TP3 – Décodage mémoire et Interruptions

1. Objectifs de la séance

- Utilisation de la carte 8051F020 – Mise en œuvre du bus mémoire
- Branchement d'un périphérique externe d'entrée sortie – Décodage mémoire
- Mise en œuvre d'une interruption externe – Configuration et Utilisation.

2. Rendus de fin de séance

Ces rendus seront faits sur le e-campus.

- Le fichier ***Base_TP3.C*** contenant l'intégralité des codes réalisés durant le TP renommé impérativement de la manière suivante : ***Base_TP3_SM1_Nom1_Nom2.ASM***
- ***La fiche de validation en séance***

3. Compétences acquises

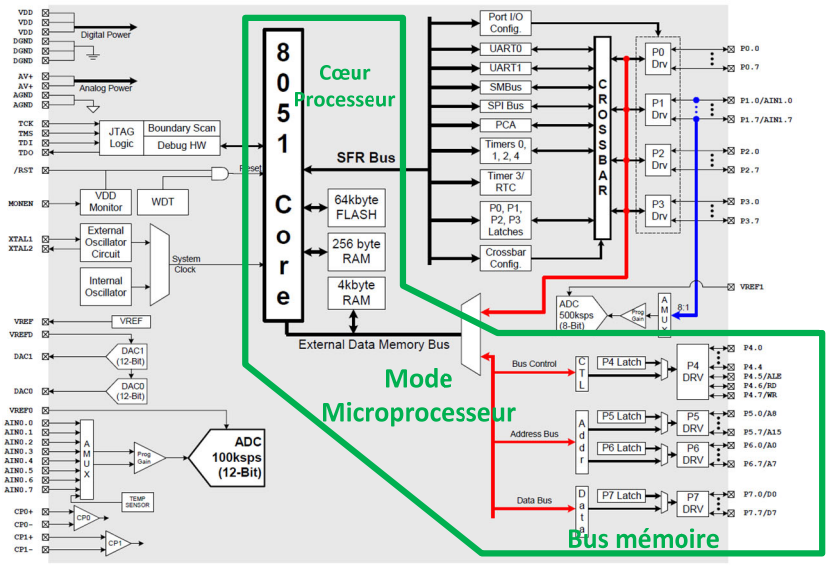
A l'issue de ce TP, vous devez pouvoir répondre par l'affirmative à ces 2 questions.

- Je sais interfacer un circuit périphérique dans l'espace mémoire XDATA du 8051
- Je sais configurer et utiliser une interruption externe du 8051F020.

Si ne pouvez pas répondre par l'affirmative à toutes ces questions, les objectifs du TP ne sont pas atteints, vous devez donc veiller à combler ces lacunes avant la prochaine séance de TP.

4. Le contexte du TP - Le 8051F020 en mode microprocesseur.

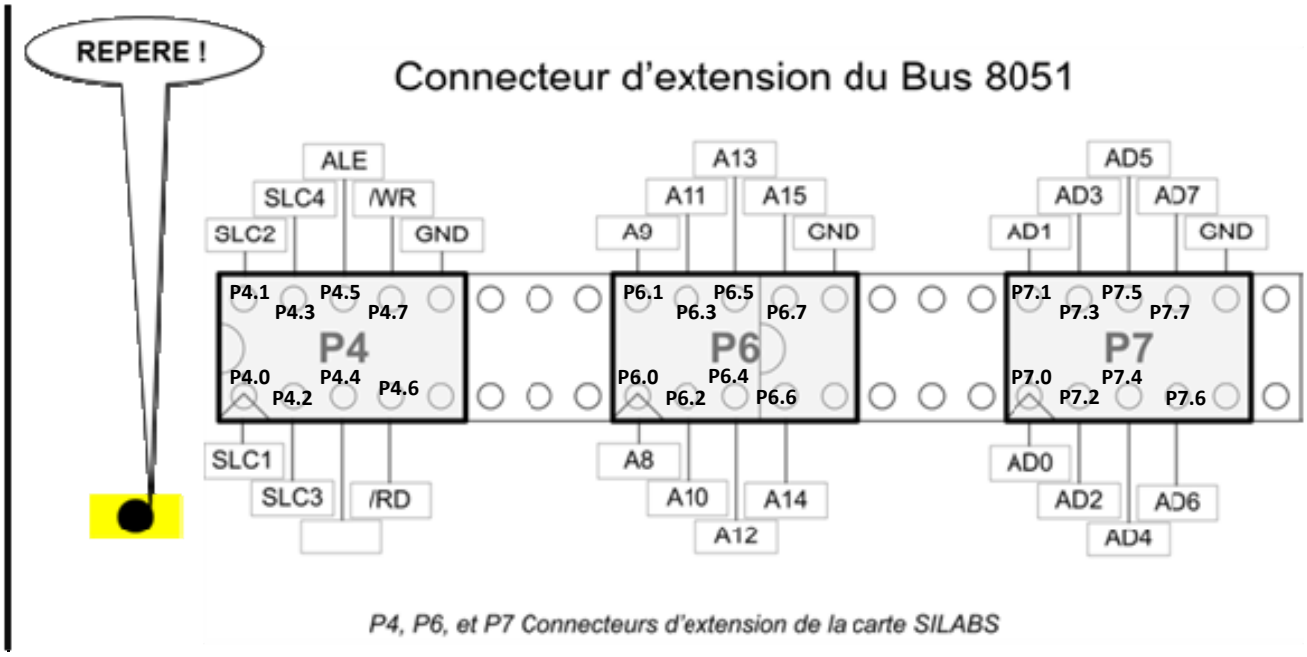
Vous allez utiliser la carte d'évaluation C8051FX20-TB qui intègre un microcontrôleur de type 8051 (référence exacte du circuit : C8051F020 de Silicon-laboratories). Dans le cadre de ce TP, on fait fonctionner le 8051F020 en mode microprocesseur.
C'est-à-dire que le 8051F020 est programmé, via le programme Base_SM3.asm, fourni, pour vous donner la possibilité de faire des accès mémoire XDATA en externe. Ce dernier est ainsi en mesure de faire des accès sur la mémoire XDATA à l'extérieur du boîtier du microcontrôleur et ceci pour la plage d'adresse **1000h-FFFFh** (la plage 0-FFFh est réservée à la mémoire XDATA interne).



Remarque : Dans ce TP, orienté vers l'apprentissage des échanges avec un bus mémoire, on laisse volontairement de côté les périphériques standards du microcontrôleur, tels que ses ports d'entrées-sorties.

Le bus mémoire est configuré pour fonctionner en mode multiplexé, et l'ensemble des signaux (Adresses, Adresses/Données, /RD, /WR, ALE) est routé sur les ports P4, P6 et P7 (obtenu par configuration du microcontrôleur).
Vous aurez accès, comme indiqué sur la figure 1, aux signaux du bus de cette carte par l'intermédiaire d'un connecteur spécifique à 3 nappes.

Figure 1 : accès aux bus mémoire via le connecteur d'extension de la carte



P4, P6, et P7 Connecteurs d'extension de la carte SILABS

Circuits logiques mis à votre disposition : 74HC00, 74HC02, 74HC04, 74HC08, 74HC32,74HC86, 74HC138 et *74HC573 (les documentations sont sur le E-campus)*

5. Etape 1 - Analyse des codes transmis – Pilotage de 2 LED

Objectif de cette étape : faire clignoter 2 LED externes grâce au code fourni et grâce à votre câblage
Remarque : Pour éviter tout problème avec Microvision, nous vous recommandons de travailler en local sur le disque C dans le dossier CPE_Users.

Récupérer sur e-campus le fichier ZIP **Code_Source_TP3**. Ce dossier contient les fichiers suivants :

- Le fichier ***Projet_TP3.uvproj*** est le fichier de configuration de Uvision adapté à cet exercice.
- Le fichier ***Base_TP3.asm*** est un fichier qui contient le programme ***main*** du projet. Vous le compléterez en modifiant le programme ***main***, et en ajoutant des fonctions de configurations et d'interruption.
- Le fichier ***Lib_Base.asm*** contient divers sous-programmes et notamment un sous-programme de temporisation. Ce fichier ne devra pas être modifié.
- Le fichier **Test_IO.C** contient une fonction (un sous-programme) qui pourra être utilisée pour faire des tests lors de l'étape2.

Le code fourni a pour objectif de faire clignoter 2 LED (une rouge et une verte), au travers d'un registre externe interfacé dans l'espace mémoire XDATA. Après l'initialisation de la pile et l'appel d'un sous-programme de configuration, le programme ***main*** se limite à exécuter une boucle infinie dans laquelle on inverse l'état des LED toutes les 50ms.

Le code mis à votre disposition, gère ces deux LED au travers d'un port de sortie implémenté dans l'espace mémoire XDATA du 8051 à l'adresse 5000H.
Nous nommerons ce port de sortie le registre CTRL_LED. Le pilotage des LED se fera comme indiqué sur le figure ci-contre.
En faisant l'hypothèse dans un premier temps qu'il n'y a aucun autre dispositif dans l'espace mémoire XDATA, **imaginez une solution matérielle de branchement simlissime pour faire en sorte que les 2 LED puissent bien clignoter à la fréquence de 10Hz**, grâce au code que nous vous avons fourni. Pour cette réalisation, vous devrez impérativement utiliser un circuit 74HC573.

- Avant de commencer le câblage dessiner le schéma de votre solution, à l'aide de l'esquisse de schéma transmise.
- Faites valider votre solution schématique par un encadrant en précisant vos choix de décodage d'adresse.
- Câblez votre montage – Testez
- Faites valider par un encadrant le clignotement des 2 LED.

Registre CTRL_LED
Pilotage de 2 LED

Périphérique de sortie: 1 octet en XDATA
Adresse: 5000H

LED R	NU	NU	NU	NU	NU	NU	LED V
7	6	5	4	3	2	1	0

MSB ——— Octet ——— LSB

LED V : 1 → Led verte allumée
LED R : 1 → Led rouge allumée

NU : Non utilisé

6. Etape 2 - – Amélioration du décodage d'adresse

Objectif de cette partie : optimiser le décodage d'adresse de votre registre CTRL_LED afin qu'il ne soit vu que dans la plage 5000H-5FFFH de l'espace XDATA du 8051.

- Faites un second schéma en tenant compte de cette nouvelle contrainte de décodage.
- Câblez votre montage – Testez
- Assurez-vous du **bon fonctionnement** du décodage. Il faut vérifier que votre décodage est **réellement** opérationnel, et donc que votre registre REG_CTRL n'est vu que dans la plage 5000h-5FFFH et pas en dehors de cette plage. Vous pourrez utiliser le sous-programme Test_IO (voir explications dans l'entête du fichier Test_IO.C.
- Faites valider par un encadrant le clignotement de vos LED et le nouveau décodage.

Remarque : Le projet Microvision mis à votre disposition contient aussi un fichier Test_IO.C. Ce fichier contient un sous-programme Test_IO qui pourra être utilisé pour vérifier votre solution. Vous reporter aux commentaires dans ce fichier pour la mise en œuvre. Attention cette version de code de test n'a pas encore été validée….

7. Etape 3 - – Mise en œuvre d'une interruption externe INT7

Objectif de cette partie : Mettre en œuvre une interruption externe déclenchée par une action sur le bouton poussoir P3.7 de la carte.

Rappel de la procédure générale de mise en place d'une interruption dans le 8051

1. *Ecrire le sous-programme d'interruption ISR (Interrupt Service Routine), à la manière d'un sous-programme. Par contre, cette routine doit obligatoirement se terminer par une instruction « **RETI** » au lieu de «**RET** ».*
2. *Placer dans la table des vecteurs d'interruptions, à l'adresse réservée pour ce vecteur d'interruption, une ligne de code permettant le saut inconditionnel (type JMP) vers la routine ISR de traitement de l'interruption**.** Pour trouver l'adresse, voir le tableau des vecteurs d'interruption dans les documentations*
3. *Configurer le périphérique pour qu'il soit en mesure de produire des demandes d'interruption au moment souhaité. Dans la majorité des cas, il convient, avant d'autoriser l'interruption, de remettre à zéro, le drapeau d'interruption (Pending flag) pour éviter un premier déclenchement intempestif.*
4. *Configurer le bit (Enable Flag) dans un des registres de validation d'interruption (IE, EIE1 et EIE2) pour autoriser cette interruption**.***
5. *Choisir le niveau de priorité donné à cette interruption par configuration du bit adéquat (Priority Control) dans un des registres de gestion des priorités (IP, EIP1 et EIP2)**.***
6. *Pour terminer, autoriser la prise en charge globale des interruptions en validant le bit EA du registre IE.*

Pour cette étape 3, reprenons les 6 points de la procédure pour la mise en place d'INT7:

1. Ecrire le squelette d'un sous-programme d'interruption que nous nommerons ISR_BP et que se terminera par un RETI. Dans ce squelette, on pourra déjà penser à remettre à zéro le bit drapeau d'interruption (Pending flag) de l'INT7 en agissant sur le registre P3IF.
2. Le vecteur d'interruption de l'interruption INT7 est à l'adresse C :0X009B. Il faut donc placer un saut vers votre sous-programme d'interruption ISR_BP à partir de cette adresse (Penser « consigne de segmentation absolue »).
3. Configurer le registre P3IF pour que l'interruption se déclenche au moment de l'appui sur le bouton poussoir (front descendant).
4. Autoriser la demande d'interruption INT7 en agissant sur le registre EIE2**.**
5. La gestion de la priorité est inutile ici, car nous ne gérons qu'une seule interruption**.**
6. Pour terminer, autoriser la prise en charge globale des interruptions en validant le bit EA du registre IE.

- Codez un sous-programme de configuration de l'INT7 « Config_INT7 »
- Codez votre squelette de code d'interruption ISR_BP
- Insérez le vecteur d'interruption INT7
- Testez le déclenchement de l'interruption sur appui de bouton poussoir grâce à des points d'arrêt.
- Faites valider par un encadrant le déclenchement de l'interruption INT7.

8. Etape 4 - – Synthèse

Objectif de cette partie : Exercice de synthèse.

- Modifiez votre code pour qu'à chaque appui sur le bouton poussoir, on commute entre 2 vitesses de clignotement des LED : une vitesse rapide 10Hz et une vitesse lente 1Hz
- Testez le bon fonctionnement de votre code.
- Faites valider par un encadrant la commutation du clignotement sur interruption.