TP4 - TP5 – Mini-Projet

1. Objectifs des 2 séances

L'objectif de ce mini projet est de mettre en application des notions étudiées dans le cadre du module d'introduction aux systèmes à microprocesseurs, à savoir :

- Choisir un plan mémoire dans une architecture à microprocesseur.
- Calculer et simplifier des équations de décodages mémoires et périphériques d'entrées-sorties.
- Réaliser un schéma électrique répondant à un besoin.
- Mettre en œuvre des périphériques d'entrées-sorties dans une architecture à microprocesseur.
- Assimiler le fonctionnement des interruptions internes et externes.
- Utiliser des directives d'assemblage pour le 8051F020.
- Ecrire, corriger, exécuter et debugger des programmes écrits en langage assembleur 8051.
- Savoir extraire les informations essentielles d'une documentation.

Ce mini-projet de 8 heures (2 séances) couvre les aspects matériels et logiciels.

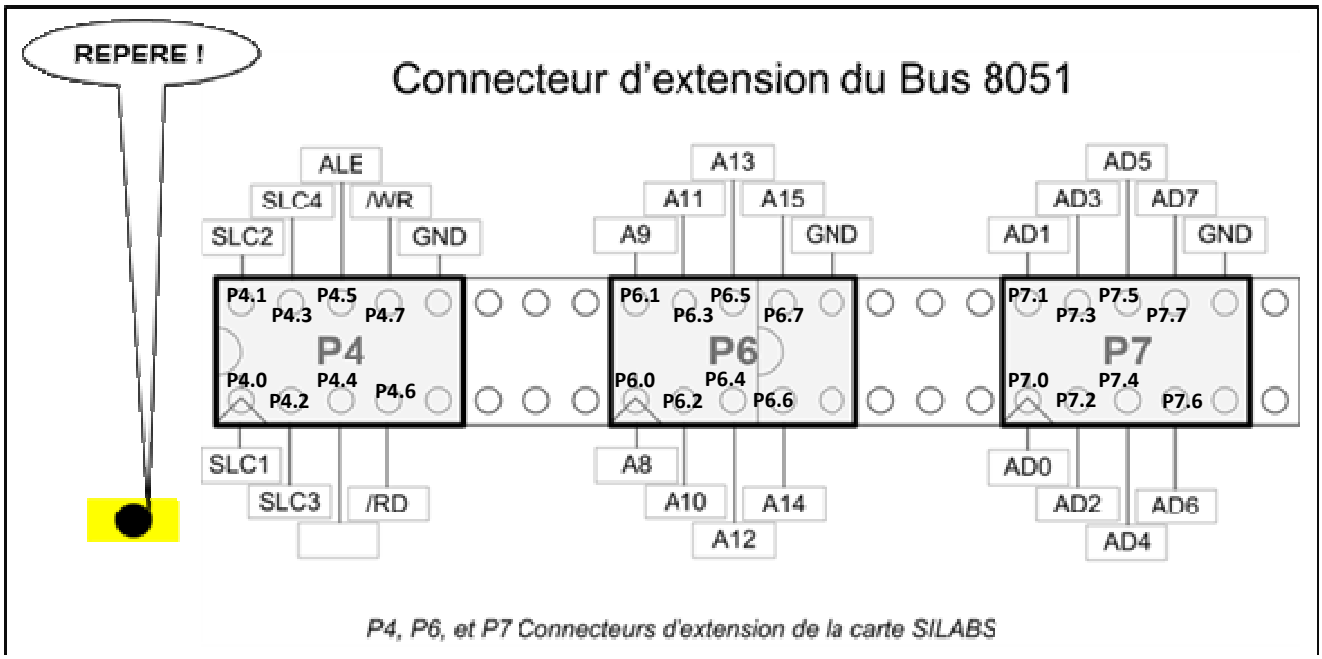
- **Du point de vue matériel**, vous devez montrer votre capacité à réaliser un schéma électrique complet pour mettre en œuvre un mini-système constitué d'un microprocesseur et de circuits périphériques. Mais, il n'y aura pas de câblage à réaliser. Les essais seront faits en simulation et sur une maquette de test.
- **Du point de vue logiciel,** vous devrez montrer votre capacité à écrire du code assembleur en maitrisant le jeu d'instructions, les modes d'adressage et les directives d'assemblage.

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

2. 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).
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

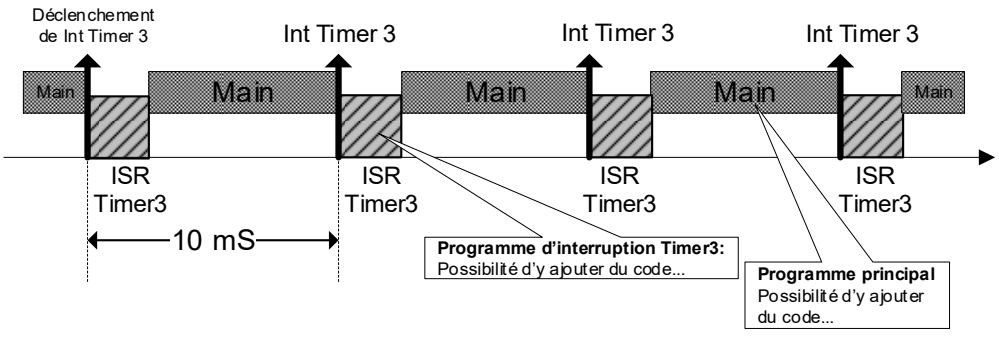


P4, P6, et P7 Connecteurs d'extension de la carte SILABS

1.1. Utilisation d'une interruption Timer

Pour faciliter le codage de l'application, nous vous fournissons un code dans lequel nous avons programmé un périphérique interne (le Timer 3) qui va provoquer une interruption toutes les 10ms. La configuration du Timer3, et tout le mécanisme de mise en place de cette interruption sont déjà présents dans le code transmis, de même que le sous-programme d'interruption.
Vous pourrez ajouter du code dans ce programme d'interruption pour répondre au cahier des charges.

Chronogramme d'exécution du programme principal et du programme d'interruption Timer 3



3. Cahier des charges du mini-projet – Gestion d'accès à un parking :

Remarque préliminaire : Ce cahier des charges est volontairement simplifié. Une application réelle serait un tout petit peu plus complexe…. Néanmoins le principe de conception reste le même.

On souhaite utiliser un système à microprocesseur basé sur le 8051F020 pour surveiller l'accès à un parking privé. Les caractéristiques et le fonctionnement du parking sont les suivants :

- Au maximum 32 personnes peuvent disposer du droit à utiliser le parking.
- Les personnes ayant le droit d'accès au parking disposent chacune d'un code d'accès unique compris entre 1 et 63.
- Le nombre de codes autorisés et la valeur de ces codes sont stockés dans la mémoire non volatile du système.
- Le parking dispose de 8 places uniquement.
- Le nombre de places occupées est continuellement affiché sur un afficheur 7 segments devant l'entrée du parking.
- L'entrée du parking est équipée d'une LED chargée d'indiquer l'autorisation d'accès ou pas.

L'entrée au parking se fait selon la séquence suivante :

- Une voiture arrive devant l'entrée ;
- Le conducteur s'identifie en rentrant son code en binaire (sur un boitier contenant des interrupteurs) ; ***en pratique : 6 interrupteurs seront utilisés pour simuler la demande.***
- Une fois le code saisi, la voiture avance alors légèrement pour déclencher un détecteur de présence. Ce dernier fait passer à 1 une entrée logique du système (nommée DCT_Park_IN) ; ***en pratique : un interrupteur permettra de simuler la détection de passage.***
- Le système vient périodiquement lire l'état de cette entrée ; ***en pratique : le système scrutera l'état de cette entrée durant une interruption Timer 3 déclenchée par le débordement du Timer (code fourni). Attention, le véhicule ne devra être détecté qu'une fois….***
    - Si elle est à 1, et si l'identifiant a saisi un des codes autorisés et si le nombre de véhicules dans le parking est inférieur à 8 :
        - Le portail d'entrée s'ouvre,
        - La LED est allumée pendant 4 secondes pour indiquer l'autorisation d'accès au parking,
        - L'afficheur est mis à jour (il indique le nombre de véhicules dans le parking),
        - Le code identifiant le conducteur autorisé est enregistré en mémoire (mémorisation de tous les accès).
    - Si elle est à 1 et que l'identifiant n'a pas saisi un des codes autorisés ou et le nombre de véhicules dans le parking est égal à 8, le portail d'entrée reste fermé et la LED clignote (Fcligno = 4Hz environ) pendant 5 secondes pour indiquer le refus d'accès.

La sortie du parking se fait selon la séquence suivante :

- Une voiture arrive devant la sortie ;
- Un détecteur détecte sa présence et fait passer à 1 une entrée logique du système (nommée DCT_Park_OUT) ; ***en pratique : un interrupteur permettra de simuler la détection de passage.***
- Le système vient périodiquement lire l'état de cette entrée ; ***en pratique : le système scrutera l'état de cette entrée durant une interruption Timer 3 déclenchée par le débordement de ce Timer. Attention, le véhicule ne devra être détecté qu'une fois…***
- Si elle est à 1, on suppose qu'un véhicule vient de quitter le parking, l'afficheur (affichage du nombre de véhicules dans le parking) est alors mis à jour.

Arrêt d'urgence:

- A tout moment, l'accès au parking pourra être invalidé. Cette interdiction d'accès sera assurée par action sur un bouton poussoir. L'accès sera de nouveau autorisé après avoir entré le code de réarmement du système (valeur 00) sur le boitier de saisie du code d'entrée. ***en pratique : c'est l'interruption externe INT7 (reliée à un bouton poussoir sur la carte d'évaluation) qui provoquera l'arrêt d'urgence.***

Règle simplificatrice :

On considère que l'on ne doit pas avoir à gérer simultanément une entrée et une sortie de véhicule.

4. Code mis à votre disposition :

Récupérer sur e-campus le fichier ZIP **Code_Source_TP4-5**. Ce dossier contient les fichiers suivants :

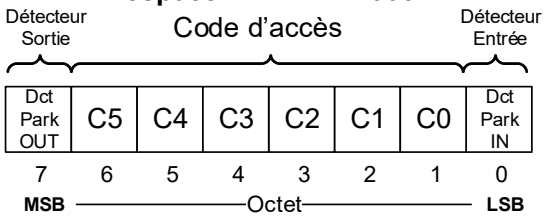
- Le fichier **Proj_TP_TP4.uvproj** est le fichier de configuration de **µvision** adapté à ce mini-projet.
- Le fichier **Lib_TP_SM.asm** est un fichier assembleur qui rassemble un certain nombre de sous-programmes assembleurs mis à votre disposition. **En aucun cas ce dernier ne doit être modifié.**
- **Le fichier Lib_VosSP_SM4.asm destiné à contenir tous les sous-programmes que vous allez devoir coder.**
- **Le fichier Main_SM4.asm, qui contiendra le programme principal de votre projet.**

Séance 4 : Conception de l'architecture matérielle, Codage des principaux sous-programmes et Tests par simulation

5. Contraintes relatives au 8051F020 :

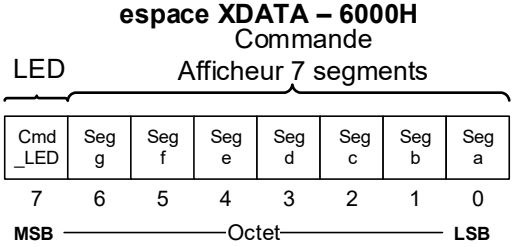
- Les 4 premier Ko de la mémoire XDATA (la plage 0-FFFh) sont réservés à la mémoire interne.
- Les 32 derniers Ko de la XDATA sont réservés pour un développement futur.
- Le port de sortie permettant le pilotage de l'afficheur 7 segments et de la LED d'autorisation sera implanté dans l'espace mémoire XDATA et sera forcément accessible à l'adresse 6000H.
- Le port d'entrée permettant au 8051F020 de lire l'état des interrupteurs de codage de l'identifiant du véhicule ainsi que les capteurs de présence de véhicules à l'entrée comme à la sortie du parking sera implanté dans l'espace mémoire XDATA, et sera forcément accessible à l'adresse 4000H.
- Les ports d'entrée et de sortie seront construits à partir de circuits 74HC573
- L'entrée d'interruption INT7 (sur P3.7) est reliée à un bouton poussoir de la carte 8051F020.

Affectation des signaux sur le Périphérique d'entrée REG_IN: espace XDATA – 4000H



Dct Park OUT = 1: voiture détectée
Dct Park IN = 1: voiture détectée

Affectation des signaux sur le Périphérique de sortie REG_OUT: espace XDATA – 6000H



Seg n = 0 : segment allumé
Seg n = 1 : segment éteint
Afficheur Anode Commune

Cmd_LED = 1: LED allumée
Cmd_LED = 0: LED éteinte

6. Conception matérielle – Décodage mémoire

Représenter sous forme de schéma la cartographie (le plan mémoire) de l'espace d'adressage XDATA du 8051F020 en faisant apparaître les limites d'adresses de la zone mémoire XDATA interne, de la zone mémoire réservée pour un développement futur, et des zones mémoires retenues pour le périphérique de sortie REG_OUT (afficheur 7 segment et LED) et le périphérique d'entrée REG_IN(les interrupteurs émulant la saisie du code et les détecteurs). Vous adopterez le décodage le plus simple possible (décodage partiel) en respectant toutefois la contrainte impérative suivante :

- **Le périphérique d'entrée REG_IN sera forcément accessible à l'adresse 4000H.**
- **Le périphérique de sortie REG_OUT sera forcément accessible à l'adresse 6000H.**

Donner, en les simplifiant au maximum, les fonctions logiques de sélection des espaces mémoires correspondants aux 2 périphériques ($CS_{entrée}$, CS_{sortie}). Justifier votre solution.

Adaptez vos fonctions logiques de sélection des espaces mémoire aux signaux de commande des 74HC573, les signaux LE et /OE

- **Démultiplexage adresses / données :**

Vu le décodage partiel adopté, avez-vous besoin dans ce cas de réaliser le démultiplexage adresses basses – données ?

- **Schéma électrique :**

Faites sur papier, un schéma électrique complet de votre dispositif.

Faire apparaître **tous** les signaux nécessaires sans oublier les circuits annexes (résistances, circuits logiques, etc.) nécessaires au fonctionnement du système.

Outre les 74HC573, votre schéma pourra contenir des circuits logiques de référence 74HC00, 74HC02, 74HC04, 74HC08, 74HC32, 74HC86 et 74HC138. Faites-en sorte de consommer le moins de boitiers possibles.

7. Codage de sous-programmes

Pour réaliser ce projet, nous avons identifié plusieurs fonctions logicielles type à coder. Ces fonctions seront mises en œuvre au travers de sous-programmes qui sont listés ci-dessous et que vous devrez coder et tester durant cette séance.
Ces sous-programmes se veulent génériques, c'est à dire que de nombreuses informations leur sont transmises au travers des paramètres passés par le programme appelant (comme par exemple, les valeurs des adresses des registres REG_IN et REG_OUT.

⚠ Tous les sous-programmes à coder dans cette séance, seront utilisés dans l'application finale. Respecter **impérativement** les noms des sous-programmes et les règles de passage de paramètres. Tous ces sous-programmes seront placés dans le fichier **Lib_VosSP_SM4.asm**.

7.1. Définitions :

Paramètre(s) d'entrée : information(s) communiquée(s) par le programme appelant au sous –programme. En général on utilise les registres R0 à R7 pour transmettre ces informations. Avant l'appel du sous-programme, le programme appelant place dans les registres R0 à R7 les valeurs à transmettre. Le sous-programme n'a alors plus qu'à récupérer ces valeurs et à les utiliser.
Valeur retournée : Résultat de calcul communiqué par le sous-programme au programme appelant. En général on utilise les registres R0 à R7 pour transmettre ces résultats. A la fin de l'exécution du sous-programme, le sous-programme place dans les registres R0 à R7 le résultat à renvoyer. Le programme appelant après l'exécution du sous-programme n'a alors plus qu'à récupérer cette valeur et à l'utiliser.

7.2. SP1__Sous-Programme: *_Read_code* (2 point)

Ce sous-programme sera chargé de lire le code d'accès via le périphérique d'entrée.

Paramètre d'entrée : R6 (MSB)- R7 (LSB) – Adresse XDATA du périphérique d'entrée (REG_IN)

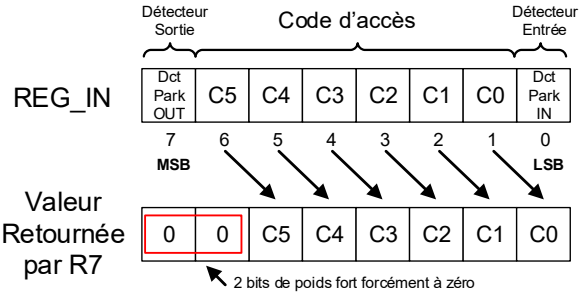
Le programme appelant placera dans R6 et R7 l'adresse du périphérique d'entrée

Valeur retournée : R7 : contient la valeur du code lu (sur les 6 bits de poids faible).

Le sous-programme placera dans R7, la valeur du code lu. Cette valeur pourra ensuite être récupérée par le programme appelant

Registres modifiés autres que ceux utilisés pour le passage de paramètre : aucun

Vu du programme appelant, le sous-programme ne modifie aucun registre, hormis les registres utilisés pour le passage de paramètres.



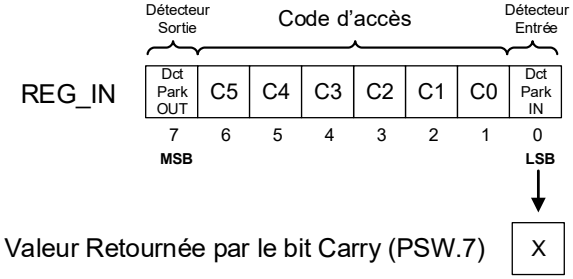
7.3. SP2__Sous-Programme: *_Read_Park_IN* (1 point)

Ce sous-programme sera chargé de lire l'état du détecteur d'entrée dans le parking (DCT_Park_IN).

Paramètre d'entrée : R6 (MSB)- R7 (LSB) – Adresse XDATA du périphérique d'entrée

Valeur retournée : Bit C (Bit Carry) du registre PSW: 0 si absence de détection, 1 si véhicule détecté.

Registres modifiés autres que ceux utilisés pour le passage de paramètre : aucun



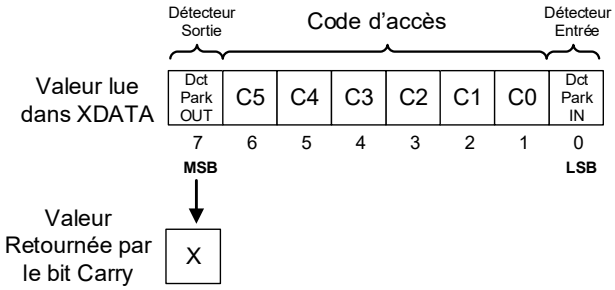
7.4. SP3__Sous-Programme: _Read_Park_OUT (1 point)

Ce sous-programme sera chargé de lire l'état du détecteur de sortie du parking (DCT_Park_OUT).

Paramètre d'entrée : R6 (MSB)- R7 (LSB) – Adresse XDATA du périphérique d'entrée

Valeur retournée : Bit C (Bit Carry) du registre PSW: 0 si absence de détection, 1 si véhicule détecté.

Registres modifiés autres que ceux utilisés pour le passage de paramètre : aucun

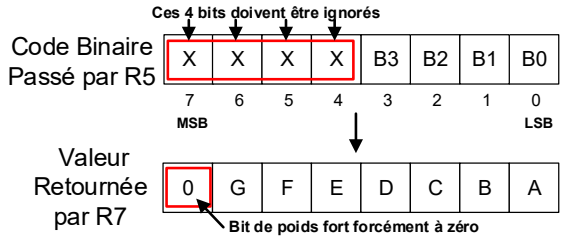


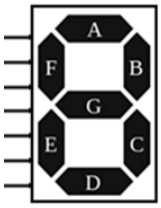
7.5. SP4__Sous-Programme: _*Decod_BIN_to_BCD* (3 points)

Ecrire un sous-programme **_Decod_BIN_to_BCD** permettant de réaliser le transcodage d'une valeur numérique codée sur 4 bits en un code permettant l'affichage de cette valeur sur un afficheur 7 segments. Vérifier le bon fonctionnement en mode pas à pas.

Pour ne pas que vous perdiez du temps, nous vous fournissons une table de décodage contenant les codes 7 segments pour afficher tous les chiffres hexadécimaux de 0 à F (valable pour un afficheur en anode commune, 0 signifie LED allumée)

Display_7S: DB 040h,079h,024h,030h,019h,012h,002h,078h,000h,010h,008h,003h,046h,021h,006h,00Eh





Exemple :
Si la valeur passée par R5 est 04H
Alors la valeur retournée dans R7est le (4+1)ième élément de la table Display_7S, soit la valeur 19H

Paramètre d'entrée : R6 (MSB)- R7 (LSB) – Adresse CODE de la table de conversion Display_7S

Paramètre d'entrée R5 – Valeur 4 bits à convertir (poids faibles) – Les 4 bits de poids fort doivent être ignorés

Valeur retournée : R7 - Code 7 segments (Bit 0-Segment a __ Bit6-Segment g).

Registres modifiés autres que ceux utilisés pour le passage de paramètre : aucun

7.6. SP5__Sous-Programme: _*Display* (3 points)

Ce sous-programme sera chargé d'envoyer les informations d'affichage vers l'afficheur et la LED, via le périphérique de sortie.

Paramètre d'entrée : R6 (MSB)- R7 (LSB) – Adresse XDATA du périphérique de sortie

Paramètre d'entrée : R5 – Code pour afficheur 7 segments (Bit 0-Segment a __ Bit6-Segment g)

Paramètre d'entrée : R3 – Commande de la LED : si R3= 0, LED éteinte, si R3 non nul : LED allumée

Valeur retournée : R7 : contient une recopie de la valeur envoyée au périphérique de sortie.

Registres modifiés autres que ceux utilisés pour le passage de paramètre : aucun

7.7. SP6__Sous-Programme: *_Test_Code* (4 points)

Les codes attribués, ainsi que le nombre de codes valides sont stockés dans la table ***Tab_code*** en mémoire CODE (d'abord le nombre de code, suivi des codes). Ecrire un sous-programme permettant de vérifier que le code d'accès lu sur le périphérique d'entrée se trouve bien dans la table ***Tab_code***. Vérifier le bon fonctionnement en mode pas à pas.
La taille maximale de ***Tab_code*** sera de 32 codes stockés.

Attention, la taille de ***Tab_code*** est donnée par la valeur du premier élément de la table. Cette valeur sera comprise entre 1 et 32 éléments. On considère que la table de constantes ***Tab_code*** respecte le format, c'est-à-dire que les 2 bits de poids forts de chaque élément sont systématiquement à zéro (code sur 6 bits de 1 à 63).

Paramètre d'entrée : R6 (MSB)- R7 (LSB) – Adresse CODE de ***Tab_code***

Paramètre d'entrée : R5 – Code 6 bits à vérifier (code sur 6 bit de pds faible, le 2 bits de pds fort doivent être rendus inopérants, et donc mis à zéro)

Valeur retournée : R7 : non nul, il retourne la position du code trouvé dans la table, nul, il indique que le code n'a pas été trouvé dans la table.

Registres modifiés autres que ceux utilisés pour le passage de paramètre : aucun

7.8. SP7__Sous-Programme: *_Stockage_Code* (6 points)

Chaque entrée de véhicule provoquera le stockage en mémoire XDATA du code autorisé, afin d'avoir un historique des personnes ayant accédé au parking. Ces informations seront stockées dans la table ***Tab_histo***. Cette table pourra au maximum contenir 100 enregistrements. La première case de ce tableau contiendra un index indiquant le nombre d'enregistrements stockés. Ecrire un sous-programme permettant de stocker le code d'accès autorisé dans la table ***Tab_histo*** à la suite des précédents enregistrements, d'incrémenter l'index, et d'avertir en cas de débordement de table.
Les informations à enregistrer seront des informations 8 bits (6 bits de code et 2 bit pour d'éventuelles informations annexes).

Paramètre d'entrée : R6 (MSB)- R7 (LSB) – Adresse XDATA de ***Tab_histo***

Paramètre d'entrée : R5 – Information 8 bits à enregistrer (code sur les 6 bits de poids faible, les 2 bits de pds fort peuvent être utilisé pour des informations annexes)

Valeur retournée : R7 : non nul, il retourne le nombre d'enregistrements, nul, il indique que la table est pleine (100 enregistrements).

Registres modifiés autres que ceux utilisés pour le passage de paramètre : aucun

8. Rendus à produire en fin de séance 4

A la fin de la séance vous devez rendre via le E-campus :

- **Le fichier : Lib_VosSP_SM4.asm. Ce fichier contiendra tous vos sous programmes. Il sera testé sur un banc de test logiciel. Chaque sous-programme sera automatiquement testé. Le nombre de points attribués par sous-programme sera binaire (respecte parfaitement le cahier des charges ou ne respecte pas le cahier des charges).**

D'autre part, vous rendrez sous-forme papier :

- **La feuille de conception matérielle de votre dispositif – Cette fiche contiendra un schéma complet du montage, le plan mémoire et les équations de décodage.**

Précautions à prendre sur le rendu Lib_VosSP_SM4.asm

- **Ce fichier sera testé sur un banc de test logiciel. Chaque fichier rendu sera inséré dans un projet microvision contenant un programme de test. Ce dernier sera chargé d'appeler vos sous-programmes, en faisant varier les paramètres d'entrée et en vérifiant les paramètres de sortie.**
- **Si votre fichier Lib_VosSP_SM4.asm ne s'assemble pas dans notre projet de test, la note globale attribuée sera de 0**
- **Si votre fichier Lib_VosSP_SM4.asm provoque le plantage du banc de test, la note globale attribuée sera de 0.**

Séance 5 : codage et mise au point de l'application globale

9. Objectifs de cette séance 5:

- Etablir un algorigramme de l'application globale en faisant bien ressortir l'enchainement des différentes actions à réaliser aussi bien dans le programme principal que dans les 2 programmes d'interruption.
- Coder l'application globale en vous appuyant sur vos sous-programmes codés lors de la séance précédente. Si vos sous-programmes n'ont pas passé les tests ou que vous n'avez pas eu le temps de tout faire, nous pourrons vous fournir des sous-programmes validés. L'utilisation de ces sous-programmes fournis par nos soins pénalisera votre évaluation.
- Tester et valider votre réalisation en simulation. La validation finale pourra avoir lieu sur une maquette réelle que nous fournirons.

10. Rendus à produire en fin de séance 5

A la fin de la séance vous devez rendre via le E-campus :

- **Le fichier** : **Lib_VosSP_SM4.asm. Ce fichier contiendra tous vos sous-programmes.**
- **Le fichier Main_SM4.asm. Ce fichier contiendra le code de l'application globale.**
- **Un fichier ZIP contenant le dossier projet Microvision complet**

D'autre part, vous rendrez sous-forme papier :

- **La feuille de conception algorithmique – Cette fiche contiendra les algorigrammes de votre programme principal et de vos codes d'interruption Timer3 et INT7.**

Vos codes seront testés sur une maquette matérielle pour valider leur bon fonctionnement.