

# TP 1- VIRGULE FIXE - VIRGULE FLOTTANTE

## CODAGE EN VIRGULE FLOTTANTE LOGICIELLE SUR 32 BITS

1. Faites la déclaration des variables suivantes :

```
float fTab1[8]={0.25, 0.75, 1.25, 3.75, 2.23, 4.56, 3.41, 2};  
float fTab2[8]={3.25, 2.75, 1.333, 4.715, 6.53, 1.58, 2.42, 3.3};  
float fs = 0;
```

2. Dans la boucle infinie tapez le code suivant et ajouter deux points d'arrêts (breakpoint), l'une au début de la boucle et l'autre au niveau de la dernière instruction :

```
/* **** */  
fs = 0;  
● for(Count = 0; Count <8; Count++)  
    {  
        fs = fs + fTab1[Count]*fTab2[Count];  
    }  
● fs = fs ;  
/* **** */
```

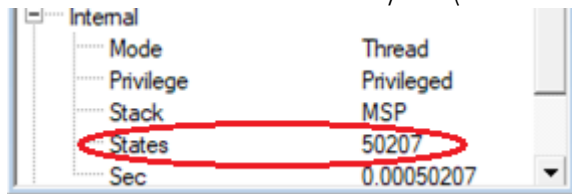
3. Compilez votre programme (F7).

4. Passez en mode débogage (Ctrl + F5).

5. Ajoutez dans la fenêtre **Watch Window** la variable **fs**.

6. Lancez l'exécution du programme jusqu'au premier point d'arrêt en cliquant sur le bouton **Run** ou la touche (F5).

7. Notez la valeur du nombre de cycles (variable **states** dans la fenêtre **Registers**).



8. Relancez l'exécution du programme jusqu'au dernier point d'arrêt.

9. Notez à nouveau la valeur du nombre de cycles (variable **states** dans la fenêtre **Registers**).

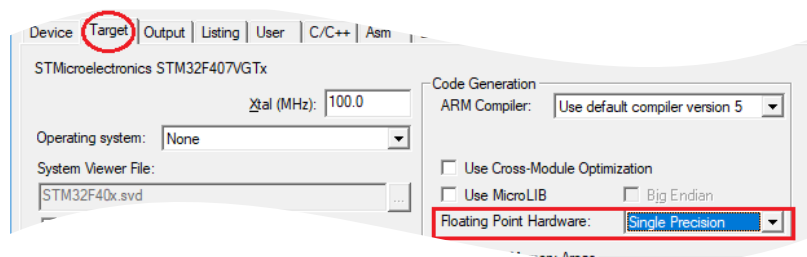
10. Déterminez alors le nombre de cycles de traitements de la boucle et notez la valeur de **fs**.

## CODAGE EN VIRGULE FLOTTANTE MATERIELLE SUR 32 BITS

1. Cliquez sur le bouton **Options for Target...**



2. Dans la fenêtre qui s'ouvre, choisissez **Single Precision** pour Floating Point Hardware ; puis cliquez sur OK.



3. Refaire le même travail que précédemment et comparer les temps de traitement en virgule fixe et en virgule flottante hardware.

## GODAGE EN VIRGULE FIXE Q4.12

1. Le calcul en virgule fixe revient à travailler sur des entiers. Faites la conversion des éléments des tableaux précédents en entiers out en respectant le format Q4.12 (1 bit de signe, 3bits pour la partie entière et 12 bits pour la partie fractionnaire).
2. Complétez dans la zone de déclaration les tableaux suivants par les valeurs déjà calculées.

```
/* **** */
int16_t vxTab1[8] = {..., ..., ..., ..., ..., ..., ..., ...};
int16_t vxTab2[8] = {..., ..., ..., ..., ..., ..., ..., ...};
int32_t vxs = 0;
/* **** */
```

3. Déclarez aussi les unions suivantes :

```
/* **** */
union {
    int16_t *vxh1;
    int32_t *vxw1;
}vx1;
union {
    int16_t *vxh2;
    int32_t *vxw2;
}vx2;
/* **** */
```

4. Tapez avant la boucle infinie, les instructions suivantes :

```
/* **** */
vx1.vxh1 = vxTab1 ;
vx2.vxh2 = vxTab2 ;
/* **** */
```

5. Tapez dans la boucle infinie le code suivant :

```
/* **** */
● for(Count = 0; Count <8; Count++)
{
    vxs = vx1.vxh1[Count]*vx2.vxh2[Count]+vxs;
}
● vxs = vxs ;
/* **** */
```

6. Compilez et refaire le même travail que précédemment. Que peut-on dire de la résolution et du temps de traitement ?

## UTILISATION DE L'INSTRUCTION SIMD (SMLAD)

1. Remplacez le code dans la boucle infinie par le code suivante :

```
/* **** */
● for(Count = 0; Count <4; Count++) // attention count<4
{
    vxs = __SMLAD(vx1.vxw1[Count], vx2.vxw2[Count], vxs);
}
● vxs = vxs ;
/* **** */
```

2. Que peut-on dire du temps de traitement ?

## GENERATION D'UNE SINUSOÏDE

On peut générer une onde sinusoïdale de deux manières :

- Look-up table : Cette méthode n'est pas utilisée pour des applications de haute précision. Si on cherche la haute précision, la mémoire doit être grande.
- Développement de série de Taylor : C'est une méthode plus efficace comparé à la méthode look-up table ; cette méthode ne consomme pas de la mémoire.

Développement en série de Taylor :

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} = x \left( 1 - \frac{x^2}{6} \left( 1 - \frac{x^2}{20} \left( 1 - \frac{x^2}{42} \left( 1 - \frac{x^2}{72} \right) \right) \right) \right) ; -\pi \leq x \leq \pi$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{8}{8!} = 1 - \frac{x^2}{2} \left( 1 - \frac{x^2}{12} \left( 1 - \frac{x^2}{30} \left( 1 - \frac{x^2}{56} \right) \right) \right) ; -\pi \leq x \leq \pi$$

1. Soit  $\pi = 3.14159265358979$  ; et  $r = 2*\pi/20$  . Tapez dans la boucle infinie le code suivant :

```
/******  
if(TeFlag == 1)  
{  
    TeFlag = 0;  
    x = Count*r - pi;  
    xx = x*x;  
    x0 = 1 - xx/72;  
    x1 = 1 - x0*xx/42;  
    x2 = 1 - x1*xx/20;  
    x3 = 1 - x2*xx/6;  
    y = x*x3;  
    Count++;  
    if(Count >= 20) Count = 0;  
}  
/******
```

1. Programmez la fonction sinus et affichez sa courbe dans la fenêtre de l'analyseur logique.

