processeur. Nous utiliserons l'environnement IAR, présent sur la machine virtuelle *AT91SAM7* .

- Téléchargez et décompressez le projet *Démarrage_TP1* depuis l'adresse http://bornat.vvv.enseirb.fr , rubrique *E2/TP_micro-contrôleur_(MI202)*. Ce projet contient tous les fichiers nécessaires à la programmation immédiate du kit.
- Connectez votre kit au PC. Deux câbles USB vous seront nécessaires : l'un permet d'alimenter la carte processeur, l'autre permet de connecter le boîtier de programmation/débugage (boitier Bleu SAM-ICE). Si votre compte est correctement configuré, la LED du boîtier bleu doit être allumée de façon continue.
  ATTENTION : Il faut connecter le kit APRES avoir lancé la machine virtuelle pour que la liaison se fasse proprement avec les pilotes de periphériques.
- Vous pouvez alors lancer l'environnement IAR disponible depuis le menu *démarrer : IAR embedded workbench* ou *IAR systems* **>** *IAR embedded workbench for ARM kickstart* **>** *IAR embedded workbench.*
  Dans certains cas, le lien n'est pas fait dans le menu de démarrage, il faut alors chercher le programme à la main dans *Disque local (c:) \ Program Files (x86) \ IAR Systems \ Embedded Workbench 6.5 \ common \ bin * le programme est *IarIdePm*. Pour accélerer le démarrage des séances suivantes, faites un raccourci sur le bureau.
  Une fois IAR lancé, il faut charger le projet à utiliser, choisissez *File > Open > Workspace...*, puis allez sélectionner le projet d'exemple que vous avez téléchargé (extension eww). Ne copiez pas ce fichier dans un autre dossier sous peine de briser les liens vers les ressources inclues dans ce dossier.
  Vous pouvez alors observer les différents fichiers impliqués dans le processus de compilation. Dans la partie *init*, les fichiers *board_lowlevel.c*, *board_memories.c* et *board_cstartup_iar.s* (assembleur ARM) servent à initialiser le mappage mémoire du processeur, et à lancer l'horloge à pleine vitesse (48MHz). Nous observerons le fichier *main.c* (dans *Software*) plus en détail dans cette première partie.
- Pour lancer la compilation du projet et/ou son exécution, vous avez accès aux différentes fonctions Compile / Make / Debug accessibles depuis le menu *project*, depuis les icônes de la barre d'outils, ou encore par des touches de raccourci (consultez le menu *project* pour les

obtenir). Pour copier le programme compilé dans le système cible, il faut entrer en mode de débuggage. Si ce n'est pas encore fait, l'entrée en mode de débuggage provoque la compilation.

Compile		CTRL + F7
Make		F7
Debug		CTRL + D

- Exécutez le programme à l'aide de la commande GO identifiée par l'icône  du menu de débuggage. Une séquence doit apparaître sur les LEDs de la carte. Cette séquence est de la forme :



Pour sortir de l'environnement de debugage, utilisez la fonction identifiée par une croix rouge, vous pouvez également utiliser les fonctionnalités classiques d'un débugger (éxécution pas à pas notamment), l'arrêt du programme sans *reset* se fait par le bouton symbolisant une main rouge, ou lorsque le fil d'exécution atteint un point d'arrêt (obtenu en double-cliquant dans la marge gauche du programme en mode d'édition).

- Pour obtenir cette séquence, le programme fait appel à une fonction d'attente (nommée *wait*). Modifiez l'attente pour accélérer ou ralentir le déroulement de la séquence et ainsi vérifier que l'ensemble de votre environnement est fonctionnel.

Déroulement du programme de démonstration

Pour dialoguer avec les composants extérieurs au processeur, il faut écrire à certaines adresses mémoires spécifiques. Derrière chaque adresse mémoire se trouve un registre spécifique qui permet de configurer un périphérique ou de lui envoyer des données. Ces adresses sont définies par des constantes aux noms génériques dans le fichier d'en-tête du projet (AT91SAM7X-EK.h) et sont regroupées selon les modules concernés.

Le programme de démonstration ne fait appel qu'à deux périphériques externes :
- l'unité de contrôle de l'énergie (Power Management Controler - PMC)
- l'unité d'entrées/sorties Parallèles (Parallel Input Output – PIO).

Le PMC permet d'activer le PIO en lui fournissant l'horloge, le PIO permet de contrôler les broches du microcontrôleur, et donc les LEDs de la carte. Les adresses de base des périphériques, ainsi que

les numéros d'identification de chaque périphérique sont définis à la fin du fichier AT91SAM7X256.h appelé par AT91SAM7X-EK.h (il est cependant déconseillé d'utiliser ces adresses directement)

L'adresse de base du PMC est déclarée comme AT91C_BASE_PMC. Le PMC est un module bas niveau identifié comme module système, son identifiant (numéro de périphérique) est donc AT91C_ID_SYS. Le microcontrôleur AT91SAM7X256 possède deux PIOs notés PIOA et PIOB. La documentation de la carte (accessible sur la page dédiée à cet enseignement) montre que les LEDs sont branchées sur les broches de PIOB. Son adresse de base est AT91C_BASE_PIOB et son identifiant est AT91C_ID_PIOB. Cependant, pour aider à la lecture du programme, on utilisera les déclarations plus claires, spécifiques à la carte (et non au microcontrôleur) BASE_PIO_LED et ID_PIO_LED qui sont définies dans le fichier AT91SAM7X-EK.h.

La description du PMC commence à la page 179 de la datasheet. Son interface commence en page 188. Le registre qui nous intéresse plus particulièrement est PCER qui active l'horloge pour les modules correspondant aux bits écrits à 1. Pour rappel, écrire des bits à 0 dans ce registre est sans effet (cf datasheet et introduction)

En syntaxe C, l'entier 1 << ID_PIO_LED est le masque du bit représentant le PIO qui gère les LEDs (cf introduction). L'accès PCER est référencé par le champ PMC_PCER pointé par la structure du PMC (défini ligne 434 du fichier AT91SAM7X256.h)
Ainsi, la ligne suivante permet d'activer le PIO qui gère les LEDs.

```
AT91C_BASE_PMC->PMC_PCER = 1 << ID_PIO_LED;
```

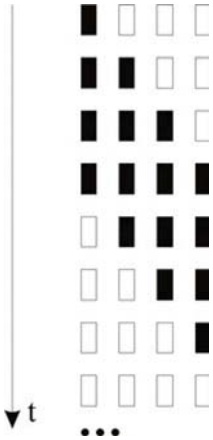
Maintenant que le PIO est activé, il est possible de le configurer. Référez-vous à la page 227 pour le fonctionnement du PIO. Vous en trouverez le schéma fonctionnel page 230 et la description des registres en page 236.

- le registre PIO_PER permet d'affecter les broches au PIO (plutôt qu'aux autres modules susceptibles d'accéder à cette broche) et donc de les contrôler directement par le programme.
- PIO_OER configure les broches associées en sortie
- PIO_SODR et PIO_CODR permettent de changer l'état des broches (Set et Clear) cf introduction, partie 4).

ATTENTION: Les LEDs sont montées en logique inversée sur la carte.

2) Écriture d'un premier programme

Affichez une séquence pour laquelle les LEDs sont allumées une par une puis éteintes de la même façon, comme indiqué sur la figure ci-dessous



Encore une fois, obtenir une séquence fonctionnelle n'est qu'une partie du travail demandé. Vous êtes tenus de fournir un algorithme et de justifier les accès au matériel que vous utilisez.

3) lecture d'évènements en entrée

L'usage d'un processeur dans le seul but de manipuler des LEDs étant d'un intérêt très limité, nous vous proposons maintenant de concevoir un système qui réagit à des directions du joystick.

a) Accès au matériel.

Le joystick fonctionne comme cinq boutons poussoirs : un pour chaque direction et un cinquième qui constitue le bouton central. Les cinq signaux sont gérables individuellement par l'intermédiaire du PIOA. Comme pour les LEDs, par souci de lisibilité, le fichier AT91SAM7X-EK.h définit les constantes BASE_PIO_PUSHB et ID_PIO_PUSHB pour l'adresse de base et le numéro d'identification. Référez-vous à ce fichier pour les autres définitions utiles.

Pour lire les différents boutons il faut que les broches du PIO soient configurées comme des entrées. Comment effectuer cette opération ? (registres concernés, et action sur ces registres)

Comment consulter la valeur que le joystick envoie sur chacune des broches ?

b) Recopie d'un bouton sur les LEDs.

Les boutons du joystick sont également gérés en logique inversée.
Écrivez un programme qui allume les 4 LEDs quand le joystick est poussé vers le haut, et qui les éteint quand on abaisse le joystick. (n'oubliez pas d'activer le PIOA / PIO_PUSHB). Lorsque le joystick n'est pas manipulé, les LEDs doivent garder leur état précédent.

c) Gestion d'évènements.

Écrivez un autre programme qui allume une et une seule LED. La LED allumée se déplace de gauche à droite ou de droite à gauche selon que l'on oriente le joystick à droite ou à gauche. Chaque appui (à gauche ou à droite) ne doit provoquer le décalage que d'une seule position. (un appui très long ne doit décaler qu'une seule fois et deux appuis brefs doivent provoquer deux décalages successifs, soit un décalage de deux positions au total)