

UTILISATION du DAC du micro LPC23xx : générer des sons....

On va générer les sons en utilisant un son échantillonné stocké dans un tableau que l'on fabrique à l'aide d'un fichier EXCEL

première colonne : numéro

seconde colonne : calcul de la réponse de la fonction $A * \sin(\text{numero}/\text{nb_elements_tableau} * 2 * \text{PI}) + \text{OFFSET_511}$

troisième colonne : caractères `,\`

on fera autant de lignes qu'il y a de cases dans notre tableau depuis `numero = 0` jusqu'à `numéro = nb_elements_tableau - 1`

par un simple copié collé dans notre code C, on va récupérer des lignes :

```
511 ,\
612 ,\
....
1022 ,\
1023 ,\
1022 ,\
...
1 ,\
0 ,\
1 ,\
....
504 ,\
```

il suffira de rajouter à la main une première ligne : `unsigned int tab_son[Nb_elements_tableau] = { \`

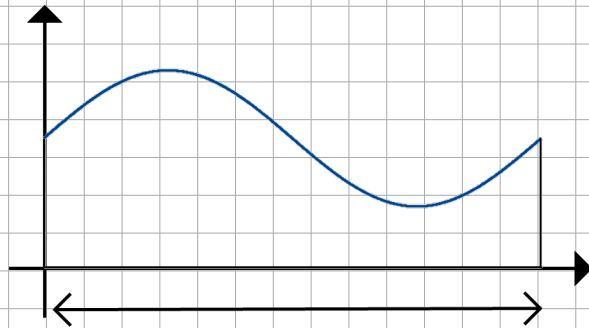
et une dernière ligne : `};`

pour avoir un buffer contenant nos échantillons de son

Regarder le chapitre 28, utiliser le DAC est très simple, il suffit de configurer la patte AOUT et ensuite d'écrire dans le registre DACR au bon format....

dans les codes des pages suivantes, on utilise symboliquement : `sortir_sur_le_DAC()` pour la mise à jour correcte du registre DACR

générer du son pour prévenir d'un obstacle pour le haut parleur du robot



**1ms de son à 1KHz
découpé en 36 échantillons**

quelle fréquence d'échantillonnage ?

sur le robot , on a un 36KHz qui est disponible à cause de l'IR et des moteurs

pour le robot, on doit emettre du 1KHz : le sinus est décrit en 36 échantillons

le DAC travaille en 1024 pas , centré autour de la valeur 512

avec Excell, on fabrique un tableau qui donne les 36 valeurs

```
unsigned int son1K[36]={511,611,707,795,872,936,983,1012,1022,1012,983,936,872,795,707,611,\
511,411,315,227,150,86,39,10,0,10,39,86,150,227,315,411,511,611,707,795};
```

dans l'IT PWM: on a une variable index_son

//son_a_jouer mis à 1 ou 0 dans module TELEM

```
if(son_a_jouer){sortir_sur_le_DAC(son1K[index_son]);} else{sortir_sur_le_DAC(511);}
```

```
index_son++;
```

```
if (index_son>35) {index_son=0;
```

```
// on vient ici toutes les ms, on actualise la décision de continuer à jouer ou pas
```

```
// on avance dans la machine à état de jeu du son en décrémentant nb_cycles_en_cours
```

```
// 2 états : jouer son, ou jouer silence
```

```
// 2 actions au franchissement : recharger nb_cycles_en_cours
```

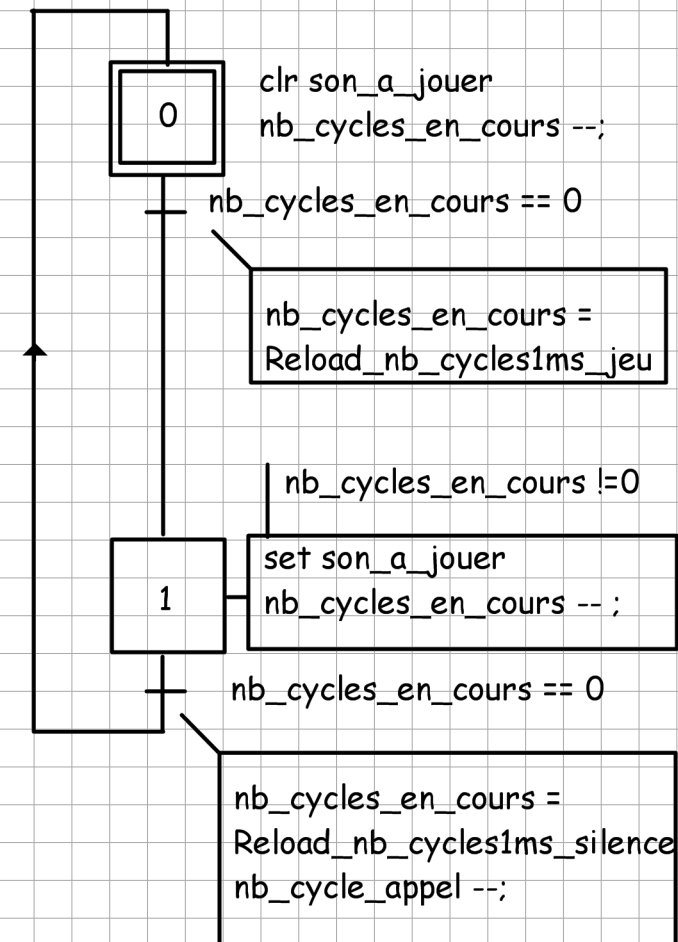
```
// et mise à jour de son_a_jouer
```

```
// avec deux variables globales qui décrivent la séquence son :
```

```
// Reload_nb_cycles1ms_silence mis à jour dans module TELEM
```

```
// Reload_nb_cycles1ms_jeu mis à jour dans module TELEM
```

```
}
```



Pour la base, il faut emettre la somme de deux sinusoides de fréquences différentes :

frequence basse = 697 , 770, 852, 941

frequence haute : 1209 , 1336, 1477, 1633

Low-Fre q. (Hz)	High-Fre q. (Hz)	Code
697	1209	1
697	1336	2
697	1477	3
770	1209	4
770	1336	5
770	1447	6
852	1209	7
852	1336	8
852	1447	9
941	1336	0
941	1209	*
941	1447	#
697	1633	A
770	1633	B
852	1633	C
941	1633	D

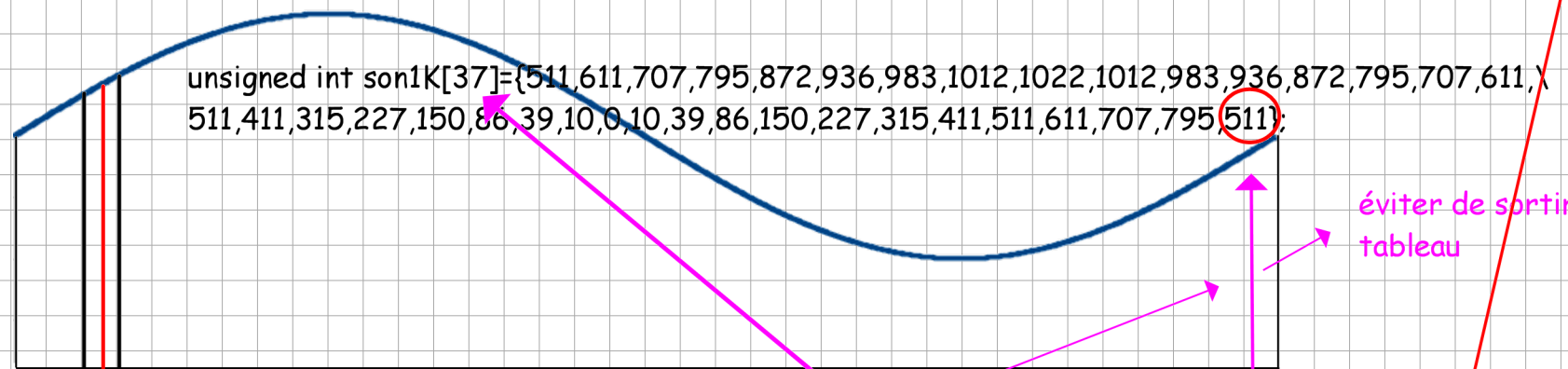
On dispose d'une sinusoide à 1KHZen 36 échantillons

On va changer la vitesse de lecture du tableau : on devrait travailler avec des index en float...
si on veut du 1KHz , on incrémente l'index de 1 à 1

si on veut une fréquence de 1209 HZ, on va incrémenter l'index de 1,209 en 1,209

si on veut une fréquence de 697 Hz, on va incrémenter l'index de 0.697 en 0.697

Le processeur est pas performant sur les calculs en nombres flottants: on va ruser...



l'échantillon 2.5 est la moyenne pondérée entre l'échantillon 2 et l'échantillon 3

on va utiliser un index avec un entier 16 bits dont les 8 bits de poids fort vont être le véritable index du tableau son1K, et les 8 bits de poids faible, la partie derriere la virgule... On augmente plus de 0,697 mais de $0,697 * 256 = 178$, non plus de 1,209 mais de 310
si index_a_virgule dépasse $36 * 256$, on lui soustrait $36 * 256$;

on récupère un index 16 bits...nommé index_a_virgule, on isole le vrai index : $\text{index} = \text{index_a_virgule} \gg 8$;

on isole la partie derriere la virgule : $\text{virgule} = \text{index_a_virgule} \& 0xFF$;

on doit calculer l'échantillon : on dispose de l'échantillon avant $\text{ech_avant} = \text{son1K}[\text{index}]$; $\text{ech_apres} = \text{son1K}[\text{index}+1]$;

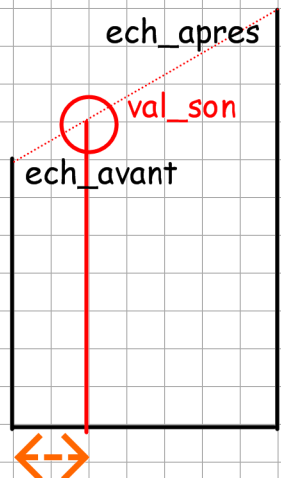
$\text{val_son1} = (\text{ech_avant1} * (256 - \text{virgule1}) + \text{ech_apres1} * \text{virgule1}) \gg 8$;

$\text{val_son2} = (\text{ech_avant2} * (256 - \text{virgule2}) + \text{ech_apres2} * \text{virgule2}) \gg 8$;

$\text{val_son} = (\text{val_son1} + \text{val_son2}) \gg 1$;

$\text{sortir_sur_le_DAC}(\text{val_son})$;

on rajoute un élément de tableau pour ne pas avoir de soucis avec le dernier + 1



technique de programmation pour emettre le signal du poste en DTMF

On se donne la possibilité de jouer deux sons avec des index à virgule que l'on va sommer

tous les 3600 interruptions à 36KHz, (je suis paresseux, j'ai repris le code d'it PWM du robot dans le poste) il s'est écoulé 100ms :

On décide si on joue encore du son pour 100ms ou pas ...

si on ne joue pas de son on va incrementer les deux variables index_a_virgule1 et index_a_virgule2 de la valeur 0 et emettre un signal inaudible

si on joue du son, chacune des variables est incrémentée d'un incrément qui dépend de la fréquence à jouer....

on veut jouer * x y # pour des durees : (* 200ms) (blanc 100ms) (x 200ms) (blanc 100ms) (y 200 ms) (blanc 100ms) (# 200ms) (blanc 100ms)

On va stocker dans des tableaux les valeurs d'increment qui permettent de jouer les bonnes fréquences, un silence se fera par incrément nul !!!

donc on changera l'index de ce tableau toutes les 100ms (détection d'un décompte 3600 -> 0), arrivé au bout, on reste sur la dernière case.


L'arrivée d'un robot devant le poste de travail va autoriser les index à refaire tout le parcours du tableau

seq_freq_haute[12] = {1.209*256, 1.209*256, 0, pas_x_h, pas_x_h, 0, pas_y_h, pas_y_h, 0, 1.477*256, 1.477*256, 0}

seq_freq_basse[12] = {0.941*256, 0.941*256, 0, pas_x_b, pas_x_b, 0, pas_y_b, pas_y_b, 0, 0.941*256, 0.941*256, 0}

~~~~~  
tableaux à initialiser quand on lit le numéro de poste au démarrage du processeur, les charger dans l'autre sens si on fait du décomptage...

Astuces d'amélioration :

-- plutôt que de compter, on peut décompter et rester bloqué à zéro (on  met un signal continu qui n'est pas audible puisque l'index ne s'incrémente pas

-- quand un robot passe devant le poste (on le sait car on reçoit son code IR), pour répondre il suffit de déplacer l'index et de le mettre à 11, ainsi que le décompte 100ms... le son va se jouer automatiquement sans qu'on ait à se préoccuper de rien, et on continue à jouer du silence ensuite... La charge CPU est quasi constante.

-- Pour minimiser les calculs et éviter des décalages inutiles, il suffit de penser que sortir un échantillon de son revient à écrire sur les bit B6 à B15 du registre DACR, on peut se débrouiller pour que l'alignement soit directement bon. Au lieu de faire un décalage >>8 pour les deux signaux, puis une moyenne entre le signal fréquence haute et le second signal fréquence basse du DTMF, puis un décalage <<6 et effacer par un masque les bits de B0 à B5 et B16, il y a beaucoup plus efficace :

On ajoute les signaux non décalés (vérifier que cela ne va pas déborder), puis on décale directement de ce qu'il faut et on fait le masque...

Bien sûr il faut commenter l'astuce pour favoriser la relecture.

-- Si on utilise les concept d'union, on a plus besoin de faire un décalage de 8 pour récupérer l'octet de poids fort d'un entier 16bits, cela peut être pratique pour manipuler les index à virgule

| Low-Freq. (Hz) | High-Freq. (Hz) | Code |
|----------------|-----------------|------|
| 697            | 1209            | 1    |
| 697            | 1336            | 2    |
| 697            | 1477            | 3    |
| 770            | 1209            | 4    |
| 770            | 1336            | 5    |
| 770            | 1447            | 6    |
| 852            | 1209            | 7    |
| 852            | 1336            | 8    |
| 852            | 1447            | 9    |
| 941            | 1336            | 0    |
| 941            | 1209            | *    |
| 941            | 1447            | #    |
| 697            | 1633            | A    |
| 770            | 1633            | B    |
| 852            | 1633            | C    |
| 941            | 1633            | D    |