ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

SYSTÈMES EMBARQUÉS (TP)

Nom	Prénom	Table

Examen pratique

Durée : 1h30

Ressources autorisées :
- Tous documents
- Internet

Note

Ce document est un extrait d'un ancien sujet d'examen de Travaux Pratiques.

À part le fait d'avoir été tronqué (et des questions bonus à la fin), il est strictement identique à ce qui a été proposé en examen, afin de vous proposer une expérience pleinement immersive !

Consignes

Appelez l'enseignant pour valider ***CHAQUE*** étape.

Ne restez pas bloqué trop longtemps : appeler l'enseignant pour des indices ou corriger une erreur persistante.

Prenez le temps de lire le fichier `main.c` et les autres fichiers du BSP pour s'approprier l'existant.

Toutes les réponses sont à intégrer dans le fichier `main.c` (sauf si des modifications de périphériques sont demandées).

En fin de séance, vous déposerez votre fichier `main.c` (et uniquement ce fichier) sur l'espace Moodle dédié.

1. Création Projet

10 min

Téléchargez l'archive d'évaluation « `eval_fisa_squelette.zip` » depuis la plateforme Moodle.

Renommez-la en « `eval_nom_prenom.zip` », puis l'extraire sur le disque dur (C://) ou réseau (Z://).

Créez un projet MPLAB dans le répertoire `eval_nom_prenom/apps/eval_tp/pjct` .

Ajoutez au projet le fichier source `eval_nom_prenom/apps/eval_tp/src/main.c`, ainsi que **tous** les fichiers pilotes contenus dans les différents répertoires de `eval_nom_prenom/bsp/`.

Assurez-vous de la **bonne compilation du projet**.

Appelez l'enseignant pour validation.

2. Initialisation des GPIO/LEDs

10 min

Les huit LEDs de la « *Daughter Board Curiosity Display ENSICAEN* » sont reliées aux huit broches du port B du PIC18 (RB0 à RB7).

Créez une unique fonction d'initialisation `leds_init()` de sorte à ce que toutes les broches du port B puissent piloter les LEDs.

Dans le `main()`, appelez cette fonction d'initialisation puis allumez toutes les LEDs.

Appelez l'enseignant pour validation.

Informations pour la suite de l'examen

/!\ Pour la suite, les questions peuvent être traitées dans l'ordre qui vous importe. /!

C'est pourquoi il est impératif d'appeler l'enseignant pour chaque validation.

Tous les réponses sont implémentés sur la carte Curiosity de l'enseignant. Vous pouvez demander une démonstration du résultat attendu si la consigne n'est pas assez claire.

3. Clignotement des LED

5 min

Déclarez et définissez une fonction `mode_clignotement()` qui impose un clignotement de l'intégralité des LEDs toutes les 200 ms (vous utiliserez la fonction assembleur `delay_200ms()`).

Testez la fonction dans le `main()` et appelez l'enseignant pour validation.

4. Chenillard

10 min

Déclarez et définissez une fonction `mode_chenillard()` qui réalise un chenillard : un seule LED s'allume, puis toutes les 200 ms on décale à gauche la LED allumée. Une fois la dernière LED allumée, on reboucle à la première LED.

Bonus : accordé si le décalage est réalisé en assembleur (sinon vous pouvez toujours faire en C).

Testez la fonction dans le `main()` (il faudra au préalable n'allumer que la première LED au moment de l'initialisation) et appelez l'enseignant pour validation.

5. Compteur manuel (SW2)

10 min

Déclarez et définissez une fonction `mode_compteur_par_sw2()` qui réalise un compteur 8-bit dont la valeur est affichée sur les LEDs. Le compteur s'incrémente à chaque appui sur le bouton poussoir SW2 de la Curiosity (broche RC5).

Testez la fonction dans le `main()` et appelez l'enseignant pour validation.

6. Bonus

Voici des questions non proposées dans le sujet d'examen original (pour des raisons de temps), mais vous permettant d'aller un peu plus loin.

Évolution du chenillard : dans une fonction `mode_chenillard_pong()`, faire un chenillard qui, au lieu de reboucler de la dernière à la première LED en fin de parcours, fait des allers-retours (ping-pong).

Évolution du compteur : dans une fonction `mode_compteur_par_delay()`, faire un compteur 8-bit dont la valeur s'incrémente toutes les 40 ms.