

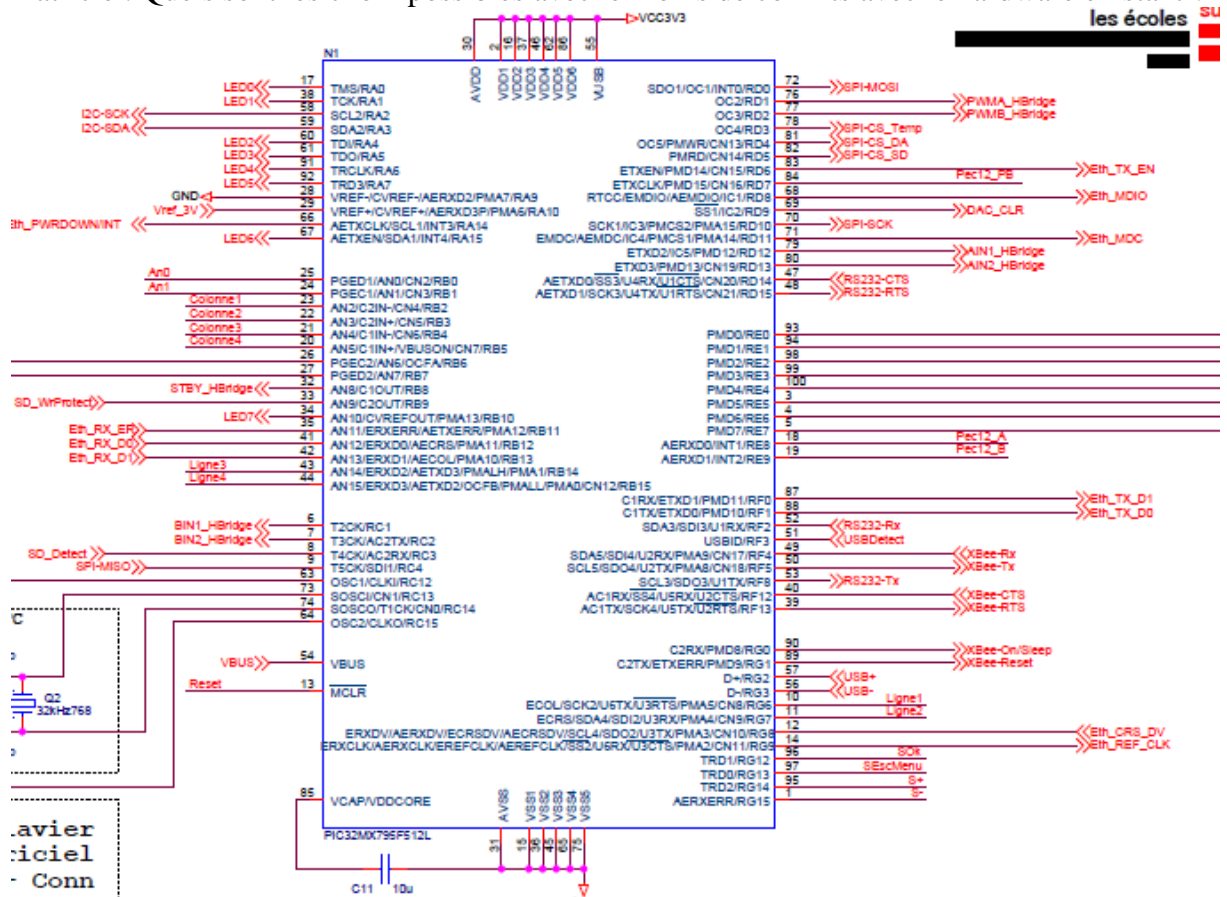
Exercice révision chapitres 2 à 6

Cette révision porte sur les chapitres 2 à 6 du cours "programmation des PIC32MX".

L'ensemble des cours de théorie et de laboratoire sont à disposition.

QUESTION 1

a) Pour effectuer un essai d'un module, on souhaite réaliser un bus de 4 bits (les 4 lignes sur le même port ainsi que des lignes contiguës). Le port E est entièrement utilisé par l'affichage LCD. On souhaite conserver le plus de fonctionnalités du kit (ports série pour RS232, SPI, I2C, sortie OC et input capture. Par contre, il est possible de sacrifier les leds et le clavier matriciel. Quels sont les choix possibles avec le moins de conflits avec le hardware existant ?



b) Il est demandé, en utilisant directement les registres TRIS, PORT et LAT, de réaliser les actions de lecture et d'écriture sans entrer en conflit ni modifier le comportement des autres lignes du port. Il est en plus demandé d'effectuer les actions d'écriture et de lecture globalement sur l'entier du port.

```
void main() {

uint16_t ValBusIn;
uint16_t ValBusOut = 0x0A;

// Configuration et écriture sur le port de ValBusOut
.....

.....

.....

.....

.....

.....

.....

// Configuration et Lecture du bus
// (On souhaite les 4 bits du bus dans le poids
// faible de ValBusIn)
.....

.....

.....

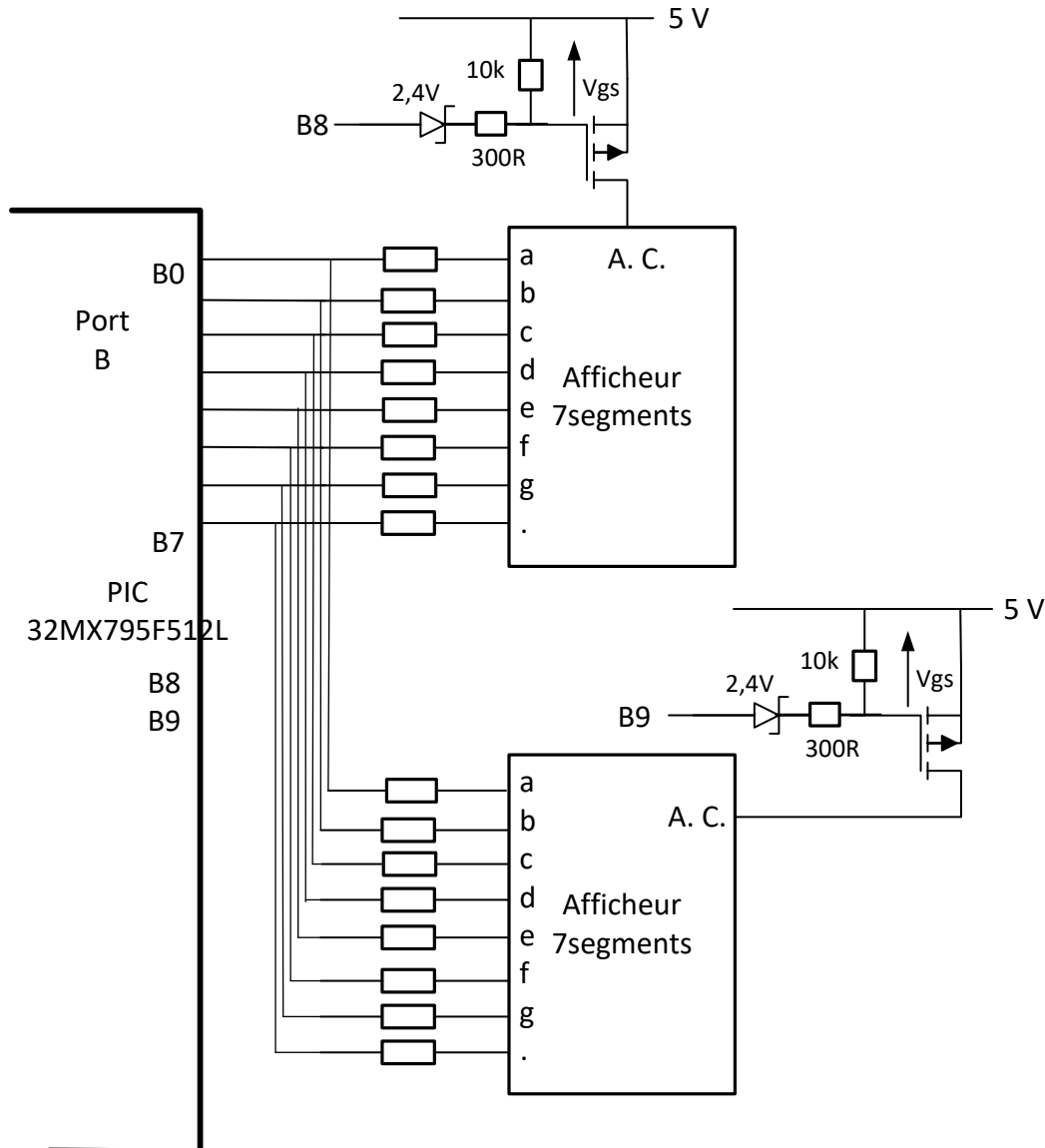
.....

.....

ValBusIn = .....
}
```

QUESTION 2

Le PIC est alimenté en 3V3, les afficheurs en 5V. Voici le câblage de 2 afficheurs 7seg. Un seul des afficheurs doit être alimenté. Donnez l'action sur la ligne B8 pour faire conduire le MOS ($V_{gs} \leq V_{gs,threshold}$) et l'action pour le bloquer ($V_{gs} = 0\text{ V}$). Utilisation des fonctions bit de plib_ports de Harmony. On suppose que B8 et B9 sont configurés en sortie.



```
// Action sur B8 pour  $V_{gs} = -5\text{ V}$ 
```

```
.....
.....
.....
.....
```

```
// Action sur B8 pour  $V_g = 0\text{ V}$ 
```

```
.....
.....
.....
.....
```

QUESTION 3

Soit l'élément suivant de programme en C :

```
int32_t val;
```

On suppose que la variable val se situe à l'offset 4 par rapport à S8 (copie du SP)

```
val = 0;  
val += 0x31;
```

Donnez l'équivalent en assembleur PIC32MX, en supposant que l'on commence à l'adresse 9D000014.

Adresse	Instruction en assembleur	Commentaire
9D000014		val = 0;

QUESTION 4

Pour les 4 instructions PIC32MX ci-dessous :

a) Donnez le code machine hexadécimal de l'instruction dans la mémoire programme, puis décrivez le mouvement des données pour chaque instruction en donnant la situation après l'exécution (mémoire DATA ou registre).

b) Donnez leur équivalent en langage C.

Start: **ADDIU V0, ZERO, 51**
 SW V0, 8(S8)
 LW V0, 8(S8)
 SW V0, 12(S8)

On suppose que l'étiquette Start correspond à 9D000024 et 8(S8) correspond à la variable val1 ainsi que 12(S8) correspond à la variable val2.

Sous K:\ES\PROJETS\SLO\1102x_SK32MX775F512L\Data_sheets\PIC32 Family Reference Manual le fichier mips32v2_InstructionSet.pdf fournit la possibilité de reconstituer le code machine.

Le registre V0 est codé 2 et le registre ZERO est codé 0.

Le registre de base S8=Fp se code 30.

a)

31	26	25	21	20	16	15	0
ADDIU		rs		rt		immediate	
001001							
6		5		5		16	

Format: ADDIU rt, rs, immediate

ADDIU V0, ZERO, 51

Adresse Mémoire programme

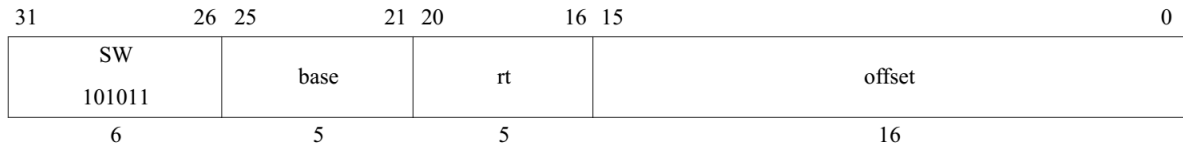
S8 →

Mémoire donnée

0
4
8
12

V0

--

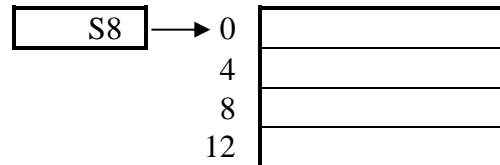


Format: SW rt, offset (base)

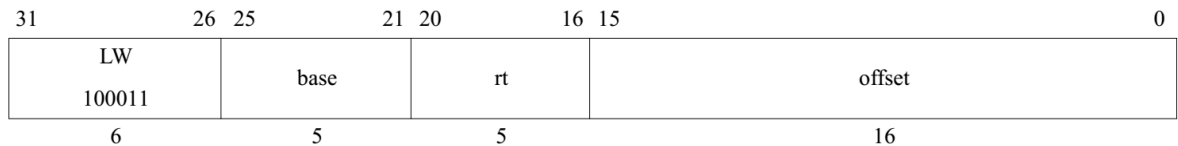
MIPS32 (MIPS I)

SW V0, 8(S8)

Adresse Mémoire programme



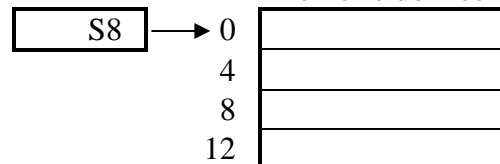
V0



Format: LW rt, offset (base)

LW V0, 8(S8)

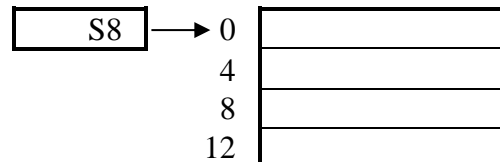
Adresse Mémoire programme



V0

SW V0, 12(S8)

Adresse Mémoire programme



V0

b) Équivalent en C

```
int32_t val1;  
int32_t val2;
```

.....

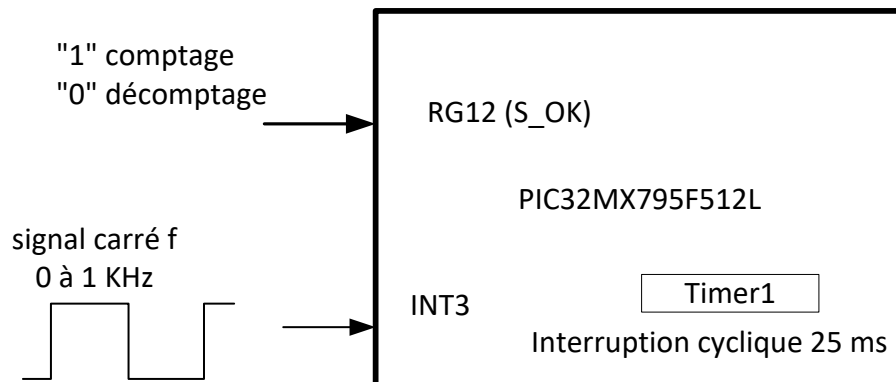
.....

.....

.....

QUESTION 5

Voici le schéma de principe du système demandé :



Dans la réponse à l'interruption du Timer1, on inverse la LED0 pour permettre le contrôle de la période et on établit le statut de l'application à SERVICE_TASKS en utilisant la fonction APP_UpdateState.

Dans la réponse à l'interruption externe 3, on inverse la LED1 à chaque flanc montant pour contrôle. L'action principale consiste à incrémenter (si S_OK = 1) ou à décrémenter (si S_OK = 0) une variable globale 32 bits signée que l'on nommera countInt3.

Remarque : la variable countInt3 doit être déclarée dans l'application. Pour incrémenter il faut appeler la fonction APP_IncCountInt3 et pour décrémenter il faut appeler la fonction APP_DecCountInt3.

Niveau avancé (facultatif) :

Au lieu de prendre l'état de S_OK pour commander le comptage/décomptage, faire en sorte que chaque appui sur S_OK commute entre un mode d'incréméntation et autre mode de décréméntation.

Vous devrez alors réaliser un anti-rebond, ou vous aider de la librairie mc32debounce à disposition. Une variable booléenne indiquant si on est en mode d'incréméntation ou de décréméntation sera toggée à chaque appui.

QUESTIONS THÉORIQUES

a) A quelles broches du PIC32MX795F512L (noms complet et No) correspondent les 4 éléments suivants :

LED_0 :

LED_1 :

INT3 :

LED_6 :

b) Pour obtenir une interruption cyclique de 25 ms avec le timer1, quelle doit être sa période ainsi que le prescaler ?

Période du timer1 pour 25 ms :

Prescaler du timer1 :

.....

.....

.....

.....

c) Faut-il faire des modifications dans la configuration des E/S pour pouvoir fournir un signal sur INT3 ? Si oui nature de la modification, si non preuve.

.....

.....

.....

.....


RÉALISATION PRATIQUE

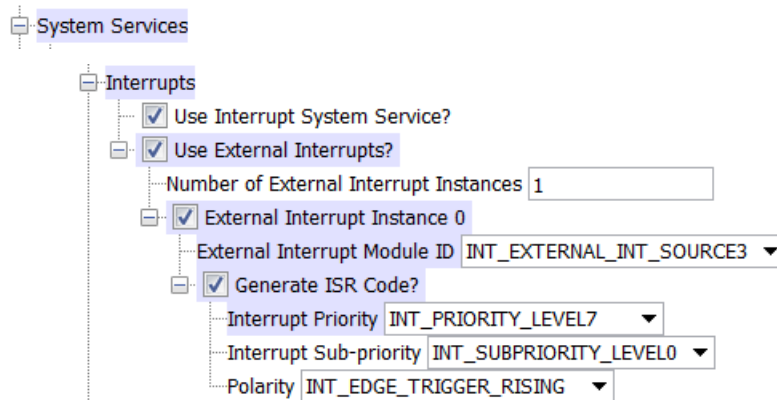
Création d'un projet avec Harmony pour le kitPIC32MX

Utilisation du Timer1 avec interruption, prendre niveau 3.

Utilisation du bsp pic32mx_skes.

Ajout par le MHC de la configuration de l'interruption externe 3, prendre niveau 7. Configurer action au flanc montant.

Se réalise dans  Harmony Framework Configuration



MODIFICATION FICHIER SYSTEM_INIT.C

☞ Il est demandé de transformer le code généré dans system_init.c, en le plaçant dans une fonction à nommer **IntExt3_Initialize**.

MODIFICATION FICHIER SYSTEM_INTERRUPT.C

Compléter l'ISR du Timer1 avec l'inversion de la LED0. Utilisez la fonction BSP_LEDToggle. Ajouter l'appel de la fonction APP_UpdateState.

Compléter l'ISR de INT3 avec l'inversion de la LED1. Utiliser la fonction BSP_LEDToggle. Utilisez IPL7SRS.

Ajouter le comptage/décomptage de la variable countInt3.

MODIFICATION FICHIER APP.C

- Effectuer l'initialisation de l'afficheur LCD et afficher (section case INIT) :

```
Ex Rev chap 2 a 6  
<Nom>
```

- Dans la section case SERVICE_TASKS :
Affichez la valeur de la variable countInt3 sur la ligne 3.

Il faut aussi réaliser les 2 fonctions APP_IncCountInt3 et APP_DecCountInt3, ainsi que la classique fonction APP_UpdateState.

TEST DE FONCTIONNEMENT

Câbler un générateur sur l'entrée int3 en veillant à disposer d'un signal 0 à 3V3.

OBSERVATION

Mesurez à l'oscilloscope la période du Timer1.

Vérifiez que le signal sur Led1 soit de la moitié de la fréquence de celui fourni sur Int3.

En fournissant un signal de 100 Hz vérifiez l'évolution de l'affichage du compteur. En pressant sur S_OK vérifiez que le compteur diminue.