# Rapport : laboratoire oscilloscope PTR

## Tâche 5 - Configuration du Timing

Comme prochaine tâche, nous voulons analyser les temps d'exécution des différents composants du logiciel. En principe, nous avons deux composants qui peuvent être exécutés à deux vitesses différentes :

# Composant Timing

1 Conversion du signal analogue 1 kHz ou plus

2 Rafraichissement de l'écran 20 à 60 fois par seconde

Question 1

Est-ce qu'il est possible d'exécuter le composant #1 avec un XF (ou un RTOS) ? Justifiez votre réponse.

non car le xf exécute sa fonction principal processevent avec un intervalle de l'ordre de 1ms min. c'est à dire une fréquence de 1khz max.

On aura donc une fréquence de sampling de 1khz max ce qui nous permettrait de mesure uniquement des signaux ayant des fréquences inférieures à 500Hz.

Question 2

Est-ce qu'il est possible d'exécuter le composant #2 avec un XF (ou un RTOS) ? Justifiez votre réponse.

oui la fréquence est suffisamment basse

Question 3

Si l'on génère un signal de l'ordre de quelques kHz à l'aide d'un timer hardware et qu'on le combine avec un XF, lequel des deux composants doit être priorisé ? Justifiez votre réponse.

LE timer hardware est prioritaire car une bonne mesure du signal est prioritaire sur le fait d'avoir un affichage régulier des mesures sur l'écran.

## Tâche 7 - Configuration de l'ADC (Software Triggered)

Pour commencer, on désire pouvoir mesurer des signaux jusqu'à une fréquence de 1 kHz.

Question 1

Combien de mesures [Samples/s] le convertisseur A/D doit-il effectuer par seconde pour pouvoir échantillonner des signaux avec des fréquences jusqu'à 1 kHz ?

min 2000/sec mais étant donné qu'on veut réaliser un oscilloscope et avoir de beaux signaux il faudra aller bien en-dessus de ce que nous donne le théorème de Shannon

je pense que pour afficher des beaux signaux à 1KHz il faut sampler à min 10KHz

Question 2

Faut-il un filtre ? Si oui, quelle sera la fréquence de coupure de ce filtre ?

Oui il faut un filtre passe-bas qui coupe les hautes fréquences générées dans les fenêtrages.

On ne va pas l'implémenter dans ce projet mais idéalement il faudrait un filtre analogique avec une fréquence de coupure de la fréquence maximale

que l'on va mesure c'est-à-dire 1KHz pour fc ici.

Question 3

Est-ce la fréquence donnée par le théorème d’échantillonnage ou devrait-elle être plus élevée ?

Pour notre démarche, il nous faut le périphérique ADC3 du microcontrôleur. Les périphériques ADC1 et ADC2 ne seront pas utilisés dans ce projet.

Non il faut une fréquence de sampling plus élevée car on souhaite faire un oscilloscope et donc avoir des beaux signaux.

Question 4

Lequel des canaux du ADC3 doit être utilisé pour pouvoir mesurer / échantillonner le signal à l'aide de la broche PA0 ?

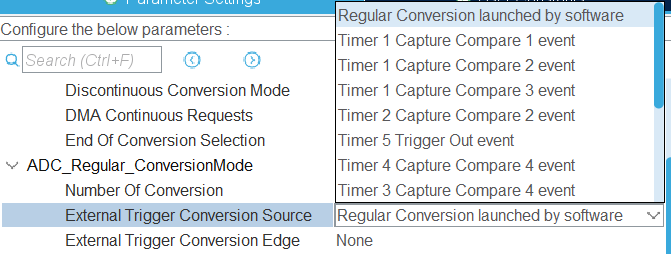
ADC3\_in0 (on le voit dans l’interface cubemx)



Question 5

Est-ce que le ADC pourrait éventuellement effectuer des mesures à des intervalles réguliers à l'aide de ses propres moyens ?

Oui, c’est directement configurable dans les paramètres de l’AD



On peut choisir quelle sera la source temporelle qui lancera régulièrement une conversion.

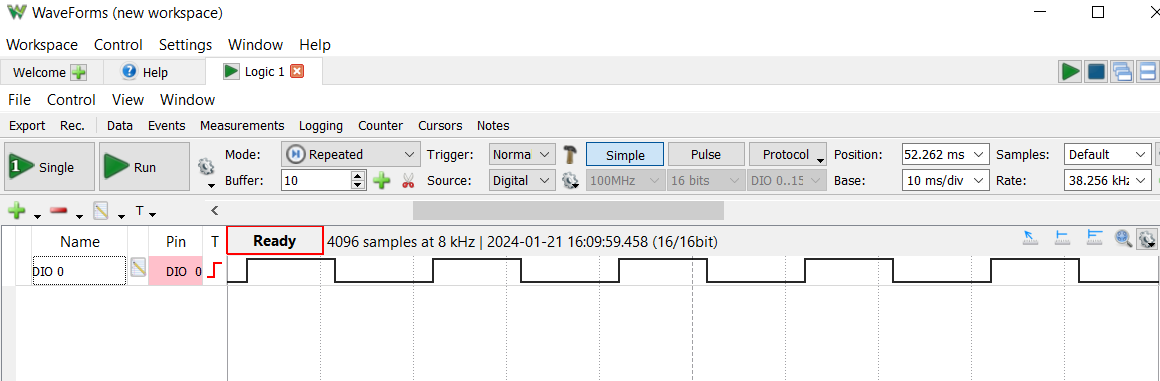
Nous allons choisir le single mod avec un timer externe afin d’être synchrone car le mode continu est asynchrone.

## Tâche 8 :

On fait compter le timer1 jusqu’à 19 999 c’est-à-dire 20 000 valeurs.

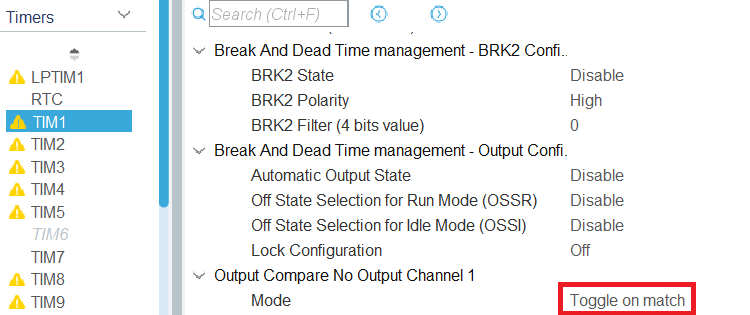
Car 200Mhz / 20000 = 10KHz.

A ce moment là je recrée mon projet dès le début car j’ai eu erreur de memory management fault. En effet, le programme s’arrête dans une fonction lors-ce que je le lance en mode debug.



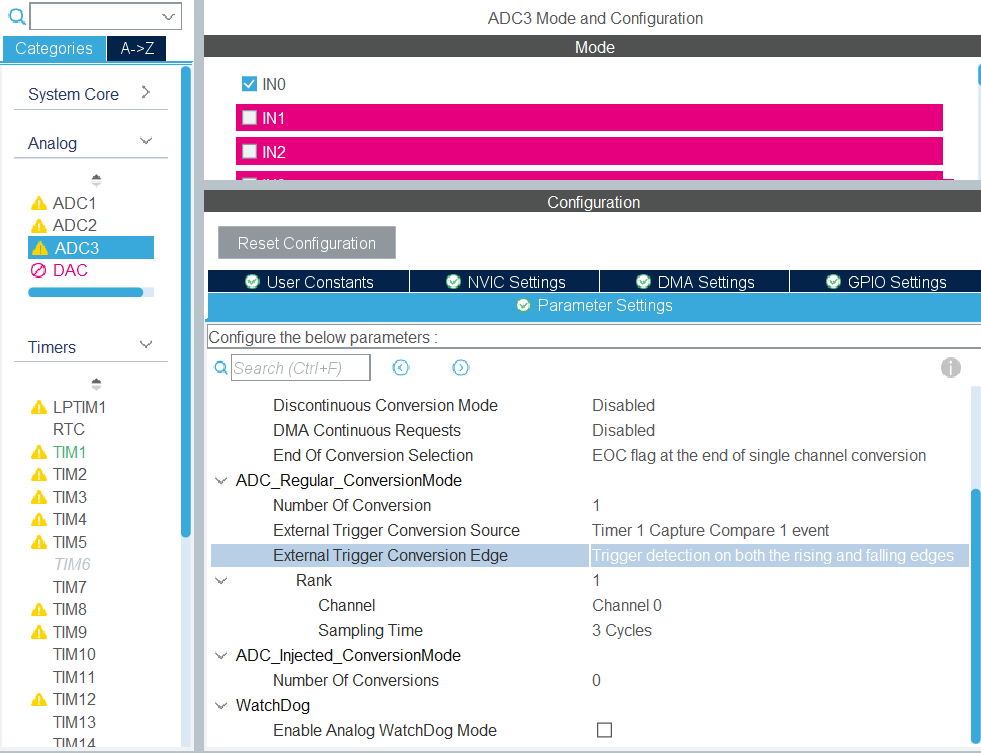
Mesure de la sortie sur le pin pf9.

## Tâche 9 :

2. 

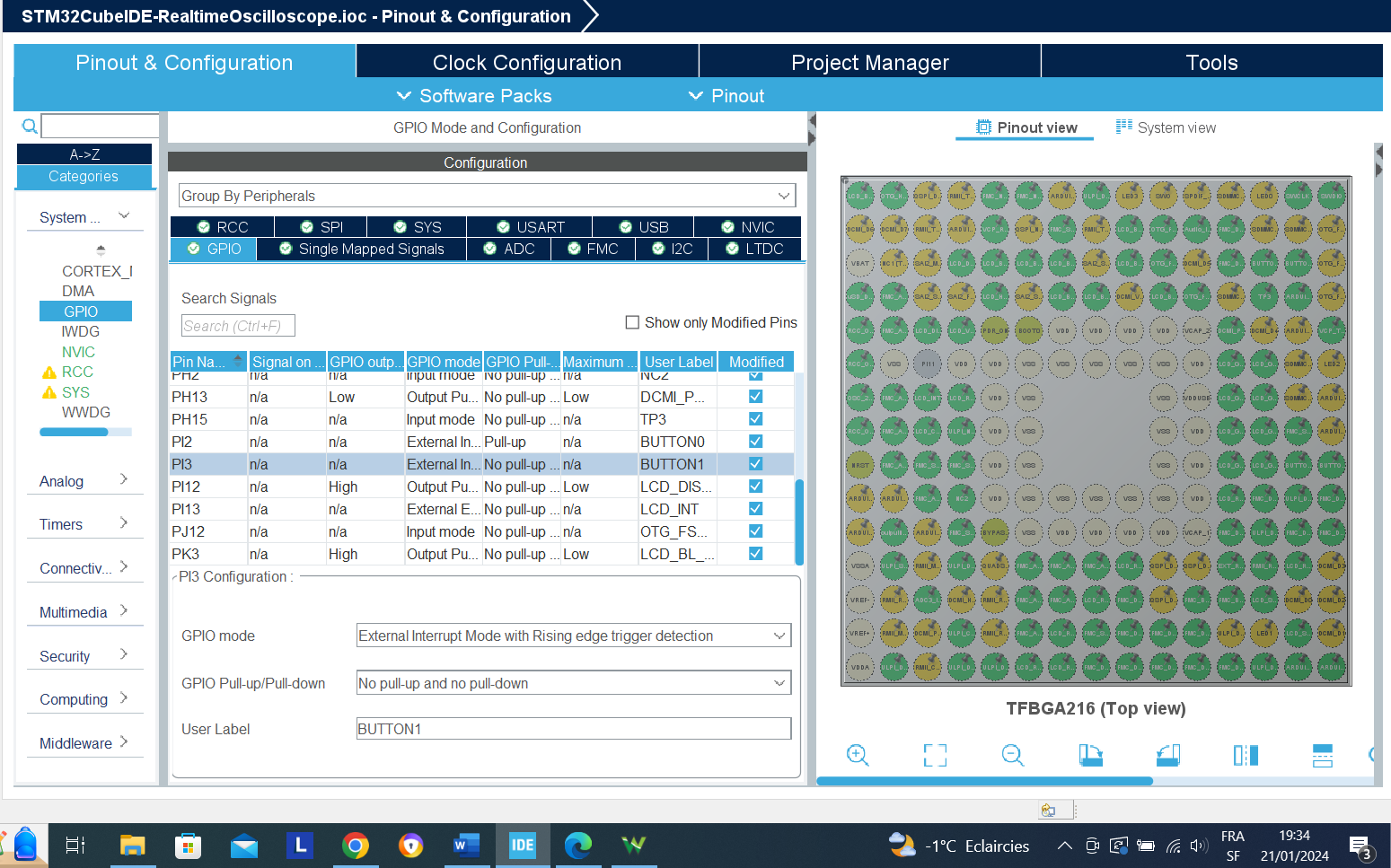
Etant donné que le timer1 est en mode toggle sa période est 2 fois plus rapide donc 5KHz.

On va donc demander à l’AD de lancer une conversion à chaque flanc montant et descendant. L’AD va donc sampler à 10KHz (parfait pour mesurer des signaux 1KHz).



Et on a évidement choisi que la source de trigger était le timer1.

## Tâche 10 :



On met les leds en gpio de sortie. On paramètre 4 inputs comme interruptions et on leur set les labels BUTTON0-3. Les 4 boutons ont des pull-up.

## Tâche 11 :

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Cette partie du code nous sert à utiliser TouchGFX. Il faut la copier dans notre main.c.

Une image contenant Appareils électroniques, Ingénierie électronique, Appareil électronique, Matériel d’ordinateur

Description générée automatiquement

Figure Etat de l’écran à la fin de la tâche 11.

## Tâche 12 :

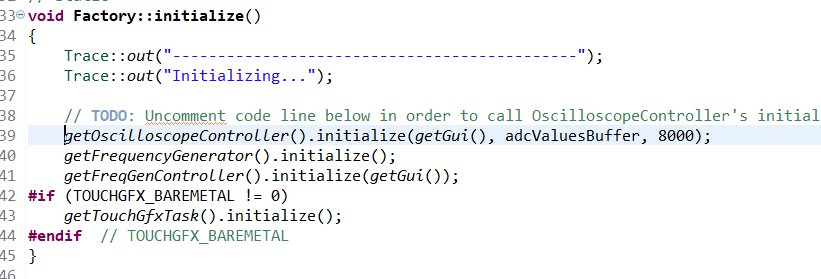


Figure init the controller in the factory class

On initialise le GUI puis on dessine les points.

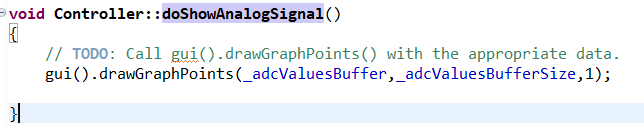


Figure We draw the point with the atributes of the class

Une image contenant Appareils électroniques, Ingénierie électronique, ordinateur, Appareil électronique

Description générée automatiquement

Figure screen works

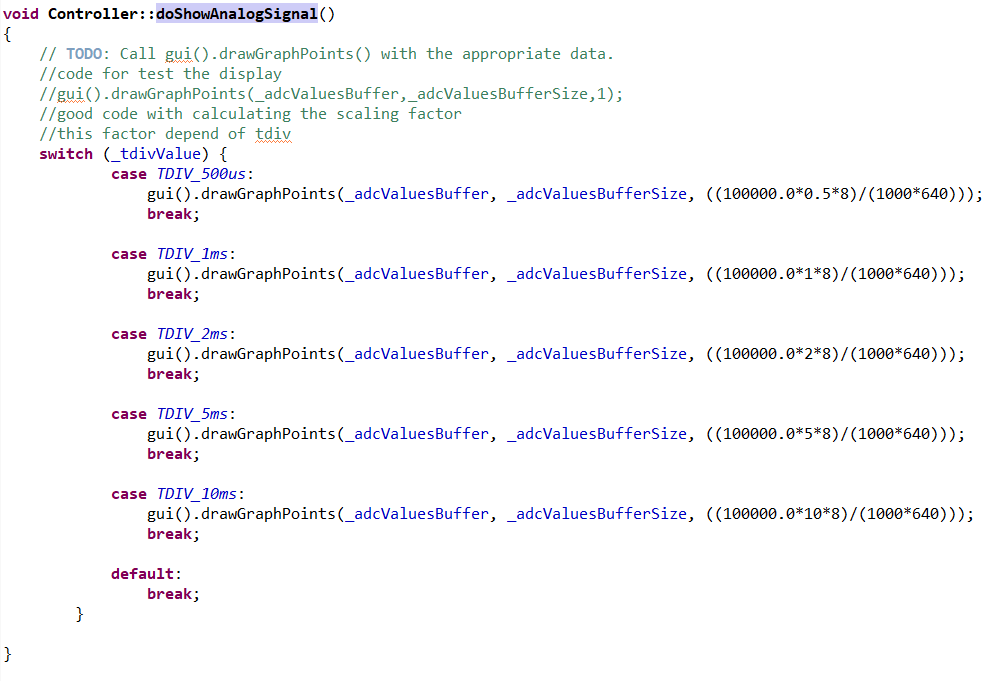


Figure scale factor calculation

On doit calculer le facteur de scaling c’est-à-dire le nombre de points du buffer qu’on va afficher.

## Tâche 13 :

Question 1 : Quelle fréquence d’échantillonnage peut être atteinte ?

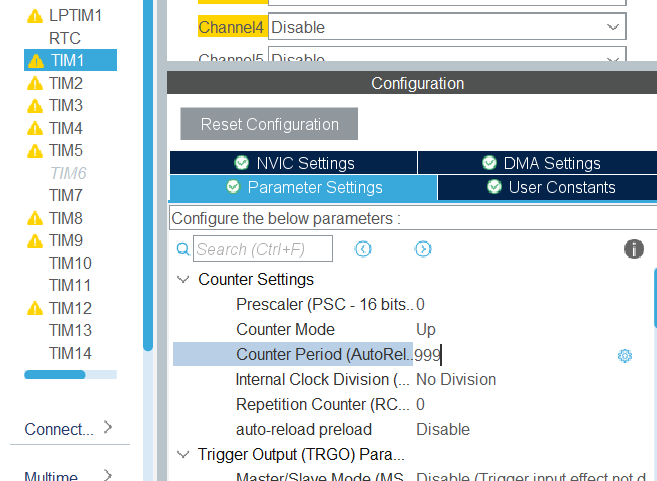


Figure counter period of TIM1 set to 999

Je vais à présent tester d’aller à 200Khz pour mon Timer1 c’est-à-dire sampler chaque 5us.

Question 2 : quel composant limite le système ? Je pense que c’est le XF car c’est le composant qui tourne le plus lentement du programme. La conception du XF ne permet pas d’aller à des vitesses plus élevées que de l’ordre de 100-1000Hz.

## Tâche 14 :

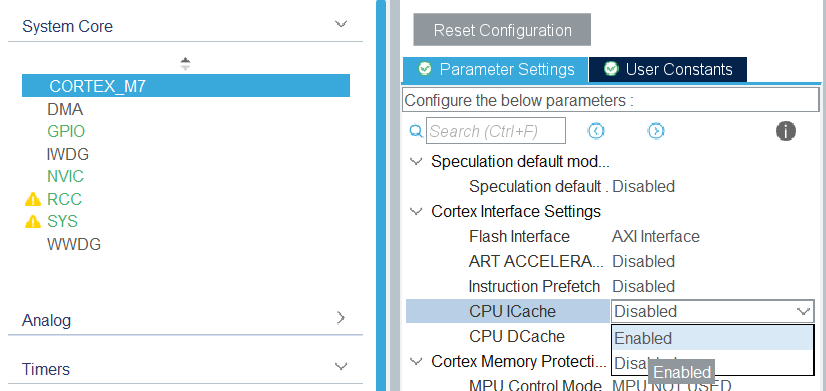


Figure Enable I and D cache

Question 1 : Quelle est à présent la fréquence d’échantillonnage maximale ? Je ne l’ai pas calculée mais je dirais qu’elle est plus élevée car le cache est une mémoire tampon qui a des accès plus rapides que la mémoire traditionnelle.

Question 2 : D’environ combien de pourcent la cache I et D améliore-t-elle la réactivité du système ?

## Tâche 15 :

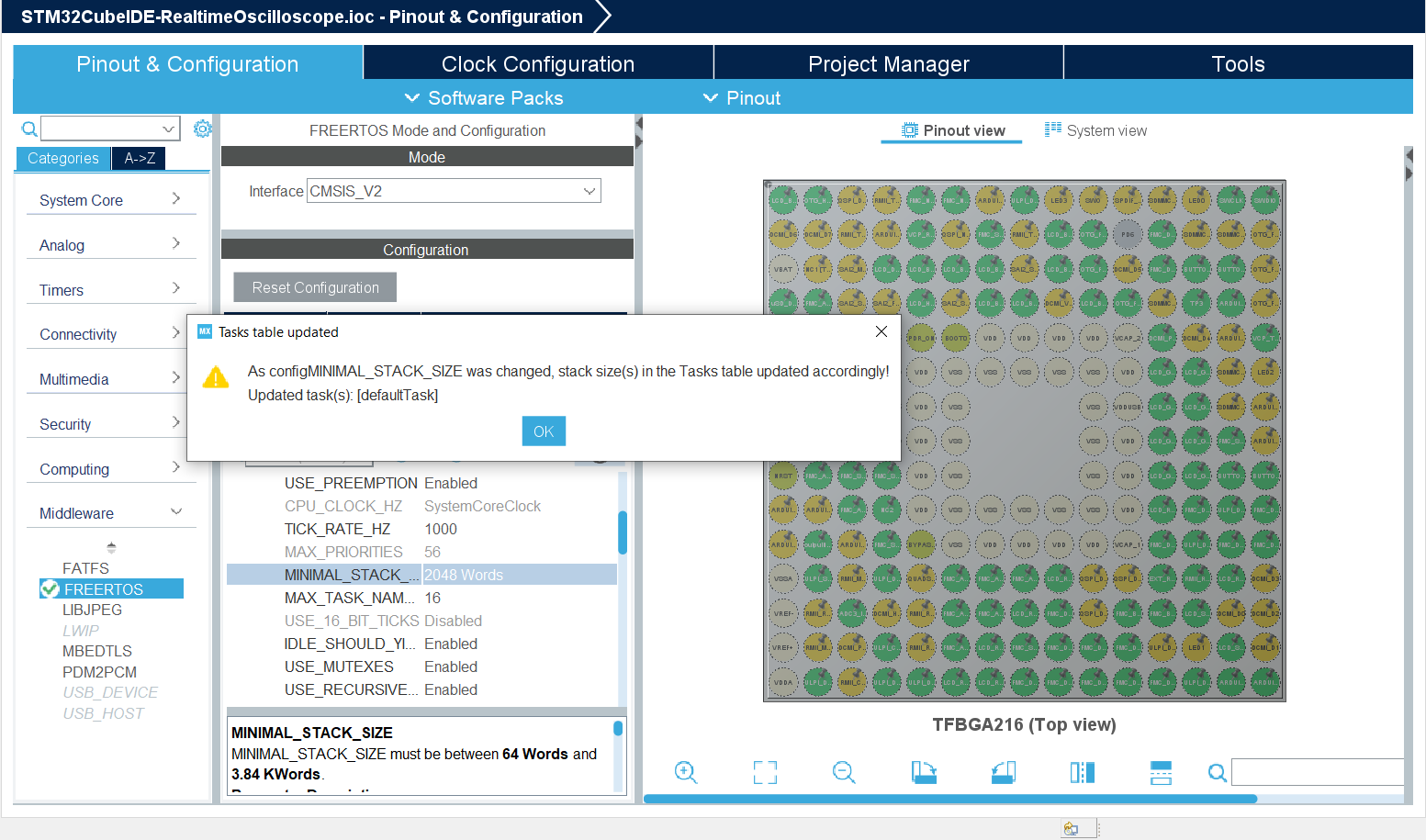


Figure Configuration of the freertOS

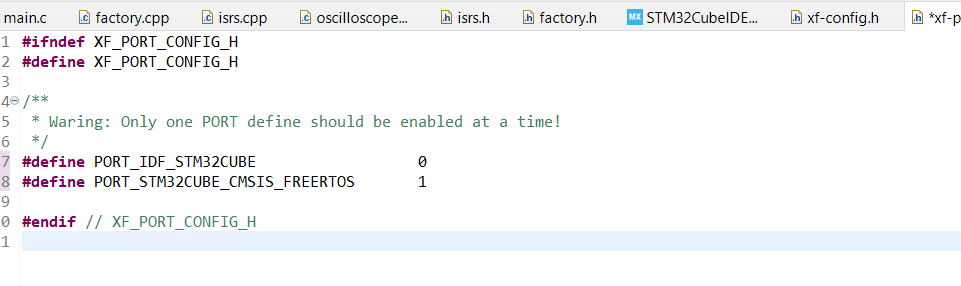


Figure Modifiez le fichier de configuration du XF (config/xf-port-config.h) afin d’utiliser FreeRTOS

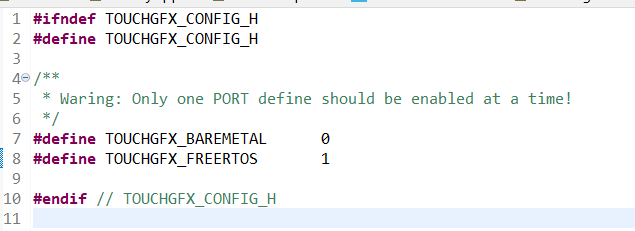


Figure Modifiez le fichier de configuration pour TouchGFX (config/touchgfx-config.h) afin d’utiliser FreeRTOS

Le build du programme affiche maintenant des erreurs (depuis la configuration du XF et l’ajout des paths).

[Question 1](https://course.hevs.io/ptr/students/documentation/realtime-oscilloscope/#/ch-fr/04-task-rtos-integration?id=question-1)

À quelle fréquence maximale peut-on régler l’échantillonnage ?

Je ne peux donc pas répondre à cette question

[Question 2](https://course.hevs.io/ptr/students/documentation/realtime-oscilloscope/#/ch-fr/04-task-rtos-integration?id=question-2)

Quels avantages voyez-vous à utiliser FreeRTOS dans cette application ?

On peut aller plus vite (Fs plus élevé) car on sépare l’affichage de la mesure.

[Question 3](https://course.hevs.io/ptr/students/documentation/realtime-oscilloscope/#/ch-fr/04-task-rtos-integration?id=question-3)

Donnez un exemple où un RTOS serait particulièrement nécessaire ?

Dans les applications où il faut un contrôle précis du temps : par exemple : placement de pièces avec un bras robotisé. En résumé, un RTOS est essentiel dans des systèmes où le respect des délais et la prédictibilité des opérations sont cruciaux, comme c'est souvent le cas dans les applications industrielles, médicales et aérospatiales.