# 3 IRC
# Mise en œuvre d'un système à microprocesseur
## Quelques rappels

1. **Objectifs globaux du module :**
   - Séance 1 : Prise en main de Microvision. Premiers codes en C
   - Séance 2 : Programmation des entrées-sorties.
   - Séance 3 : Mise en œuvre d'interruptions
   - Séance 4 : Mise en œuvre de Timer
   - Séance 5 : Mise en œuvre d'UART
   - Séance 6 : Mise en œuvre de périphériques analogiques DAC et ADC
   - **Séance 7 : Evaluation individuelle - Mise en œuvre d'un périphérique**

2. **Documentations disponibles sur e-campus :**
   - Supports de cours
   - Initiation à Microvision
   - Documentation Microcontrôleur 8051
   - Architecture et outils de développement
   - Datasheet 8051F020
   - 8051 Instruction Set
   - Architecture 8051
   - Langage C et microcontrôleur 8051

3. **Organisation des TP**

   Les TP se font en binômes, voire en trinômes (selon les contraintes de salle)

4. **Répertoire de travail :**

   Travailler de préférence en local. Projet Microvision sur le disque dur du PC
   Placer le projet dans le dossier : **C:\CPE_users\TP_IRC\TPn\noms_étudiants**

5. **Evaluation**

   La note finale de TP est établie en fonction des résultats observés en TP et d'une évaluation individuelle en dernière séance

Pour nous contacter : nacer.abouchi@cpe.fr, francois.joly@cpe.fr

Mise en œuvre d'un système à microprocesseur

# TP2 - Périphériques d'entrée-sorties

## 1. Objectifs de la séance

- Utilisation de périphériques élémentaires : les ports d'entrée-sortie

## 2. Rendus de fin de séance

Ces rendus seront faits sur le e-campus.

- Le fichier ***Main_TP2.C*** contenant l'intégralité des codes réalisés durant le TP renommé impérativement de la manière suivante : ***Main_TP2_Nom1_Nom2.C.***

## 3. Compétences acquises

A l'issue de ce TP, vous devez pouvoir répondre par l'affirmative à cette question.

- Je sais configurer et utiliser un port d'entrée-sortie sur un 8051F020 (utilisation en entrée, en sortie Push Pull, Drain ouvert)

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

Pour ce TP, vous allez travailler dans un dossier : **C:\CPE_users\TP_IRC\TP2**
Récupérer sur e-campus le fichier ZIP Code_Source_TP2. Ce dossier contient les fichiers suivants :

- Le fichier ***Projet_TP2.uvproj*** est le fichier de configuration de Microvision adapté à cet exercice.
- Le fichier ***Main_TP2.C*** est un fichier C, qui contient la fonction ***main***. Vous le compléterez en modifiant d'une part la fonction ***main***, et en ajoutant d'autre part des fonctions de configurations et d'interruption.
- Le fichier ***Lib_Base.C*** contient les fonctions de configuration du microcontrôleur. Ce fichier ne sera pas modifié.
- Le fichier ***TP1_Lib_ASM.asm*** contient des fonctions écrites en assembleur et appelables par des programmes C. Ce fichier ne sera pas modifié.

Le code fourni a pour objectif de faire clignoter la LED verte placée sur la carte de développement. Après une initialisation du Timer chien de garde, de l'horloge et de la matrice d'interconnexion, le programme ***main*** se limite à exécuter une boucle infinie dans laquelle on inverse l'état des signaux LED et Tst4 à une cadence dictée par une fonction de temporisation.

## 4. Activités durant la séance

### 4.1. Activité 1 – Configuration du Crossbar

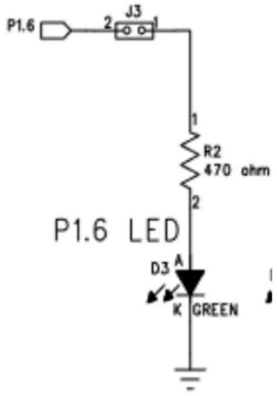
Compte tenu de fichiers source transmis, déterminez la configuration du Crossbar. Vérifiez au minimum la configuration en visualisant le signal SYSCLK.
Puis modifiez la configuration du Crossbar pour ajouter le périphérique UART1. Combien de broche va-t-il utiliser ? Sur quelles broches, les signaux de UART1 sont-ils affectés ? Quelle est l'incidence sur les affectations précédentes ? vérifiez en visualisant SYSCLK.

### 4.2. Activité 2 - Utilisation des ports P0 à P3

**Configuration en sortie**

- Sans modifier une seule ligne de code, compilez le code et faites-le s'exécuter sur la carte.
- Bien que le code s'exécute correctement (un point d'arrêt placé dans la boucle infinie montrera que le processeur exécute bien le code à l'intérieur de cette boucle), vous constaterez que la LED reste éteinte.
- Pour tenter de résoudre le problème, commencez par observer à l'oscilloscope le signal LED (P1.6) et Tst4 (P3.4).

- Bilan de l'observation :
  - Les signaux observés sont-ils périodiques ?
  - La fréquence des signaux est-elle compatible avec une fréquence de clignotement observable ?
  - Quelle est l'amplitude des signaux ? les amplitudes des signaux LED et Tst4 sont-elles identiques ?
  - La diode LED pour s'allumer a besoin d'une tension supérieure à 2,2V ? Est-ce le cas ?
  - Déterminez la configuration de la broche LED et Tst4 en recherchant l'information dans le code.




- Concluez et agissez sur le fichier source Main_TP2 pour corriger le problème.
- Vérifiez expérimentalement et observez de nouveau les signaux LED et TST4.

**Configuration en entrée**

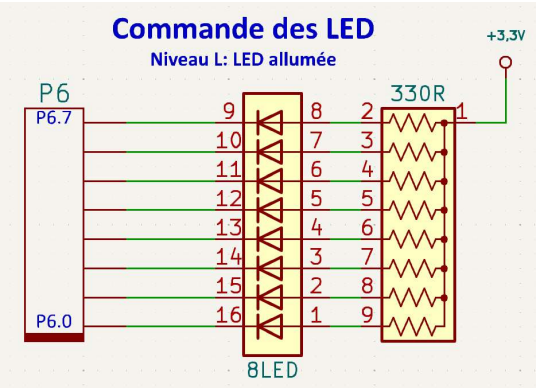
Modifier le fichier Base_TP3_IRC afin d'obtenir le fonctionnement suivant :

- Au démarrage, sans action sur le bouton poussoir (P3.7), la LED clignote.
- Mais toute action sur le bouton poussoir éteint la LED. La LED reste éteinte tant que l'on appuie sur le bouton poussoir. La détection de l'appui sur le bouton poussoir est gérée par scrutation.

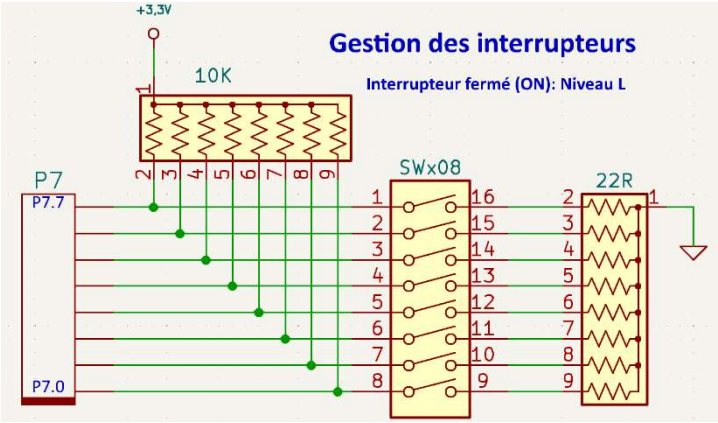
## 4.1. Activité 3 – Utilisation des ports P4 à P7

Dans les activités 3 et 4, vous utiliserez des LED pour visualiser l'état des ports et des interrupteurs pour entrer des informations sur les ports.

- Configurer le port P6 pour fonctionner en sortie et câbler les LEDs. Tester le fonctionnement. Compte tenu du schéma proposé, les LED s'allument quand on met le port à l'état bas.




- Configurer le port P7 pour fonctionner en entrée et câbler les interrupteurs. Tester le fonctionnement



- Configurer P4.0 en entrée et P4.7 en sortie quelle est la conséquence sur les autres broches du port4 ?
- Lire P7 et envoyer le résultat sur P6 selon l'état de P4.0 (si P4.0=1 alors P6=P7, sinon P6=/P7). A chaque lecture de P7, complémenter P4.7.

## 4.1. Activité 4 – Application de synthèse

Modifier le câblage afin de relier aussi bien les LED que les interrupteurs sur le même port (P6) selon le schéma proposé ci-dessous.
Coder un programme qui permet de faire fonctionner les LED en mode chenillard programmable.
La programmation sera réalisée grâce aux 8 interrupteurs selon l'encodage suivant :

- Bit P6.0 : sélection du sens de défilement
- Bit P6.1 : Marche/Arrêt du chenillard
- Bit P6.2 à P6.4 : sélection de la vitesse de défilement de 50ms à 400ms par pas de 50ms
- Bit P6.5 à P6.7 : défilement de 1 à 7 LED allumées.

Pour parvenir à coder cette application, il va falloir penser à alterner la configuration du port P6 en entrée et en sortie (entrée afin de lire les interrupteurs et sortie pour piloter les LED.
Par ailleurs l'allumage des LED sera inopérant tant qu'un des interrupteurs restera en position fermée (niveau 0).
La validation de la lecture des interrupteurs sera obtenue lorsqu'on appuiera sur le bouton poussoir relié à P3.7
Pour vous aider, une fonction nommée *Gestion_chenillard* est à votre disposition dans le fichier Lib_Base.C.