Fiche explicative : les Timers

Periph'Team - INSA de Toulouse

1 Principe et fonctionnement

1.1 Le timer on peut toujours compter dessus

La fonction première d'un timer est de compter le temps qui s'écoule. Pour réaliser cela, son fonctionnement est basé sur un **compteur électronique**. Comme tout compteur numérique, il est d'une taille bornée et peut donc avoir une **résolution** de 8 bits (0 à 255), 16 bits (0 à 65 535) ou encore 32 bits (0 à 4 294 967 295)[1].

Le compteur va s'incrémenter (ou se décrémenter en fonction de sa configuration) à chaque fois qu'il reçoit un événement donné. Comme présenté sur le schéma 1, la source de l'événement peut être :

- une broche du microcontrôleur : dans ce cas le compteur compte les fronts sur cette broche,
- une horloge : dans ce cas le compteur compte les fronts d'horloge et donc le temps.

C'est ce dernier cas qui nous intéresse et on parle alors bien d'un **timer**.

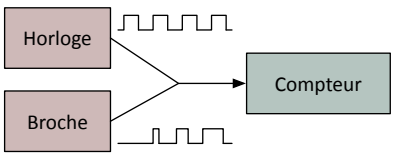

Figure 1: *Sources d'un compteur*

L'entrée d'horloge est très souvent précédée d'un **Prescaler** (voir figure 2). Son rôle est d'opérer une première division de la fréquence de l'horloge avant d'attaquer concrètement l'horloge du compteur[2].

Enfin, précisions que le compteur est très souvent associé à un registre dit **Autoreload**. Celui-ci contient la valeur de redémarrage du compteur après un débordement (*overflow* ou *underflow*). Un débordement survient quand la valeur du compteur doit s'incrémenter (dans le cas d'une configuration par incrément) au delà de la valeur de l'Autoreload. Dans ce cas, le compteur ne prend pas la valeur Autoreload+1, mais il repart à 0 (voir schéma 3).

Remarquez que si l'Autoreload et le Prescaler ne changent pas, un débordement survient de manière périodique. Le timer permet donc de provoquer un nouvel événement avec une **période** fixe. Le programmeur peut ainsi utiliser cet événement pour provoquer des traitements de manière périodique par exemple en utilisant le débordement comme la source d'une interruption.

[1] Il est possible de mettre en cascade des Timers pour en augmenter la résolution, mais nous n'en parlerons pas ici.
[2] Le schéma est très simpliste, et ne permet d'appréhender le timer que dans le but de générer un évènement périodique.

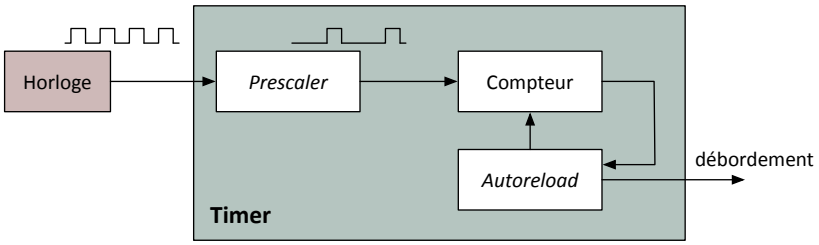

Figure 2: *Décomposition d'un timer*

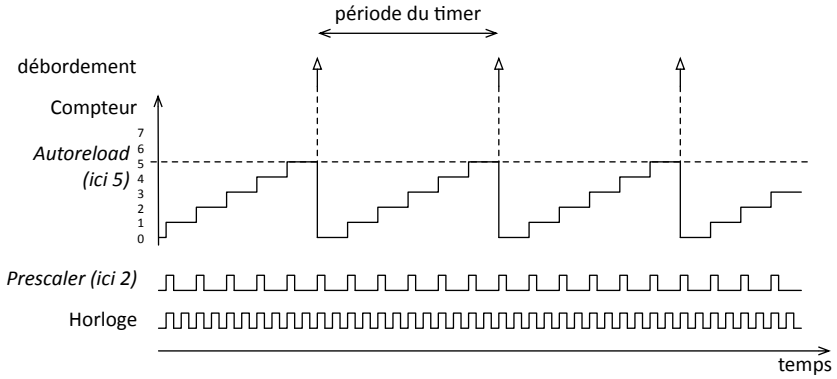

Figure 3: *Evolution d'un timer (ici en mode incrémentation avec une valeur de 5 pour l'Autoreload)*

1.2 Et les horloges dans tout cela ?

Vous avez dû comprendre que pour utiliser un timer, il faut une horloge et connaître sa fréquence. Sur les STM32 il existe une unique source d'horloge, mais celle-ci est ensuite divisée en différentes "branches" (voir schéma dans le RM page 92). Chaque périphérique est ensuite connecté à une de ces branches. Par défaut les branches d'horloge sont toutes désactivées, c'est-à-dire que les périphériques ne reçoivent pas de signale d'horloge. Un périphérique étant un circuit synchrone, s'il n'y a pas d'horloge rien ne se passe ! L'intérêt est l'économie d'énergie. En effet, un circuit électronique relié à une horloge, même s'il ne pilote pas de dispositif, dissipe une puissance au cube de la fréquence d'horloge. C'est la raison pour laquelle, il faut, **pour chaque périphérique activer l'horloge locale au périphérique** et cela avant même de commencer à le configurer.

Par exemple, pour activer l'horloge du *GPIO A*, il faut écrire:

```
RCC->APB2ENR = RCC->APB2ENR | RCC_APB2ENR_IOPAEN ;
```

En effet le GPIO A est lié à la branche d'horloge *APB2*.

Donc avant même de se poser la question de la fréquence d'horloge d'entrée d'un timer, il faudra penser à activer son horloge si on veut pouvoir l'utiliser !

Pour trouver la fréquence des horloges du STM32, vous aurez remarqué sur le schéma de l'arbre d'horloge page 92 du *RM* qu'il y faut connaître la fréquence de la source initiale et ensuite la valeur des *prescaler* de chaque branche (sans parler des PLL dont seuls les électroniciens en comprennent les secrets)...

La source initiale peut être soit un quartz externe, soit un circuit interne. Pour la carte Nucleo, l'entrée *HSE* du *STM32* est relié à un 8 MHz venant du quartz de la partie debug de la board[3].

Au niveau de la configuration des PLL et prescaler, heureusement tout cela est déjà fait dans les fichiers de `startup` ! Dans la plupart du temps, et nous nous contenterons de cette hypothèse dans un premier temps, il est plus simple de considérer que la fréquence du coeur (*SYSCLK_FREQ*) et de celle de différents timers (*TIMxCLK_FREQ*) sont de 72 Mhz.

[3] Si on casse le pcb pour extraire la partie debug alors il n'y a plus de quartz à 8 MHZ : il faut souder un quartz sur l'emplacement *X3* laissé libre pour retrouver un *HSE* un à 8MHz (sinon se fier au circuit RC interne de la puce).

2 Documents de référence

Pour configurer le timer, vous aurez besoin de deux sections du *Reference Manual* :

- La section 7.2 *Clocks* dans laquelle vous trouverez les différentes registres permettant d'activer les horloges des périphériques,
- La section 15 sur les *General-purpose timers* en particulier les section 15.3.1 et 15.3.2.

Pour l'interruption, le principe est le même que pour le TP précédent et vous aurez donc besoin du *Reference Manual* (section 10.1.2) pour identifier l'interruption du timer et du *Programming manual* (section 4.3) pour configurer le *NVIC*.