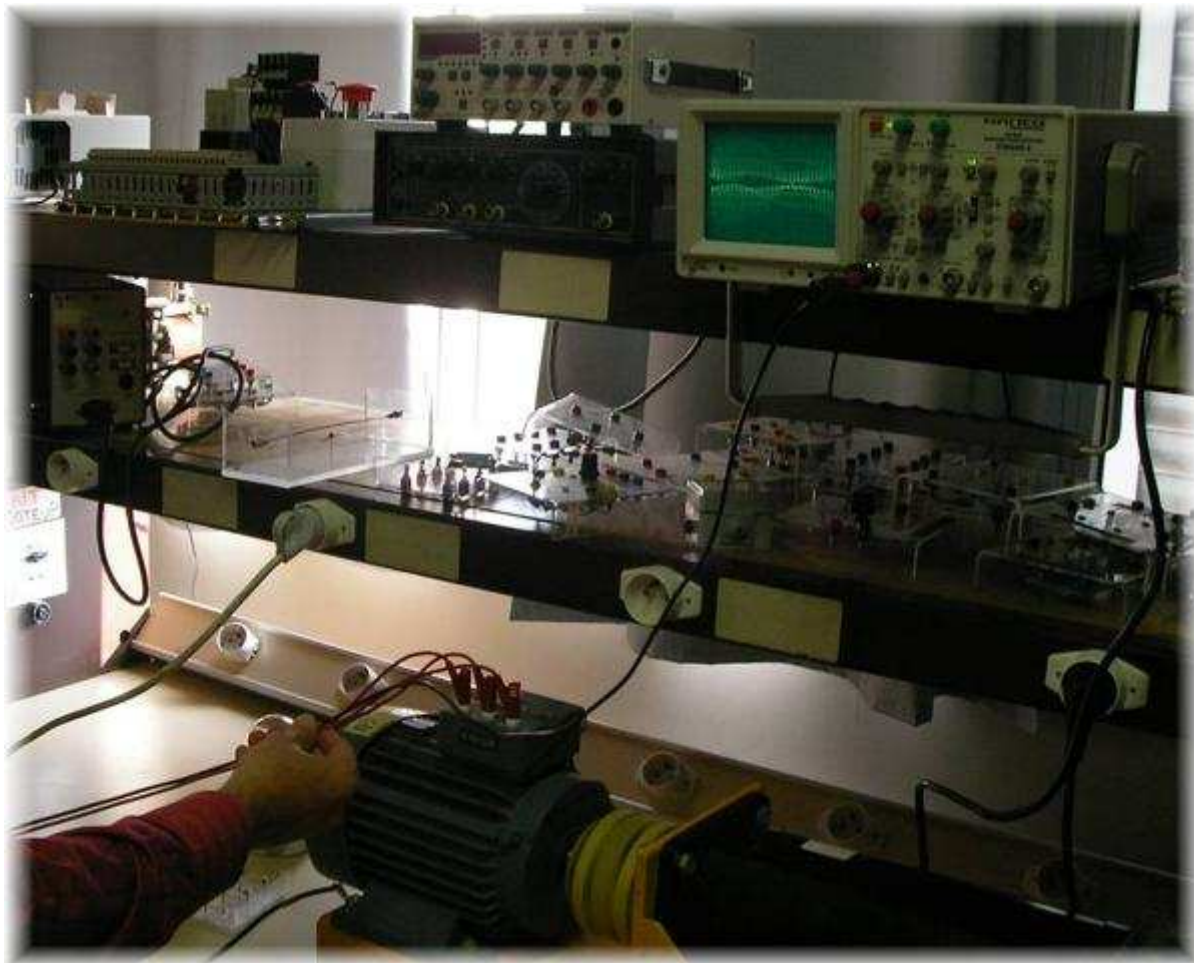


SAM1A – TRAVAUX PRATIQUES

V2.0



 **Ecole Nationale
Supérieure des Mines**
SAINT-ETIENNE
CMP - Georges Charpak

Christian Dupaty
Professeur de génie électrique
Académie d'Aix-Marseille

Plan d'action pédagogique, cours SAM1A – première partie : l'assembleur

8 séances de 3h par groupe (24h)

Test écrit 1h30 coef3 + projet coef1

S1 : Cours uC 8051, modes adressages, organisation mémoire, ports parallèles, modes d'adressage, sous programmes

S2 : fin du cours et TP1 prise en mains de l'outil de développement, du simulateur, gestions des ports parallèles

S3 : fin du TP1, tests (<=>), sous programmes, retenue (CY), lecture d'un algorithme.

S4 : TP2 utilisation avancée du simulateur, mouvements de données, gestion d'une table,

S5 : TP3 présentation de VSM (ISIS, uVision2) Approfondissement, branchements, calculs arithmétiques et logiques, révisions.

S6 : TP 4 et Cours interruptions (INT0, 1)

S7 : TP5 et Cours TIMER 0,1

S8 : TP6 et Cours TIMER 2

Sommaire

TP n°1.	Prise en mains, simulateur, ports //, tests et boucles, codage à partir d'un algorithme.	3
TP n°2.	Gestion mémoires, tableaux.....	8
TP n°3.	Simulateur, ISIS-VSM, révisions	10
TP n°4.	Interruptions INT0, INT1	12
TP n°5.	TIMER modes 0, 1, 2, production de signaux, PWM	12
TP n°6.	TIMER, capture, mesure de durées, de périodes	13



La documentation est nombreuse. Il est indispensable de la connaître et de savoir retrouver la bonne information pour réussir les exercices des TP.

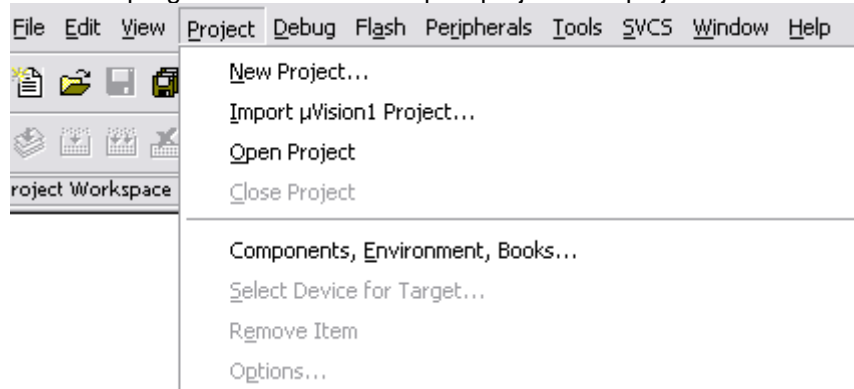
- **Prise en main de uVISION2**
GS51.pdf (document KEIL)
- **Prise en main de PROTEUS/ISIS**
Guide_Isis_v7.pdf (le minimum pour utiliser le simulateur PROTEUS-ISIS)
- **Assembleur 8051**
A51.pdf (le databook de l'assembleur KEIL)
Instruction_set_C500.pdf (documentation simple et claire des instructions du 5051)
- **Organisation mémoire et périphériques intégrés**
c517a_um.pdf (description simple et claire de la mise en œuvre des TIMER)
INTEL_MCS51_52.pdf (doc complète)

TP n°1. Prise en mains, simulateur, ports //, tests et boucles, codage à partir d'un algorithme

- Objectifs : prise en main de l'environnement de développement KEIL uVISION 2, utilisation élémentaire de l'assembleur
- Durée : 3h
- Conditions : PC + uVISION, mode simulation

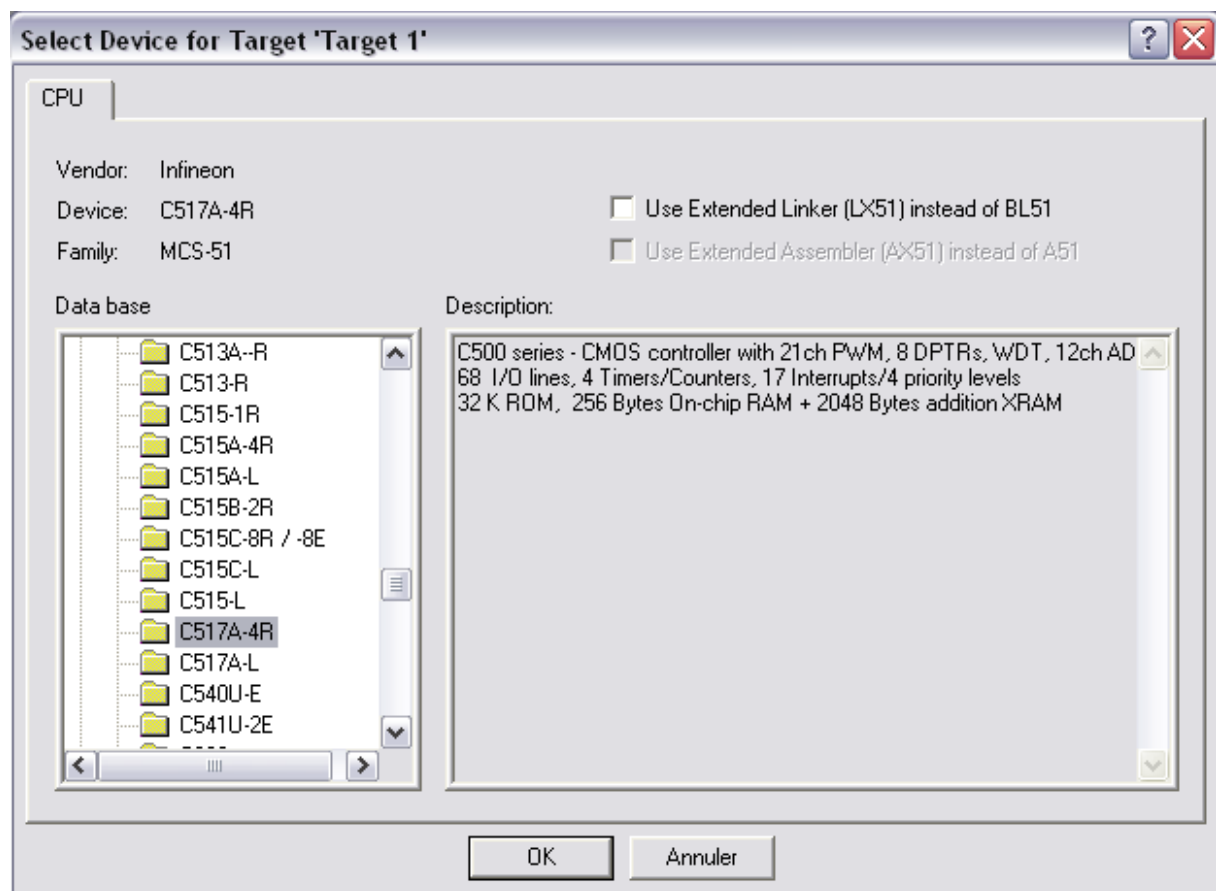
Exercice 1. prise en main

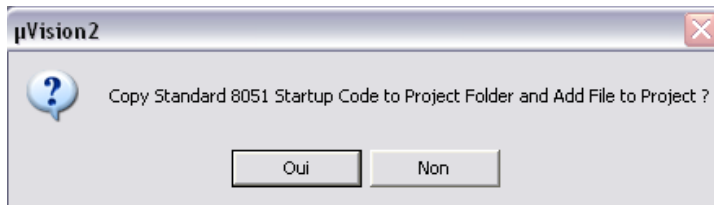
Lancer le programme « uVISION » puis project-new project



Dans le dossier « projets » créer un projet « tpsmu2 »

Choisir ensuite le microcontrôleur « C517A-4R » du fondateur « INFINEON », c'est une des versions du 8051 les mieux équipés. (Le uC8051 simulable dans PROTEUS ISIS 7.5 est le Phillips 80/87C51)

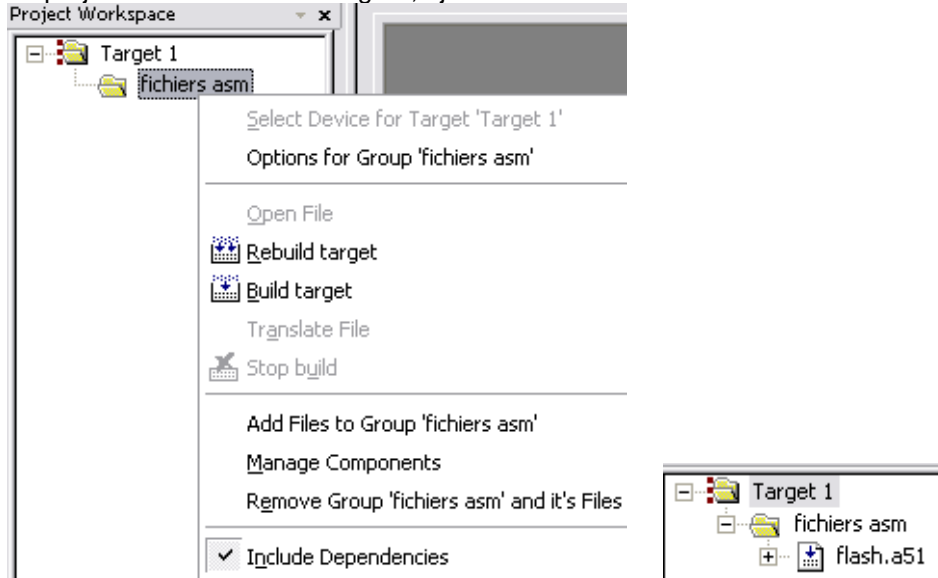




Répondre NON.

Renommer « Source Group1 » en « fichiers asm ». (deux clics sur le nom du dossier)

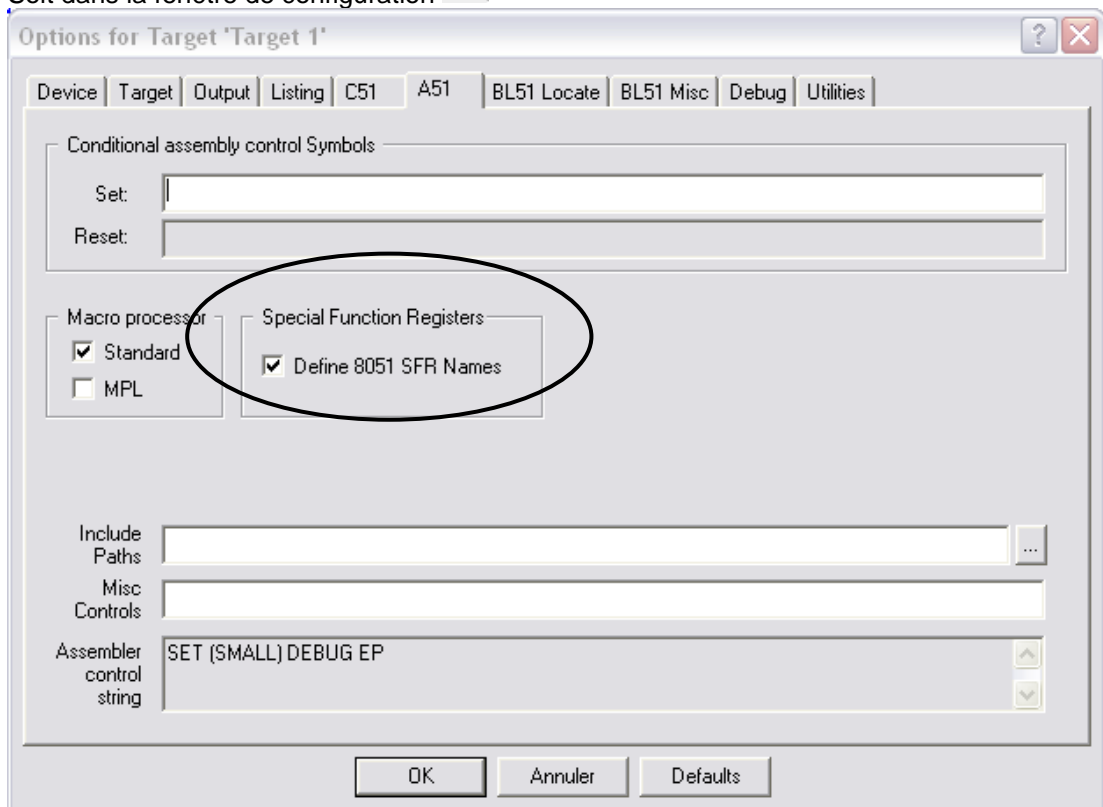
Le projet est maintenant configuré, ajouter le fichier source « flash.a51 » à « fichier asm »



Double clic sur « flash.a51 » ouvre la fenêtre d'édition

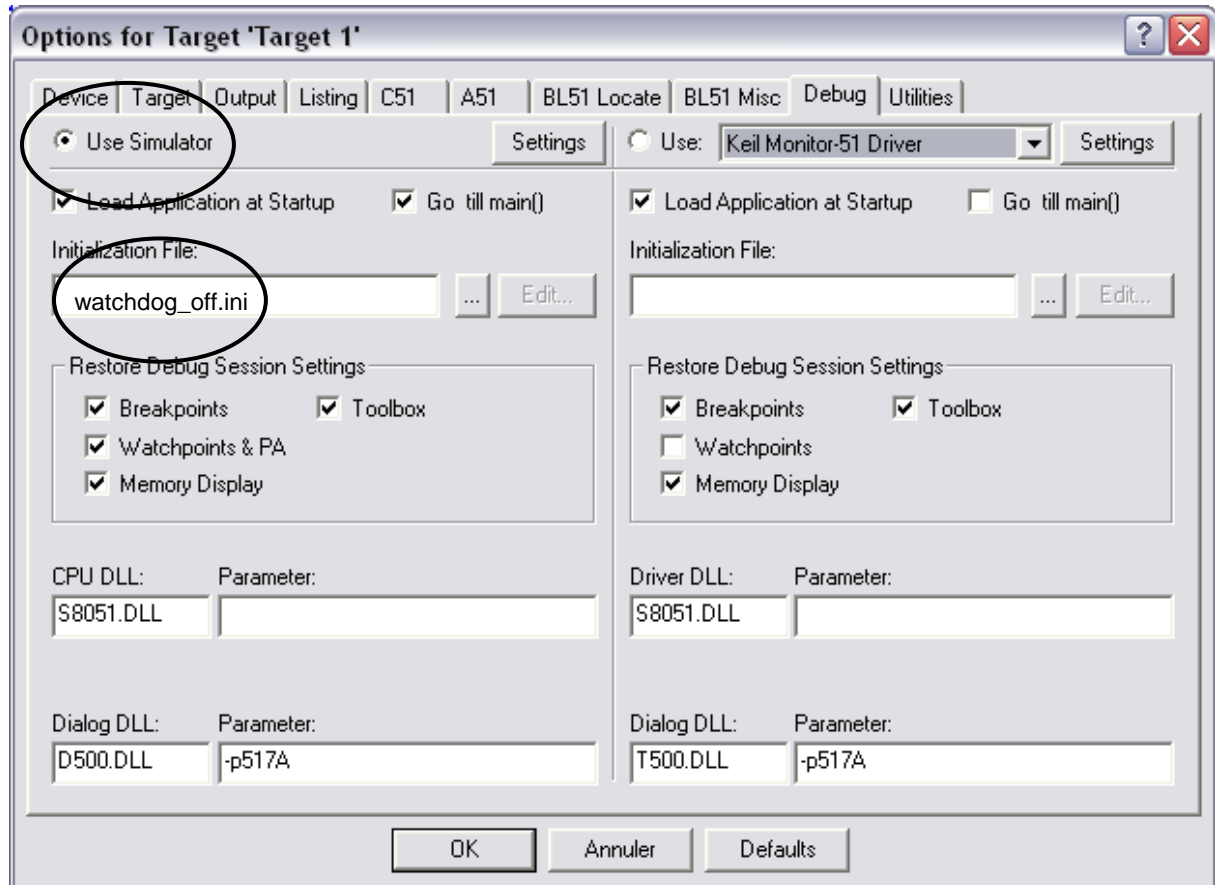
Les équivalences des registres des microcontrôleurs peuvent être ajoutées soit :

- Par la directive « #include <reg517A.inc> » dans le fichier source assembleur
- Soit dans la fenêtre de configuration



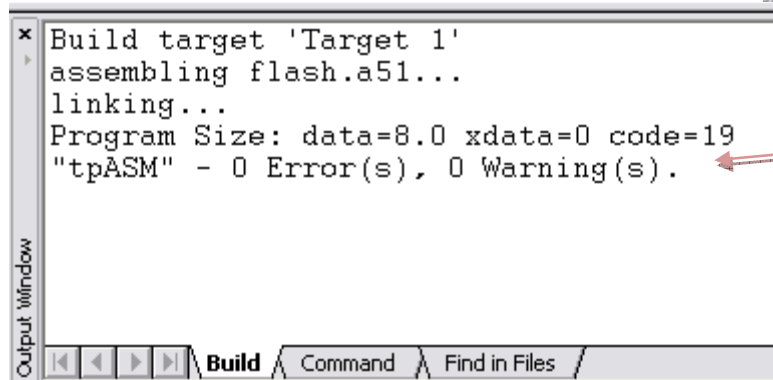
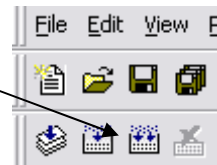
MAIS PAS LES DEUX !!!

Choix du procédé de mise au point (debug) .Sectionner « target1 » clic-droit « debug»



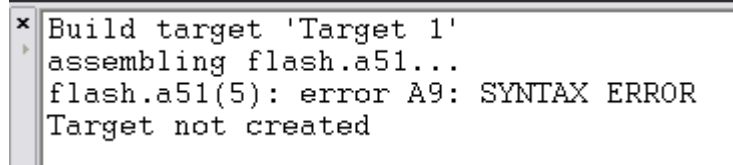
Sélectionner « use simulator » et ajouter le fichier d'initialisation watchdog.ini qui désactivera la fonction « chien de garde » inutile ici.

Pour compiler le programme : « project-build target » ou




Le compilateur a pu transformer le fichier A51 en fichier HEX (codes machine). Le linker a trouvé une place dans les mémoires pour votre programme et vos données.

Créer une erreur de syntaxe en doublant le « m » du premier « move » et recompiler.



Double-clic sur la « error A9 » positionne automatiquement le curseur sur la ligne où il y a une erreur. Corriger l'erreur, recompilez, le programme est prêt pour les tests.

Cliquer sur « stop-start-debug session » 

L'environnement de mise au point apparait (ici en mode simulateur) le microcontrôleur est remis à « 0 » une flèche jaune indique la position du compteur de programme (PC)

- Sur la partie gauche sont visibles les registres du microcontrôleur.
- En bas à gauche une fenêtre de ligne de commande
- En bas à droite une fenêtre d'édition des mémoires.

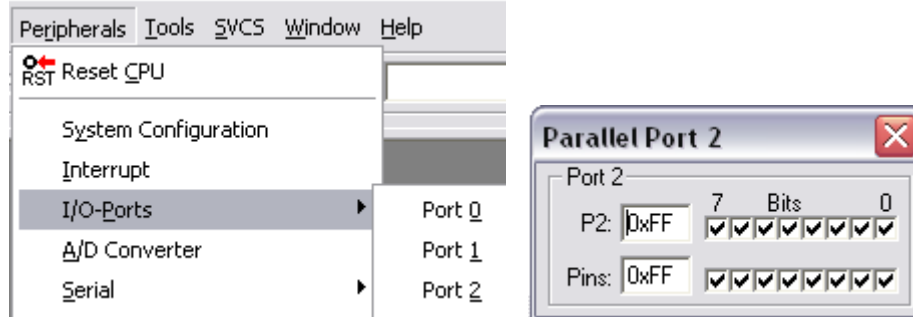



Dans cette dernière partie taper C :0, le contenu de la ROM apparaît :

```
Address: c:0
C:0x0000: 75 81 7F B2 A0 11 09 80 FA 78 FF 79 FF D9 FE D8 FA
C:0x0011: 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0022: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```






Avec les onglets « memory 2 et 3 » :
visualiser la RAM directe, D :0 , la RAM indirecte I :0 et la RAM étendue X :0

Afficher le port P2 par peripherals-I/O-Ports-Port2.



- Vérifier les options mémoire et la configuration de l'horloge par peripherals – System Configuration, en particulier la fréquence de l'oscillateur : 24MHz
- Le chien de garde est activé par défaut, pour le désactiver il est nécessaire de placer la broche PE_SWD à 0 avant le RESET. Pour cela entrer PE_SWD = 0 dans la fenêtre de commande puis cliquer RESET  (cette action est inutile si le fichier watchdog.ini a été chargé)



- Lancer le programme :  P2.0 doit clignoter.
- Arrêter la programme : 
- Double-clic devant call tempo afin d'y place un point d'arrêt 
- Relancer le programme, ce dernier s'arrête à chaque appel du sous-programme tempo.
- Tester les fonctions « step-into » et « step-over »   , indiquer leur rôle (par rapport à l'exécution du SP tempo).
- En mode pas à pas contrôler l'évolution des registres SP-R0-R1
- Quel est la durée de la temporisation ? (temps indiqué en secondes dans la fenêtre projet)
- Tracer l'algorithme du programme principal et du sous programme tempo

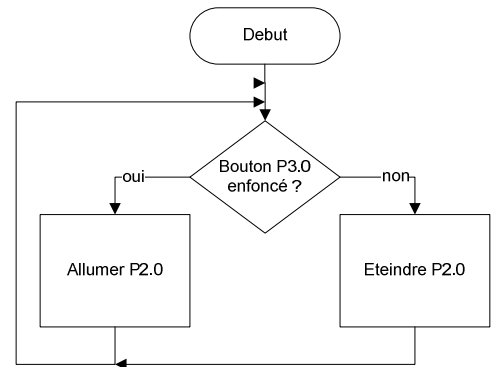
Exercice 2. test

En vous inspirant du programme flash.a51

Réaliser le programme correspondant à l'algorithme ci contre.

- P3.0=5v si le bouton n'est pas enfoncé
- Si P2.0=0, la LED est allumée

Utiliser les instructions JB, JNB, SETB, CLR, SJMP

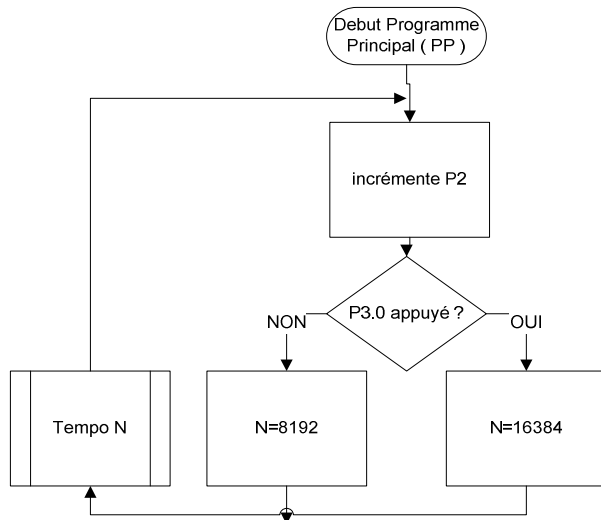


Exercice 3. Anti-rebonds

Modifier le programme précédent afin d'incrémenter P2 (instruction INC) à chaque pression sur P3.0, dessiner son algorithme.

- attendre appui
- attendre relâchement
- incrémenter
- boucler

Exercice 4. Transfert de paramètres

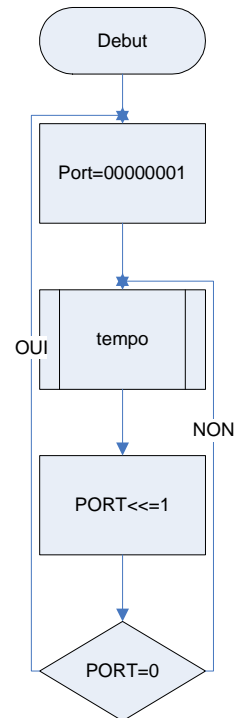


Modifier le programme flash.a51 de manière à modifier la tempo (passer de 65536 boucles à 16384 boucles) si P3.0=0. Le paramètre N sera transmis au sous programme tempo dans ACCA (Utiliser la tempo de l'ex1 ou N=R0xR1, charger ACCA dans R0 avant de lancer la tempo)

Exercice 5. chenillard

A partir du programme flash.a51, créer un programme chenille.a51 réalisant un chenillard sur P2. (instructions RL et RLC). Prendre une tempo suffisamment longue pour visualiser le chenillement.

```
00000001
00000010
00000100
00001000
00010000
00100000
01000000
01000000
10000000
00000000
```



Exercice 6. Test retenue (cy)

Créer un programme inversant le chenillard lorsque la retenue (cy) du registre PSW passe à 1 (valeurs de 1 à 80h puis de 80h à 1)

En utilisant les instructions rl, rlc,rr,rrc, setb,clr, jb,jnb , sjmp réaliser la séquence sur P2:

TP n°2. Gestion mémoires, tableaux

Objectifs : Etre capable de déplacer les données et des zones mémoires, maîtriser les modes d'adressage ainsi que l'organisation mémoire du 8051.

Durée : 3h

Conditions : PC + uVISION



Exercice 7. Recopie ports

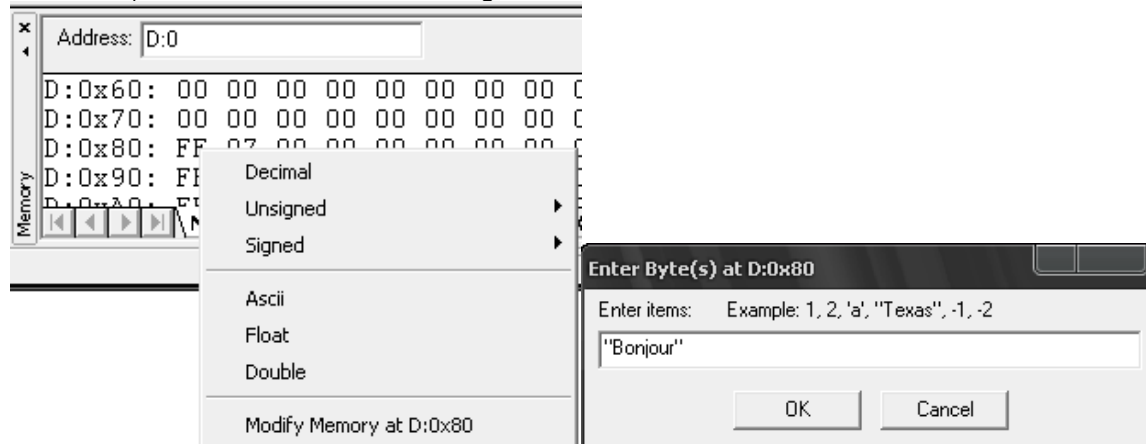
En utilisant les instructions « mov » réaliser un programme qui en permanence recopie le port3 sur le port2. Vérifier le résultat sur le simulateur.

Exercice 8. Copie RAM-RAM

Recopie d'une zone mémoire RAM-RAM.

En utilisant l'adressage indirect (registres R0, R1) recopier la zone RAM 80 à 8F vers A0 à AF. Lors des essais la zone 80-8F sera initialisée manuellement avec un texte en ASCII (dans la fenêtre en bas à droite)

En mode simulateur : afficher la RAM (D :0) sur la ligne 0x80 faire un clic-droit puis « Modify Memory at D :0x80), entrer un texte ASCII entre guillemets .



Dérouler le programme pas à pas alors de visualiser la recopie octet par octet.

Exercice 9. Copie ROM-RAM

Recopie d'une zone mémoire ROM-RAM.

En utilisant l'adressage indirect (registres R0, R1 et DPTR) recopier la zone ROM 500h à 500Fh vers la zone RAM A0 à AF. La zone ROM sera initialisée par programme avec un texte en ASCII. (Utiliser la directive d'assemblage appropriée, voir doc asm).

Comme précédemment, dérouler le programme pas à pas alors de visualiser la recopie octet par octet. (Pour visualiser la ROM dans la zone mémoire : C :0x500)

Exercice 10. Gestion d'une table

Donner l'algorithme puis réaliser le chenillard (ou toute autre animation qui vous semblera esthétique) de l'exercice 3 à l'aide d'une table de données placée en ROM (utilisation de movc et DPTR, pour la lecture).

Lecture de données, sortie, temporisation, incrémentation du pointeur, boucle.

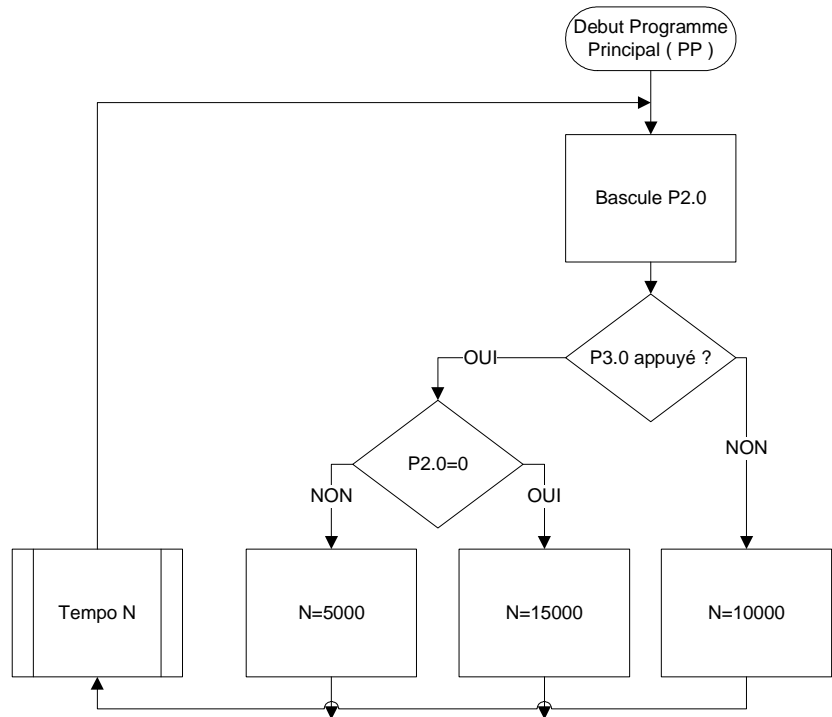
La fin de la table sera un 0.

Exercice 11. Bilan- Evaluation formative

Pour les plus rapides :

Modifier le programme flash.c de manière à obtenir sur P2.0 un rapport cyclique $\frac{1}{4}$ si P3.0=0 et $\frac{1}{2}$ sinon.

Utiliser le compteur de temps (sec) pour vérifier le rapport cyclique.



TP n°3. Simulateur, ISIS-VSM, révisions

Objectifs : Etre capable de développer et tester un programme très simple en assembleur 8051 à l'aide de uVISION et d'une maquette virtuelle sur ISIS.

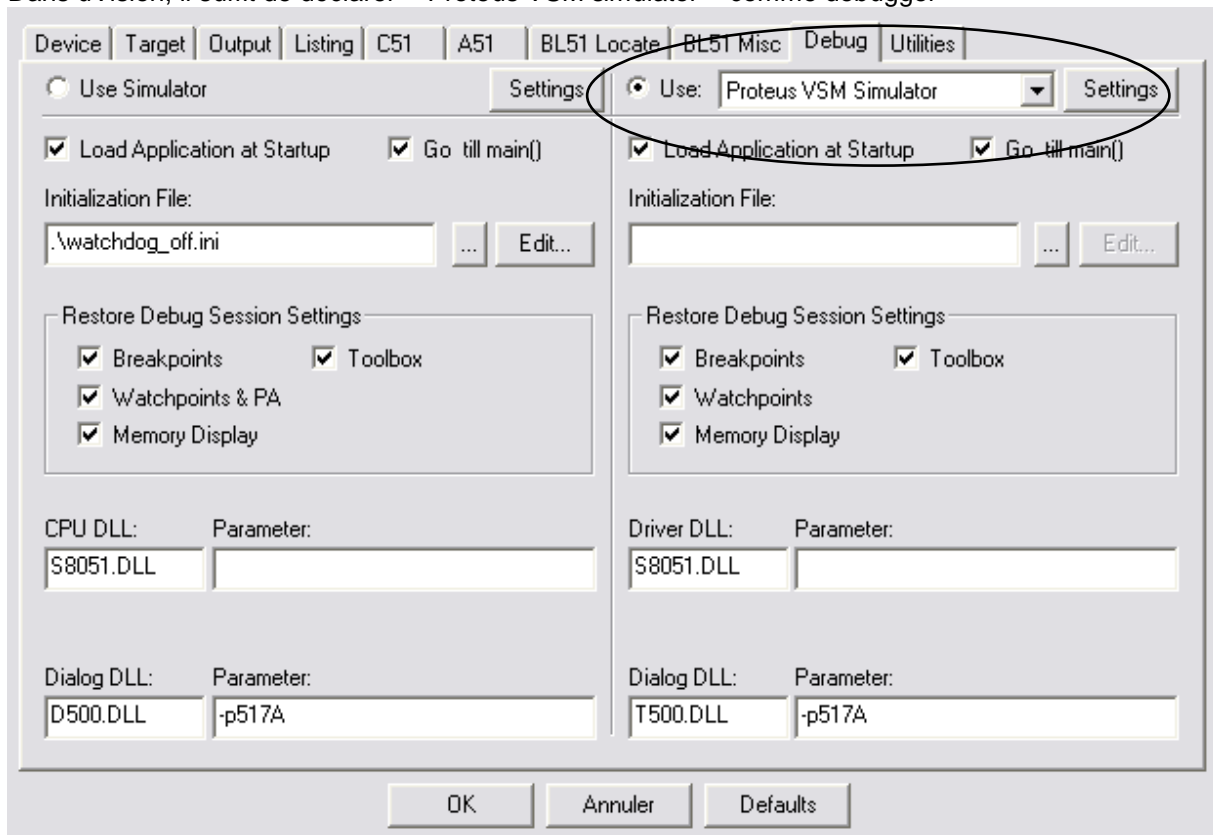
Durée : 3h

Conditions : PC + uVISION + PROTEUS-ISIS

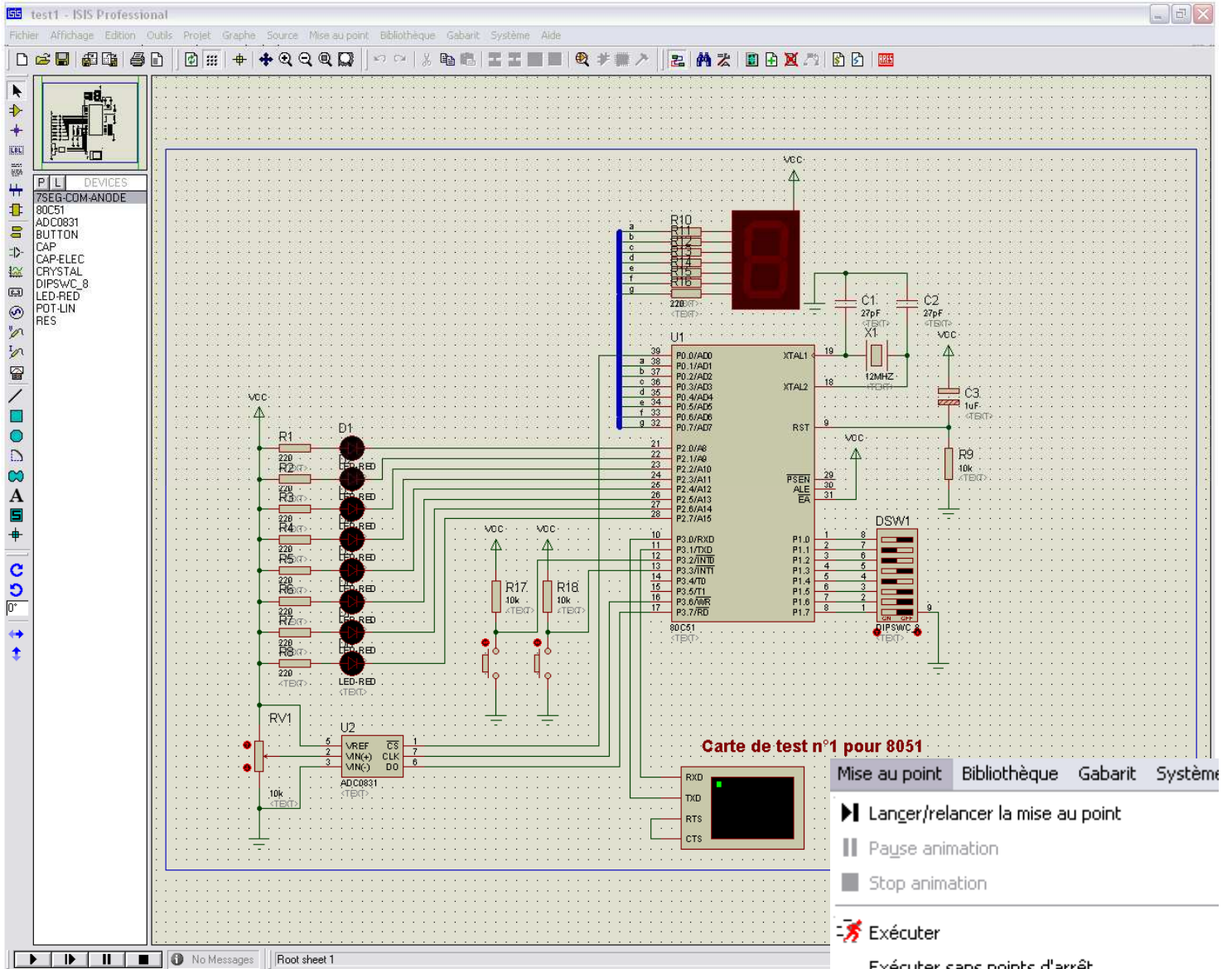
L'outil de CAO électronique PROTEUS possède un module de simulation de processeurs (VSM) permettant la mise au point de systèmes électroniques analogiques, numériques et programmés par simulation.

Un plugin doit être installé sur l'ordinateur supportant uVISION et PROTEUS : vdmagdi.exe (disponible sur <http://www.labcenter.co.uk/support/vdmkeil.cfm>)

Dans uVision, il suffit de déclarer « Proteus VSM simulator » comme debugger



Lancer ISIS-PROTEUS avec un projet comportant le microcontrôleur de uVISION (par exemple test1-isis)



Il est possible alors d'utiliser le schéma sur ISIS comme une maquette réelle. La seule contrainte est le nombre de microcontrôleurs simulables sur ISIS-VSM (*le 80C51 mais pas le C517A*)

Exercice 12. Prise en main uVISION-ISIS-VSM

Reprendre les exercices 1 à 6 et les adapter afin de les simuler sur la maquette virtuelle « tst1 »

Exercice 13. Gestion d'un afficheur 7seg, table de conversion

Réaliser un compteur hexadécimal des appuis sur P3.2 sur l'afficheur 7 segments à anode commune.

(voir <http://www.stielec.ac-aix-marseille.fr/cours/abati/aff7seg.htm>)

Pour les plus rapides, réaliser une « mini calculatrice » .

Afficher le l'afficheur 7 segment la somme N1+N2 ou un tiret s'il y a débordement.

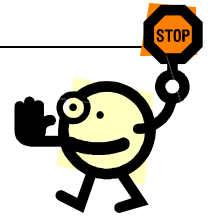
Lors de l'appui sur le bouton 1 P1 est recopié dans N1.

Lors de l'appui sur le bouton 2 P1 est recopié dans N2.

Mise au point Bibliothèque Gabarit Système

- ▶ Lancer/relancer la mise au point
- || Pause animation
- Stop animation
- Exécuter
- Exécuter sans points d'arrêt
- Exécuter pendant un certain temps
- Pas à pas principal
- Pas à pas détaillé
- Sortir de
- Aller à
- Réinitialiser fenêtres popup
- Réinitialiser modèle de données persistant
- Configurer les diagnostics...
- Utiliser moniteur de debug distant

Activer le lien IP entre ISIS et uVISION



TP n^o4. Interruptions INT0, INT1

Objectifs : Analyser le mécanisme de l'interruption. Etre capable de mettre en œuvre un programme traitant plusieurs sources d'interruptions.

Durée : 3h

Conditions : PC + uVISION + PROTEUS-ISIS

Exercice 14. Test interruptions

Tester à l'aide du simulateur le programme demoIT sur la maquette virtuelle tst1 précédente.

Placer un point d'arrêt en début de sous-programme d'interruption. Dérouler le programme en pas à pas, visualiser et interpréter l'évolution du registre SP ainsi que celle de la zone mémoire de la pile. **Analyser et expliquer chaque ligne du programme**

Exercice 15. Exercice interruption /INT1

Compléter le programme demoIT de manière à faire cheniller le PORT3 lors d'un front montant sur /INT1 (utiliser la documentation C517a_num.pdf). Tester ce programme sur la maquette virtuelle test1 (Proteus ISIS).

Exercice 16. Plusieurs IT.

A partir de demoIT réaliser un chenillard décalant une fois à gauche sur l'interruption /INT1 et une fois à droite sur /INT0. Pour cela utiliser deux sous programmes d'interruptions, un pour chaque vecteur (/INT0, /INT1), le programme principale ne faisant « rien » (boucle sans fin)

TP n^o5. TIMER modes 0, 1, 2, production de signaux, PWM

Objectifs : Etre capable de mettre en œuvre les TIMER dans tous leurs modes de production de signaux en utilisant les interruptions.

Durée : 3h

Conditions : PC + uVISION + PROTEUS-ISIS

Tous les exercices seront validés sur Proteus-ISIS par *l'utilisation d'instruments de mesure virtuels (oscilloscope, graphe)*

uVISION propose lors de la création d'un projet d'inclure le fichier « entete.a51 » qui facilite la gestion des interruptions et des segments des mémoires. Ce fichier peut être utilisé comme base d'un projet, le code utilisateur doit être placé après les lignes :

```
;-----
```

```
; Insert your assembly program here.
```

```
;-----
```

Exercice 17.

MODE 0 : produire un signal carré de fréquence 1KHz



Exercice 18.

(Effectuer une recherche pour pwm et rapport cyclique sur <http://fr.wikipedia.org/>).

MODE 0 : produire un signal carré de 100 Hz modulé en PWM dont le rapport cyclique sera proportionnel à la valeur de Port1 (0->0% , FFh-> 100%)

MODE 0 : produire un signal carré de 100 Hz modulé en PWM de 0% à 100% par interruption sur le TIMER 1 toutes les 100mS (visualisation d'un « accordéon » sur l'oscilloscope)

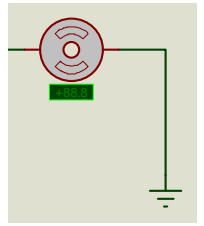
Exercice 19.

MODE 1 : Produire une impulsion positive de 30 mS lors de l'appui sur le bouton.

Exercice 20.

MODE 2 : Produire un signal rectangulaire de fréquence 200Hz et de rapport cyclique 10%.

Pour les plus rapides : modifier le schéma de la carte de test en incorporant un moteur à courant continu (motor-dc) et vérifier que la vitesse de rotation est proportionnelle au rapport cyclique.



TP n°6. TIMER, capture, mesure de durées, de périodes

Objectifs : Etre capable de mettre en œuvre les TIMER dans tous leurs modes de mesure de temps en utilisant les interruptions. (Fonction CAPTURE)

Durée : 3h

Conditions : PC + uVISION + PROTEUS-ISIS

Exercice 21.

Réaliser un programme produisant un nombre proportionnel à la période du signal sur /INT0. (Placer un signal périodique sur INT0). Indiquer les limites de la mesure avec un quartz à 12Mhz.

Exercice 22.

Réaliser un programme mesurant la durée de l'état haut du signal sur /INT1

Le traitement des données s'effectuera en interruption.

Les résultats de mesure seront stockés en RAM.

Proposer une procédure de vérification du fonctionnement, par le simulateur uVISION et par ISIS.

Comparer les deux méthodes.