Informatique embarquée 5/16

J.-M Friedt

FEMTO-ST/département temps-fréquence

jmfriedt@femto-st.fr

transparents à `jmfriedt.free.fr`

5 février 2018

Virgule fixe/virgule flottante

- Représentation d'un nombre en virgule flottante :
  - mantisse/exposant (mantisse $\in [0, 1-1[$), similaire à la notation scientifique $0,1234 \times 10^3$
  - supporte une large gamme de valeurs, au détriment de la précision (codage de la position de la virgule)
  - exposant 8 bits-mantisse 23 bits (float), 11-52 (double) qui code donc $2^{\pm 1024} \simeq 10^{\pm 308}$ sur 15 décimales ($2^{52} \simeq 4 \times 10^{15}$)
  - $\Rightarrow$ arithmétique complexe, addition passe par la dénormalisation pour avoir le même exposant (donc des 0 en début de mantisse du nombre le plus petit), multiplication aisée
  - implémentation matérielle sur processeurs haut de gamme (FPU)
- Représentation en virgule fixe : homothétie pour se ramener à des calculs sur des entiers.
- on note Qm.n pour représenter la partie entière sur m bits et la partie fractionnaire sur n bits [1] (notation en complément à deux, donc inclut un bit de signe)

1. Maxim, Application Note 3722 : Developing FFT Applications with Low-Power Microcontrollers (2006)

Opérations sur les flottants

- flottant = mantisse $\cdot 2^{exposant}$
- $\Rightarrow$ multiplication somme les exposants et multiplie les mantisses (problème de précision)
- $\Rightarrow$ sommer nécessite d'aligner les mantisses en ajustant les exposants
- en l'absence de FPU, opérations logicielles gourmandes en ressources

Exemple sur STM32F1 (pas de FPU)[2] :

Opération	durée (μs)
ADC $\rightarrow$ entier	1,73
ADC $\rightarrow$ flottant	3,36
entier $\times 10^6/10^6$	1,08
entier $(<< 20)(>> 20)$	0,83
IIR à 2 coefficients flottants	30
IIR à 2 coefficients entiers	3

2. données acquises par G. Matten, LMA, FEMTO-ST

Cas de la fonction affine

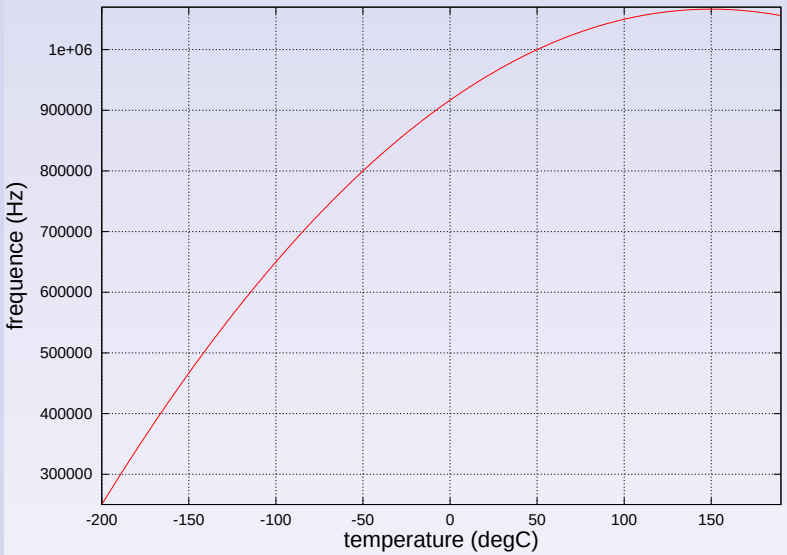
1. Un microcontrôleur manipule une donnée dans sa représentation : une fréquence de DDS est comprise entre 0 et f_{CK}, représentée par $[0..2^N - 1]$,
2. un humain veut connaître une fréqeunce en Hz

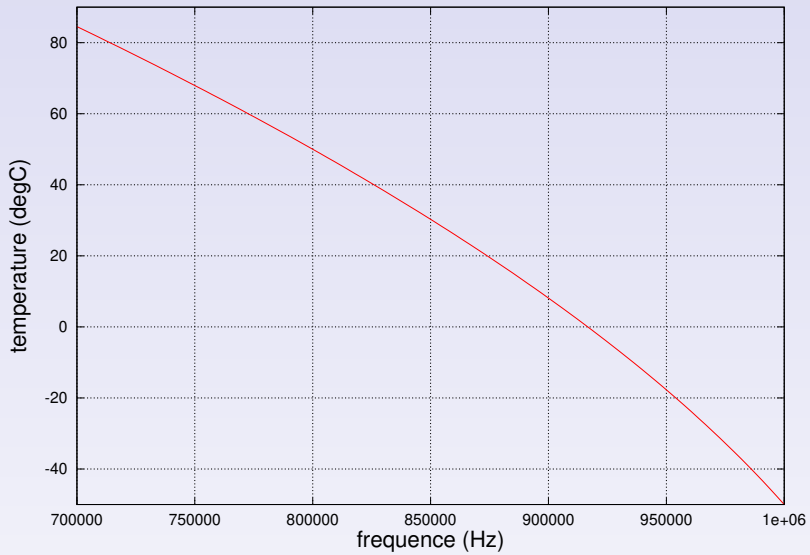
$$f_{Hz} = \frac{mot}{2^N} \cdot f_{CK}$$

3. application numérique, $f_{CK} = 70$ MHz et $N = 28$
$\Rightarrow 70 \cdot 10^6 / 2^{28} = 0.260770320892334$
4. GNU/Octave (Matlab) : `rats(0.260770320892334,5)=6/23` et `rats(0.260770320892334,8)=914/3505`
5. $mot \in [0..2^{28}] \Rightarrow mot \times 914$ sur 38 bits
6. mais si la bande de fréquence est connue (e.g. 8-10 MHz), bits de poids fort peuvent être éliminés
`(mot&0x003fffff)*914/3505` sera exacte, tandis que 0x00400000*914/3505 est précalculé (bits de poids forts connus)
7. si la fréquence est imprécise, bits de poids faible peuvent être éliminés : `(mot>>8)*914/3505` perd les 100 derniers Hz

Exemple de capteur

- soit une mesure de fréquence f issue d'un capteur oscillant, dont la fréquence de vibration change avec la température T
- sachant que $f \simeq 1$ MHz avec une précision de 10 Hz, on veut obtenir la température à 0,1 K près :





$$f = -6,6667 \times T^2 + 2000 \times T + 916670$$

$$T = -150 + \sqrt{160000 - 0,15 \times f}$$

Exprimer cette loi d'étalonnage avec des coefficients entiers

Rappel : la suite $x_{n+1} = \frac{1}{2}\left(x_n + \frac{y}{x_n}\right)$ converge vers $\sqrt{y}$

(méthode de Newton sur $x^2 - y = 0$ avec $x_{n+1} = x_n - f/f'$)

Application des coefficients d'étalonnage

$$\boxed{T = A + \sqrt{B + C \cdot f}}$$

Hypothèses :

- $f \simeq 10^6$ (1 MHz)
- f à 10 Hz près
- $A \simeq 100$
- $C \simeq 0,1$
- T à 0,1 K près

Application des coefficients d'étalonnage

Exemple de calcul en virgule fixe compte tenu de la précision recherchée :

1. la loi liant fréquence et température est du type $f = \alpha T^2 + \beta T + \gamma \Rightarrow T = A \pm \sqrt{B + Cf}$
2. alors $(T-A)^2 = B + C \times f$ donc $2(T-A)dT - 2(T-A)dA = dB + dC \times f \Rightarrow |dT| = \left|\frac{dB}{2(T-A)}\right| + \left|\frac{dC \times f}{2(T-A)}\right| + |dA|$
3. on doit donc travailler avec chacun de ces termes inférieurs à 0,1.
4. Application numérique : par conception, $T - A \geq 100$ et $f \leq 10^6$ donc $dB \leq 10$, $dA \leq 0,1$ et $dC \leq 10^{-5}$
5. on va donc travailler sur $A \times 10$ et $C \times 10^5$ pour respecter la résolution recherchée
6. cependant, $C \simeq 0,1$ donc $C \times 10^5 \times f \simeq 10^{10}$ qui nécessite 34 bits,
7. mais f à 10 Hz donc on peut travailler sur $C \times 10^5 \times (f/10) \simeq 10^9$ qui ne nécessite que 30 bits

Conclusion : $10 \cdot A$, $10^4 \cdot B$ et $10^5 \cdot C$ pré-calculés

$$
\begin{aligned}
10 \times T = \quad & 10 \times A + 10\sqrt{B + C \cdot f} \\
& 10 \times A + 10\sqrt{B \cdot 10^4 + C \cdot f \cdot 10^4}/100 \\
& 10 \times A + \sqrt{B \times 10^4 + (10^5 C)\,(f/10)}/10
\end{aligned}
$$

Mise en pratique

Nous quittons l'AVR qui a choisi de développer sa propre libc, pour passer au STM32

- processeur 32 bits capable d'embarquer des bibliothèques (puis environnements exécutifs)
- `summon-arm-toolchain` fournit un script pour générer un ensemble "cohérent" d'outils – explication de sa compilation à `http://jmfriedt.free.fr/summon_arm.pdf`
- le processeur se décline en une multitude de classes : *linker* script approprié.
- Émulateur pour STM32 : `qemu` pour ARM complété de l'émulation de périphériques – `https://github.com/beckus/qemu_stm32`.

Les divers outils de compilation : `Makefile`, `configure`, `cmake`
Les divers outils d'archivage : `svn`, `git`, `hg` (mercurial)