Électronique numérique

J.-M Friedt

FEMTO-ST/département temps-fréquence

jmfriedt@femto-st.fr

transparents à `jmfriedt.free.fr`

22 novembre 2020

Plan des interventions :

7 cours/TP d'introduction au STM32 en *C baremetal* :

1. Électronique numérique et conception du circuit
   L3 : aspects analogiques, consommation électrique, lecture de datasheet
   Survol des divers périphériques qui seront abordés (RS232, SPI, timer, ADC)
   représentation des données (tailles/encodage), masques, architecture
   Rappels sur Atmega32U4 (Makefile, compilation, masques ...)
2. Premiers pas sur le STM32, adresses des périphériques, architecture
3. Fonctionnement de gcc et optimisations :
   préprocess–compilateur–assembleur–linker, passage C à assembleur, pointeurs
4. bibliothèques et séparation algorithme/matériel, simulateurs
   libopencm3, newlib & stubs, ressources requises par les bibliothèques
5. Bus de communication série, synchrone/asynchrone
6. arithmétique sur systèmes embarqués
   entiers, flottants, convertir un algorithme exprimé en nombres à virgules vers des entiers, timers
7. interruptions et acquisition de données analogiques, fréquence d'échantillonnage
   vecteurs d'interruption, gestion des horloges du STM32, ADC

Optimisations du code C

Taille des variables [1]

- `(unsigned) char` : 8 bits
- `(unsigned) short` : 16 bits
- `(unsigned) long` : 32 bits
- `(unsigned long long)` : 64 bits
- virgule flottante simple précision `float` et double précision `double`

Utiliser le type présentant le **meilleur compromis espace/précision** des données (analyse rationnelle)

Types dérivés explicitent la taille et la nature de la variable

`/usr/include/stdint.h` :

```
/* ISO C99: 7.18 Integer types <stdint.h> */
typedef signed char             int8_t;
typedef short int               int16_t;
typedef int                     int32_t;
# if __WORDSIZE == 64
typedef long int                int64_t;
# else
__extension__
typedef long long int           int64_t;
# endif
#endif
```

Types dérivés explicitent la taille et la nature de la variable

`/usr/lib/avr/include/stdint.h` :

```
/* ISO/IEC 9899:1999 7.18 Integer types */

/** 8-bit signed type. */
typedef signed char int8_t;

/** 8-bit unsigned type. */
typedef unsigned char uint8_t;

/** 16-bit signed type. */
typedef signed int int16_t;

/** 16-bit unsigned type. */
typedef unsigned int uint16_t;

typedef signed long int int32_t; // 32 signed
```

1. Atmel AVR4027, *Tips and Tricks to Optimize Your C Code for 8-bit AVR Microcontrollers*, Rev. 8453A-AVR-11/11

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 [2] (notation en complément à deux, donc inclut un bit de signe)

2. 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) [3] :

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

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

Imprécision de l'arithmétique à virgule flottante

```
int main()
{volatile float  f1=0.1,f2=10.;
 volatile double d1=0.1,d2=10.;
 int k;
 for (k=0;k<1000;k++)
   {f2+=f1;
    d2+=d1;
   }
 printf("%f %lf %0.9lf\n",f2,d2,(double)(d2-f2));
}
```

Alignement des exposants : un nombre flottant n'est pas exact et perd en précision lors d'opérations sur des ordres de grandeur très différents.

Réponse : 109.998894 110.000000 0.001106262

Cas de la fonction affine

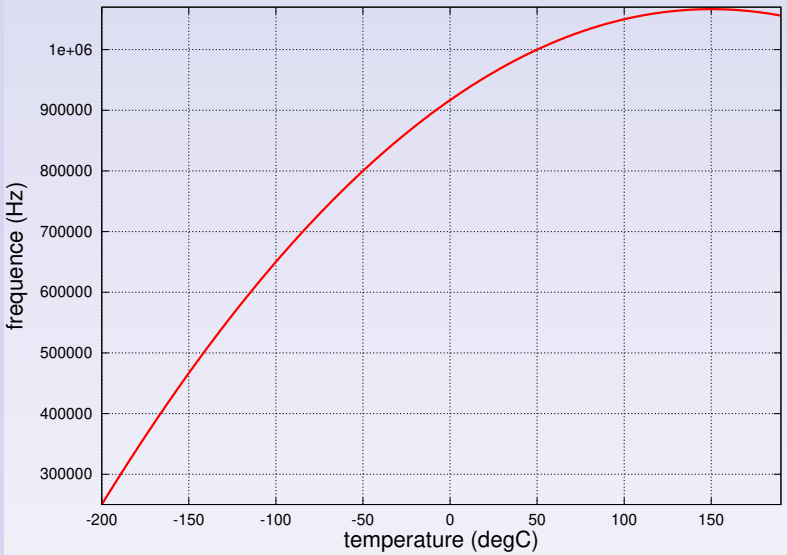
1. Un microcontrôleur manipule une donnée dans sa représentation : une fréquence de DDS est comprise entre 0 et f_{CK}, représentée par $[0..2^N - 1]$,
2. un humain veut connaître une fréqeunce en Hz

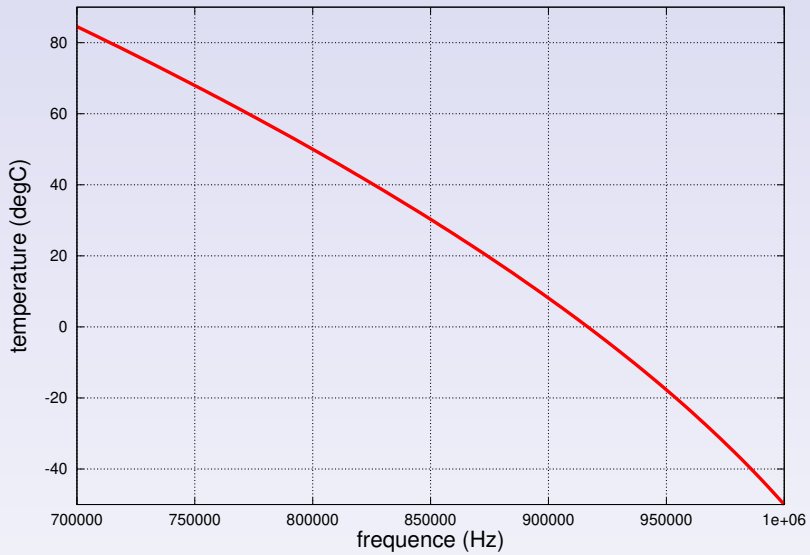
$$f_{Hz} = \frac{mot}{2^N} \cdot f_{CK}$$

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

- la loi liant fréquence et température est du type
  $f = \alpha T^2 + \beta T + \gamma \Leftrightarrow \alpha T^2 + \beta T + \gamma - f = 0 \xrightarrow{\text{racine}} T = A \pm \sqrt{B + Cf}$

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

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

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

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

1. $(T-A)^2 = B + C \times f$ donc[4] $2(T-A)dT - 2(T-A)dA = dB + dC \times f \Rightarrow |dT| = \left|\frac{dB}{2(T-A)}\right| + \left|\frac{dC \times f}{2(T-A)}\right| + |dA|$

4. dérivées partielles selon les inconnues A, B, C, T puisque la mesure f est parfaitement connue donc $df = 0$

Application des coefficients d'étalonnage

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

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

1. $(T-A)^2 = B + C \times f$ donc $2(T-A)dT - 2(T-A)dA = dB + dC \times f \Rightarrow |dT| = \left|\frac{dB}{2(T-A)}\right| + \left|\frac{dC \times f}{2(T-A)}\right| + |dA|$
2. on doit donc travailler avec chacun de ces termes inférieurs à 0,1.

Application des coefficients d'étalonnage

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

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

1. $(T-A)^2 = B + C \times f$ donc $2(T-A)dT - 2(T-A)dA = dB + dC \times f \Rightarrow |dT| = \left|\frac{dB}{2(T-A)}\right| + \left|\frac{dC \times f}{2(T-A)}\right| + |dA|$
2. on doit donc travailler avec chacun de ces termes inférieurs à 0,1.
3. Application numérique : par conception, $T - A \geq 100$ et $f \leq 10^6$ donc $dB \leq 10$, $dA \leq 0,1$ et $dC \leq 10^{-5}$

Application des coefficients d'étalonnage

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

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

1. $(T-A)^2 = B + C \times f$ donc $2(T-A)dT - 2(T-A)dA = dB + dC \times f \Rightarrow |dT| = \left|\frac{dB}{2(T-A)}\right| + \left|\frac{dC \times f}{2(T-A)}\right| + |dA|$
2. on doit donc travailler avec chacun de ces termes inférieurs à 0,1.
3. Application numérique : par conception, $T - A \geq 100$ et $f \leq 10^6$ donc $dB \leq 10$, $dA \leq 0,1$ et $dC \leq 10^{-5}$
4. on va donc travailler sur $A \times 10$ et $C \times 10^5$ pour respecter la résolution recherchée sur des calculs entiers

Application des coefficients d'étalonnage

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

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

1. $(T-A)^2 = B + C \times f$ donc $2(T-A)dT - 2(T-A)dA = dB + dC \times f \Rightarrow |dT| = \left|\frac{dB}{2(T-A)}\right| + \left|\frac{dC \times f}{2(T-A)}\right| + |dA|$
2. on doit donc travailler avec chacun de ces termes inférieurs à 0,1.
3. Application numérique : par conception, $T - A \geq 100$ et $f \leq 10^6$ donc $dB \leq 10$, $dA \leq 0,1$ et $dC \leq 10^{-5}$
4. on va donc travailler sur $A \times 10$ et $C \times 10^5$ pour respecter la résolution recherchée sur des calculs entiers
5. cependant, $C \simeq 0,1$ donc $C \times 10^5 \times f \simeq 10^{10}$ qui nécessite 34 bits,

Application des coefficients d'étalonnage

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

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

1. $(T-A)^2 = B + C \times f$ donc $2(T-A)dT - 2(T-A)dA = dB + dC \times f \Rightarrow |dT| = \left|\frac{dB}{2(T-A)}\right| + \left|\frac{dC \times f}{2(T-A)}\right| + |dA|$
2. on doit donc travailler avec chacun de ces termes inférieurs à 0,1.
3. Application numérique : par conception, $T - A \geq 100$ et $f \leq 10^6$ donc $dB \leq 10$, $dA \leq 0,1$ et $dC \leq 10^{-5}$
4. on va donc travailler sur $A \times 10$ et $C \times 10^5$ pour respecter la résolution recherchée sur des calculs entiers
5. cependant, $C \simeq 0,1$ donc $C \times 10^5 \times f \simeq 10^{10}$ qui nécessite 34 bits,
6. 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

Application des coefficients d'étalonnage

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

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

1. $(T-A)^2 = B + C \times f$ donc $2(T-A)dT - 2(T-A)dA = dB + dC \times f \Rightarrow |dT| = \left|\frac{dB}{2(T-A)}\right| + \left|\frac{dC \times f}{2(T-A)}\right| + |dA|$
2. on doit donc travailler avec chacun de ces termes inférieurs à 0,1.
3. Application numérique : par conception, $T - A \geq 100$ et $f \leq 10^6$ donc $dB \leq 10$, $dA \leq 0,1$ et $dC \leq 10^{-5}$
4. on va donc travailler sur $A \times 10$ et $C \times 10^5$ pour respecter la résolution recherchée sur des calculs entiers
5. cependant, $C \simeq 0,1$ donc $C \times 10^5 \times f \simeq 10^{10}$ qui nécessite 34 bits,
6. 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

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

Conclusion : $10 \cdot \mathrm{A}$, $10^4 \cdot \mathrm{B}$ **et** $10^5 \cdot \mathrm{C}$ **précalculés**

Application des coefficients d'étalonnage

En exprimant le problème

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

comme

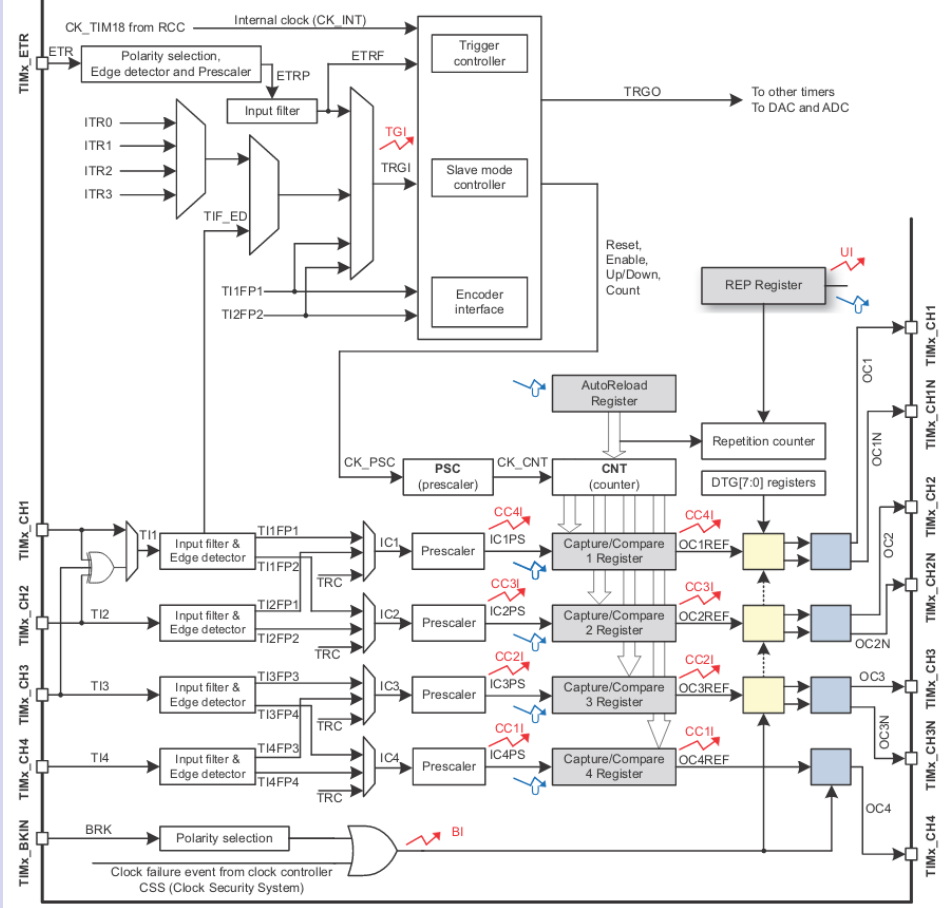
$$10 \times T = 10 \times A + \sqrt{B \times 10^4 + (10^5 C)\,(f/10)}/10$$

- tous les calculs se font sur des entiers
- calculer $10T$ garantit la résolution recherchée (0,1 K)
- l'exactitude du calcul est garanti
  - pas d'imprécision du calcul flottant
  - pas de dépassement de capacité de stockage

Timers

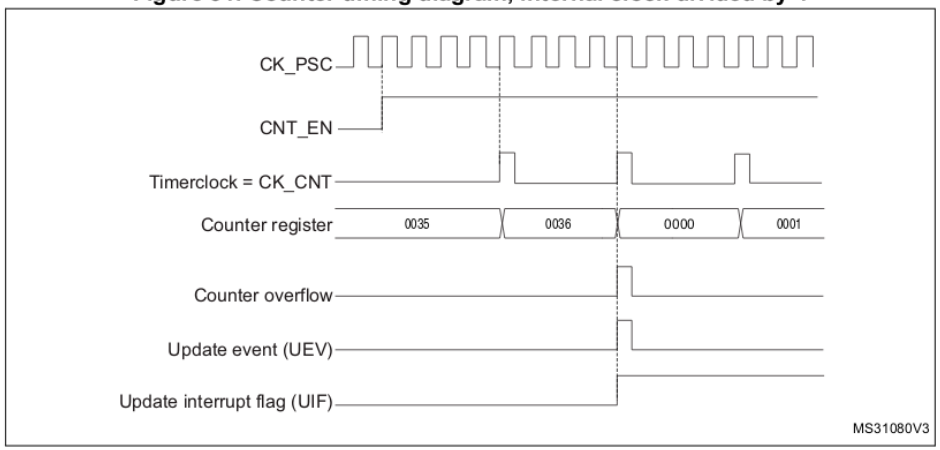
https://www.st.com/resource/en/datasheet/stm32f410cb.pdf

1. comptage par incrément ou décrément
2. fréquence du processeur divisé
3. remise à zéro quand la borne du compteur est atteinte
4. Output Compare : changement d'état de la broche lorsque le compteur atteint une valeur




```
#include <libopencm3/stm32/timer.h>
rcc_periph_clock_enable (RCC_TIM1);  // apb2_frequency=70 MHz, cf p.40 de RM0401
gpio_set_output_options(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ, GPIO8 );
gpio_mode_setup (GPIOA, GPIO_MODE_AF, GPIO_PUPD_NONE, GPIO8);
gpio_set_af (GPIOA, GPIO_AF1, GPIO8);
rcc_periph_reset_pulse(RST_TIM1); // timer_reset (TIM1); modif 180826 /w new libopencm3 ver
timer_set_mode (TIM1, TIM_CR1_CKD_CK_INT, TIM_CR1_CMS_EDGE, TIM_CR1_DIR_UP);
// TIM_CR1_CKD_CK_INT=clock division ratio
// TIM_CR1_CMS_EDGE=counting mode
// TIM_CR1_DIR_UP=count direction
// https://github.com/libopencm3/libopencm3/blob/master/lib/stm32/common/timer_common_all.c
timer_set_oc_mode(TIM1, TIM_OC1, TIM_OCM_PWM2);
timer_enable_oc_output(TIM1, TIM_OC1);
timer_enable_break_main_output(TIM1);
timer_set_oc_value(TIM1, TIM_OC1, 1);
timer_set_prescaler(TIM1,0);  // pp.94 & 130: prescaler !=1 => CK=2*APB2
timer_set_period (TIM1,1);
timer_enable_counter (TIM1);
```

Timers



https://www.st.com/resource/en/datasheet/stm32f410cb.pdf

1. comptage par incrément ou décrément
2. fréquence du processeur divisé
3. remise à zéro quand la borne du compteur est atteinte
4. Output Compare : changement d'état de la broche lorsque le compteur atteint une valeur

```
#include <libopencm3/stm32/timer.h>
rcc_periph_clock_enable (RCC_TIM1);  // apb2_frequency=70 MHz, cf p.40 de RM0401
gpio_set_output_options(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ, GPIO8 );
gpio_mode_setup (GPIOA, GPIO_MODE_AF, GPIO_PUPD_NONE, GPIO8);
gpio_set_af (GPIOA, GPIO_AF1, GPIO8);
rcc_periph_reset_pulse(RST_TIM1); // timer_reset (TIM1); modif 180826 /w new libopencm3 ver
timer_set_mode (TIM1, TIM_CR1_CKD_CK_INT, TIM_CR1_CMS_EDGE, TIM_CR1_DIR_UP);
// TIM_CR1_CKD_CK_INT=clock division ratio
// TIM_CR1_CMS_EDGE=counting mode
// TIM_CR1_DIR_UP=count direction
// https://github.com/libopencm3/libopencm3/blob/master/lib/stm32/common/timer_common_all.c
timer_set_oc_mode(TIM1, TIM_OC1, TIM_OCM_PWM2);
timer_enable_oc_output(TIM1, TIM_OC1);
timer_enable_break_main_output(TIM1);
timer_set_oc_value(TIM1, TIM_OC1, 1);
timer_set_prescaler(TIM1,0);  // pp.94 & 130: prescaler !=1 => CK=2*APB2
timer_set_period (TIM1,1);
timer_enable_counter (TIM1);
```

https://www.st.com/resource/en/reference_manual/dm00180366.pdf section 14

Timers

https://www.st.com/resource/en/datasheet/stm32f410cb.pdf

1. comptage par incrément ou décrément
2. fréquence du processeur divisé
3. remise à zéro quand la borne du compteur est atteinte
4. Output Compare : changement d'état de la broche lorsque le compteur atteint une valeur

14.4.10 TIM1 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CNT[15:0]**: Counter value

14.4.11 TIM1 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

```
#include <libopencm3/stm32/timer.h>
rcc_periph_clock_enable (RCC_TIM1);  // apb2_frequency=70 MHz, cf p.40 de RM0401
gpio_set_output_options(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ, GPIO8 );
gpio_mode_setup (GPIOA, GPIO_MODE_AF, GPIO_PUPD_NONE, GPIO8);
gpio_set_af (GPIOA, GPIO_AF1, GPIO8);
rcc_periph_reset_pulse(RST_TIM1); // timer_reset (TIM1); modif 180826 /w new libopencm3 ver
timer_set_mode (TIM1, TIM_CR1_CKD_CK_INT, TIM_CR1_CMS_EDGE, TIM_CR1_DIR_UP);
// TIM_CR1_CKD_CK_INT=clock division ratio
// TIM_CR1_CMS_EDGE=counting mode
// TIM_CR1_DIR_UP=count direction
// https://github.com/libopencm3/libopencm3/blob/master/lib/stm32/common/timer_common_all.c
timer_set_oc_mode(TIM1, TIM_OC1, TIM_OCM_PWM2);
timer_enable_oc_output(TIM1, TIM_OC1);
timer_enable_break_main_output(TIM1);
timer_set_oc_value(TIM1, TIM_OC1, 1);
timer_set_prescaler(TIM1,0);  // pp.94 & 130: prescaler !=1 => CK=2*APB2
timer_set_period (TIM1,1);
timer_enable_counter (TIM1);
```

https://www.st.com/resource/en/reference_manual/dm00180366.pdf section 14

Exercices

1. Configurer le timer (fréquence, rapport cyclique) auquel est connectée la LED (lequel est-ce [4] ?) pour commuter suffisamment lentement pour que le clignotement soit visible
2. implémenter la fonction racine pour ne calculer *que sur des entiers* et que le résultat soit garanti à 4 décimales exact

4. on pourra consulter la datasheet du STM32F410 – `https://www.st.com/resource/en/datasheet/stm32f410cb.pdf` –en se rappelant que la LED bicolore est commandée par PC1 – voir page 43