XF : Rapport

Favre Lenny

PTR

I5 - 2020

Table des matières

[Introduction 3](#_Toc54189678)

[Modifications importantes 3](#_Toc54189679)

[addTimeout 3](#_Toc54189680)

[tick 4](#_Toc54189681)

[Résultats 5](#_Toc54189682)

[Testbench 1 6](#_Toc54189683)

[Testbench 2 7](#_Toc54189684)

[Testbench 3 8](#_Toc54189685)

[Testbench 4 9](#_Toc54189686)

[Testbench 5 10](#_Toc54189687)

# Introduction

Dans ce laboratoire, nous avons dû compléter la Framework incomplète du XF donnée par le professeur. Ce rapport va décrire les fonctions/méthodes importantes implémentées pour compléter le code et obtenir des bons résultats quelle que soit la cible, Windows avec Qt ou embarqué sur STM32 avec System Workbench.

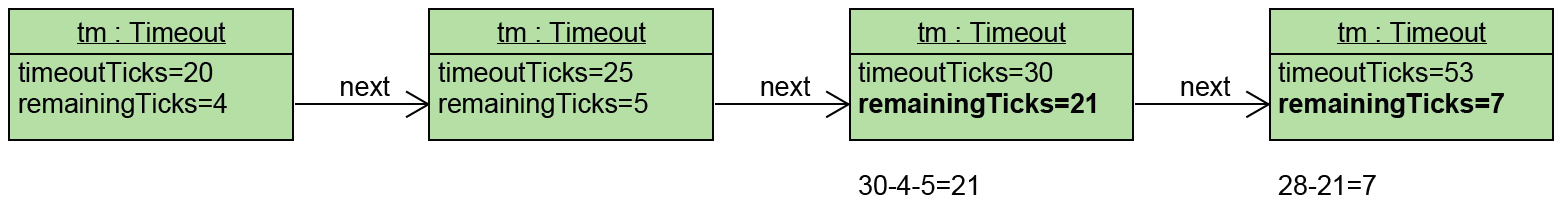
# Modifications importantes

Voici les méthodes importantes implémentées avec une explication pour chacune.

La classe XFTimeoutManagerDefault contient 2 fonctions importantes : addTimeout(XFTimeout \*pNewTimeout) et tick().

## addTimeout

La méthode addTimeout ajoute un timeout dans la liste des timeouts. Les timeouts ont 2 attributs : timeoutTicks qui contient le nombre total de millisecondes après les quelles le timeout doit s’exécuter. Et remainingTicks qui contient le nombre de millisecondes restantes avant que le timeout doive s’exécuter. Les timeouts sont organisés en liste pour qu’à chaque tick, seul le premier timeout est décrémenté.



La méthode addTimeout cherche donc où doit se positionner le timeout dans la liste et calcule ses remainingTicks.

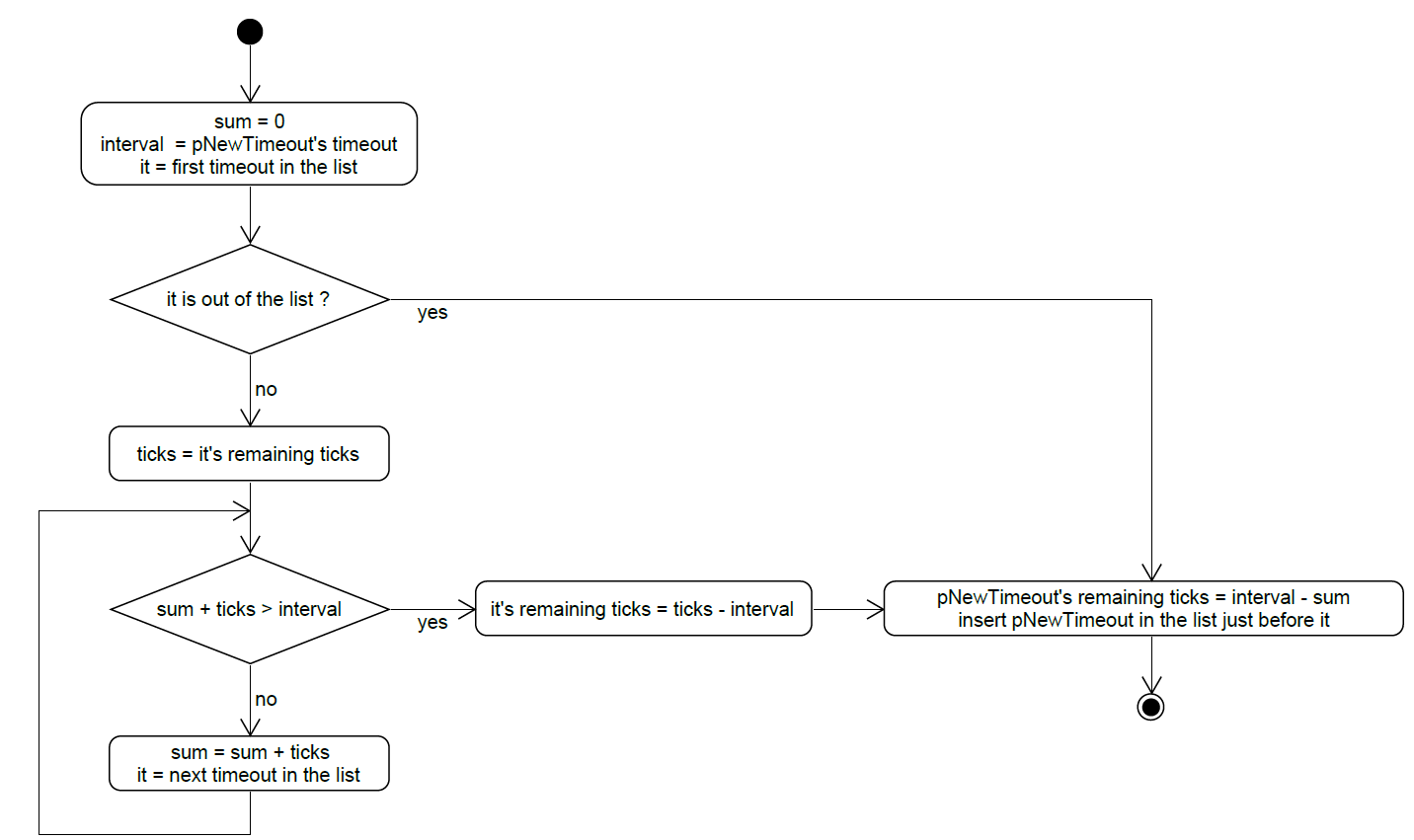


Figure : Diagramme d'activité addTimeout

## tick

La méthode tick décrémente les remainingTicks du premier timeout dans la liste. S’ils sont à zéro, ce timeout est exécuté et effacé de la liste des timeouts. Le prochain timeout est ensuite vérifié. Si ses remainingTicks sont aussi à zéro (Les deux timeouts doivent s’exécuter en même temps), il est aussi exécuté. On continue jusqu’à ce qu’un timeout n’ait pas ses remainingTicks à zéro ou si la liste est vidée.

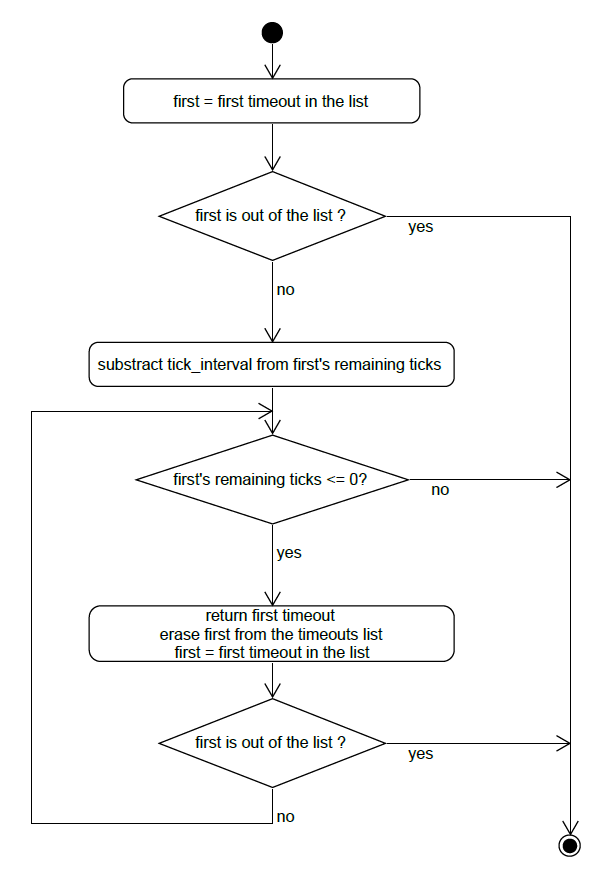


Figure : Diagramme d'activité ticks

# Résultats

Dans cette section, le code écrit sera testé sur 5 testbenches. Ces testbenches fourniront en sortie sur l’invité de commande sur Qt ou via l’USB virtual COM port sur la carte de développement STM32. Nous pourrons comparer les temps entre les événements du XF écrit avec des temps fournit par le professeur.

## Testbench 1

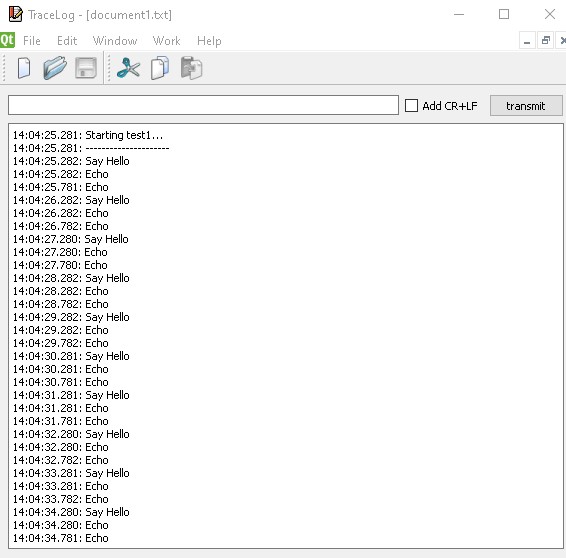
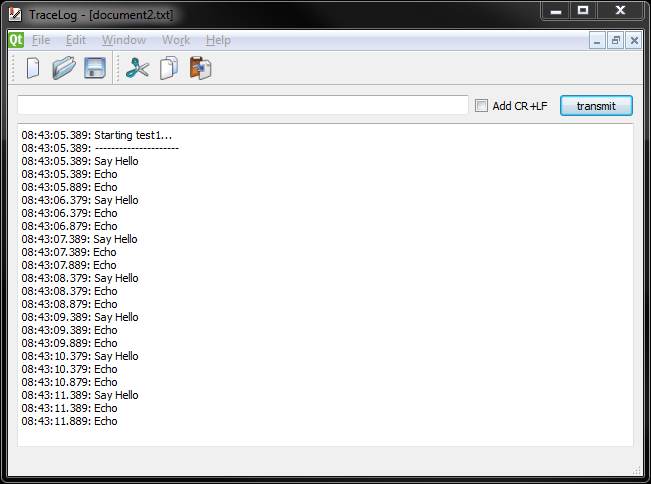


Figure 3: Temps attendus

Figure 4 : Temps mesurés avec Qt

Figure 5 : Temps mesurés avec STM32

## Testbench 2

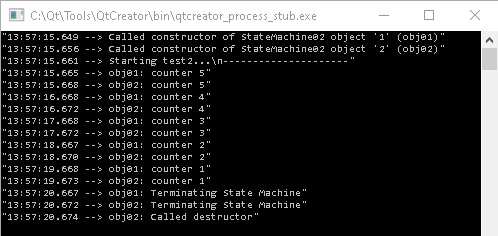
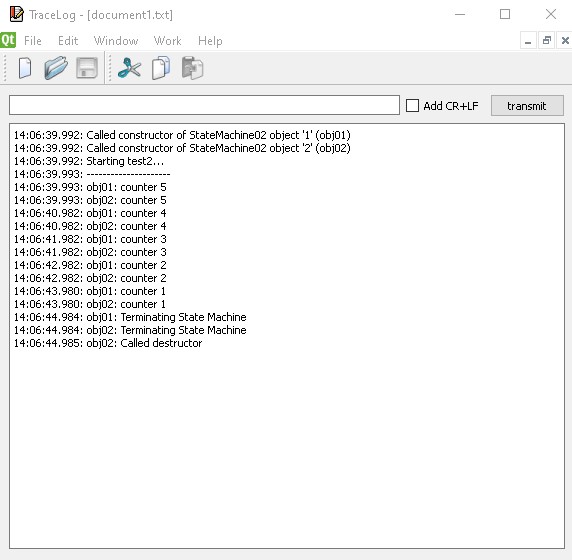
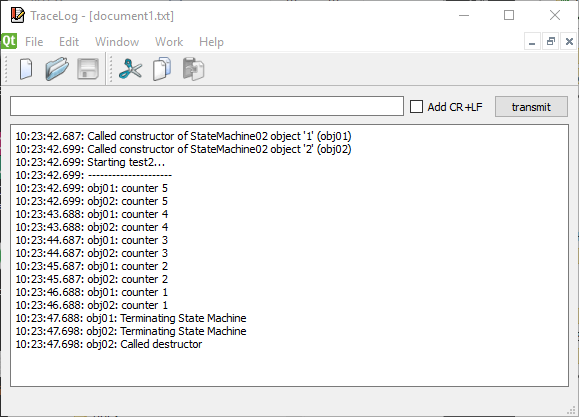


Figure 6 : Temps attendus

Figure 7 : Temps mesurés avec Qt

Figure 8 : Temps mesurés avec STM32

## Testbench 3

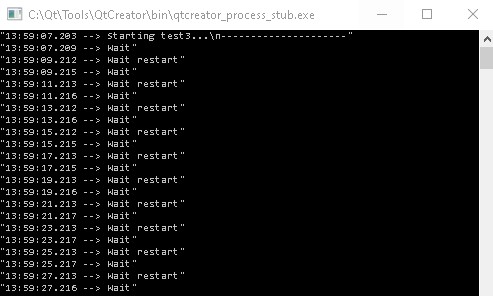
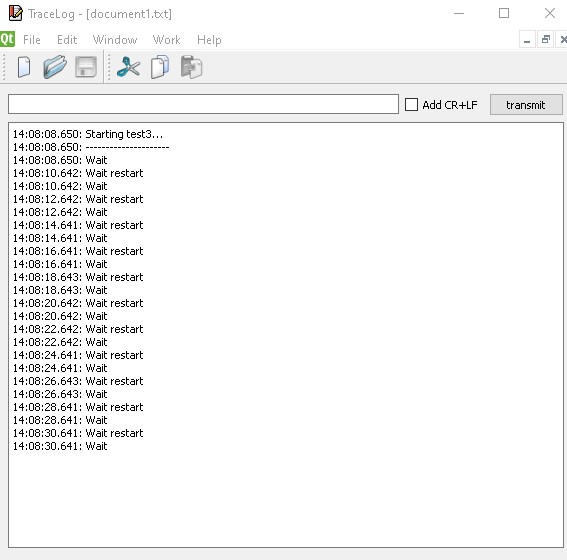
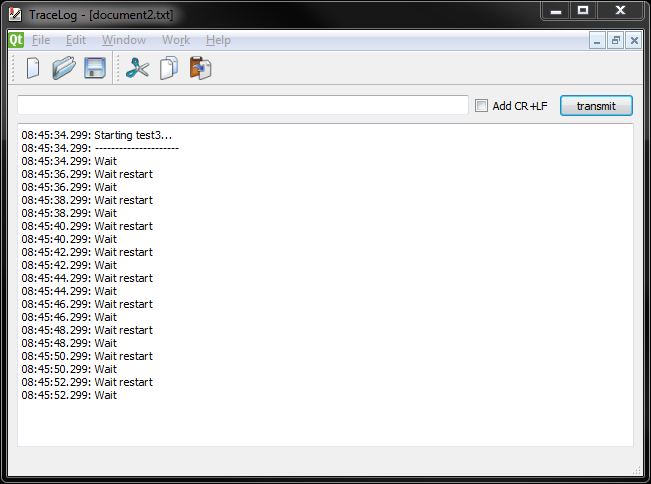


Figure 9 : Temps attendus

Figure 10 : Temps mesurés avec Qt

Figure 11 : Temps mesurés avec STM32

## Testbench 4

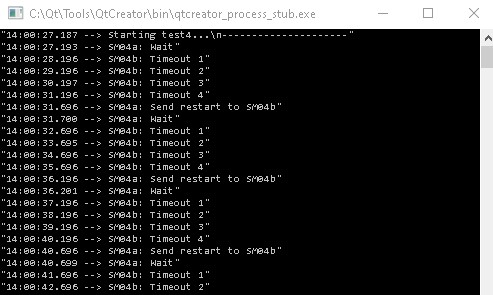
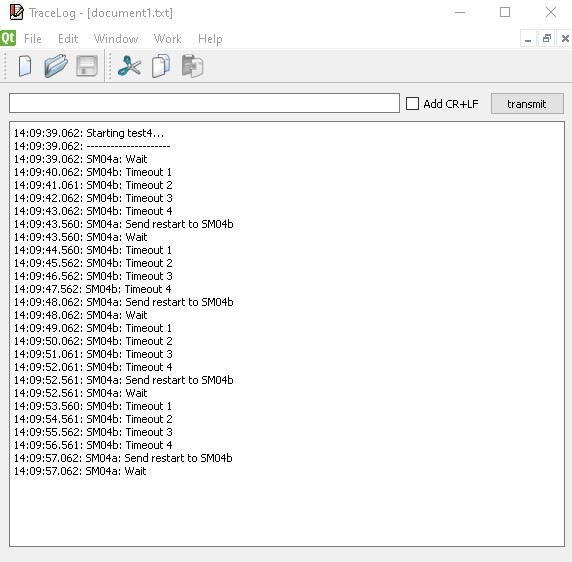
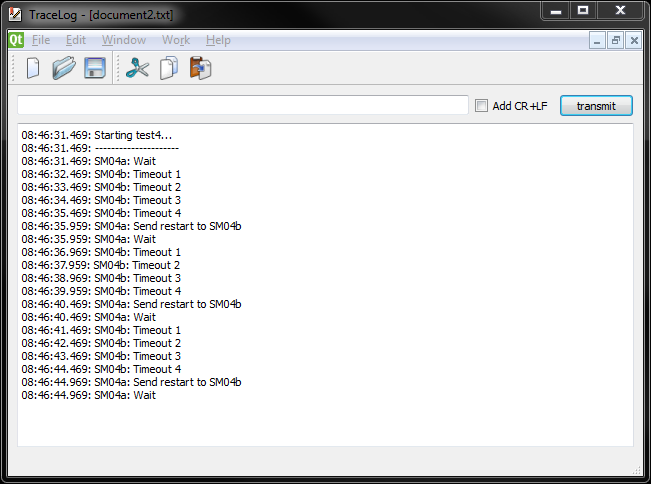


Figure 12 : Temps attendus

Figure 13 : Temps mesurés avec Qt

Figure 14 : Temps mesurés avec STM32

## Testbench 5

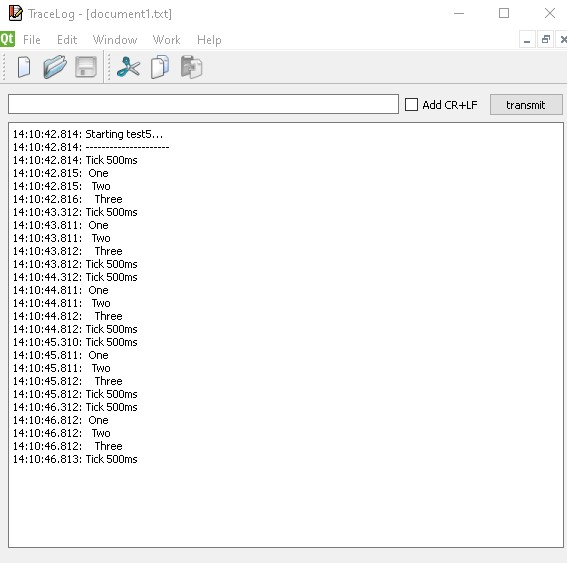
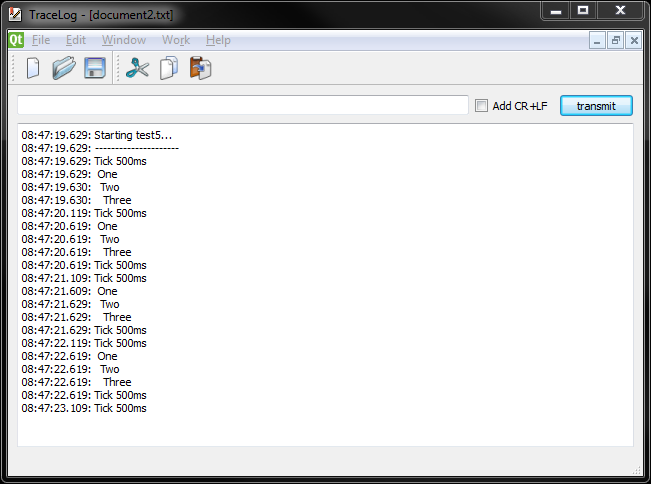


Figure 15 : Temps attendus

Figure 16 : Temps mesurés avec Qt

Figure 17 : Temps mesurés avec STM32