Setup PC :

Je branche ma carte Nucleo-64 en USB et mon Logic Analyzer en USB.

Je regarde le PIN de la carte Nucleo-64 pour me brancher à la masse (GND), sur la LED 2 et sur le User B1.

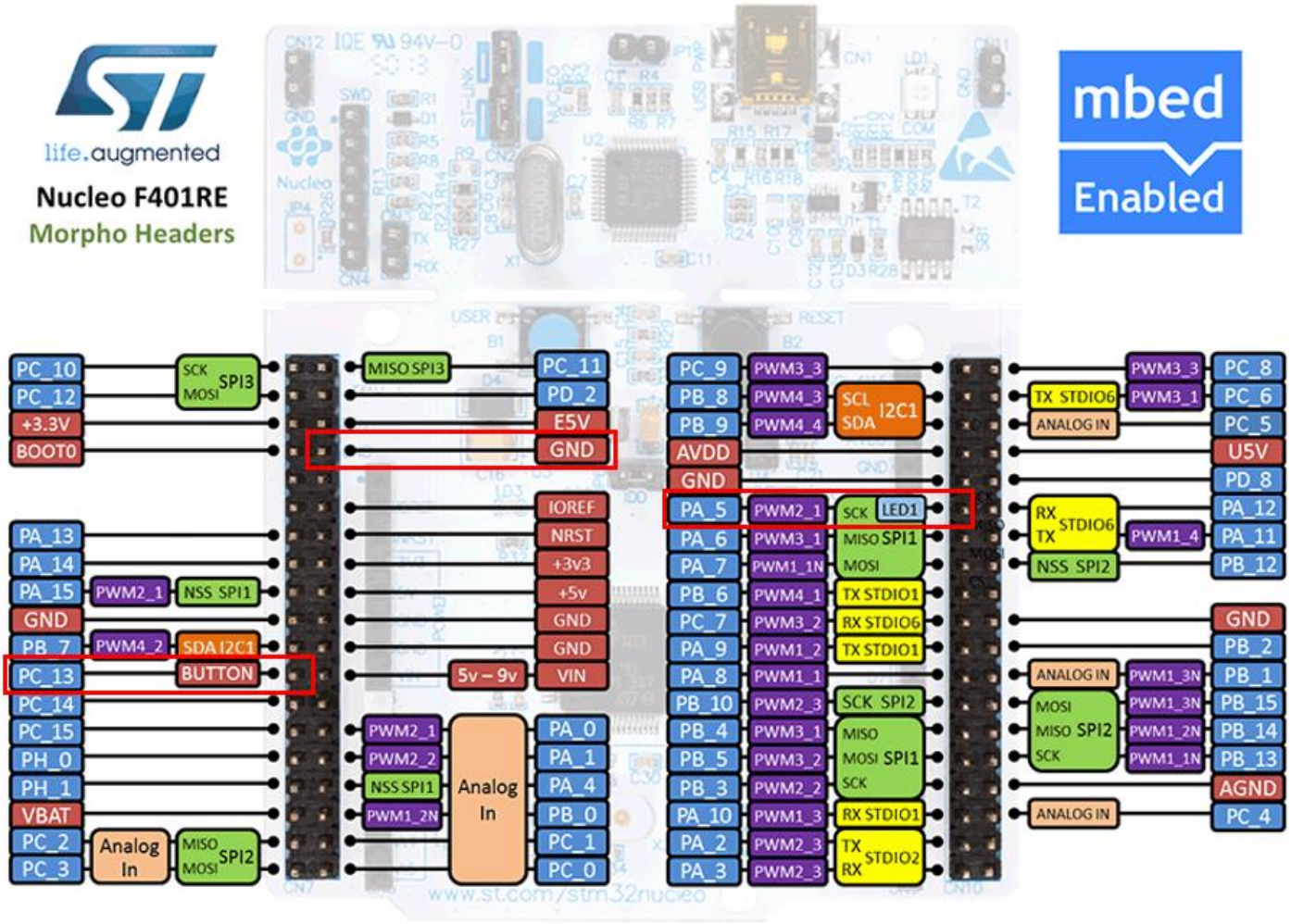
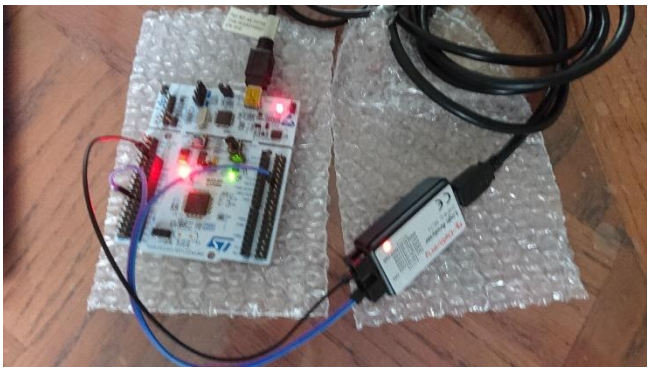
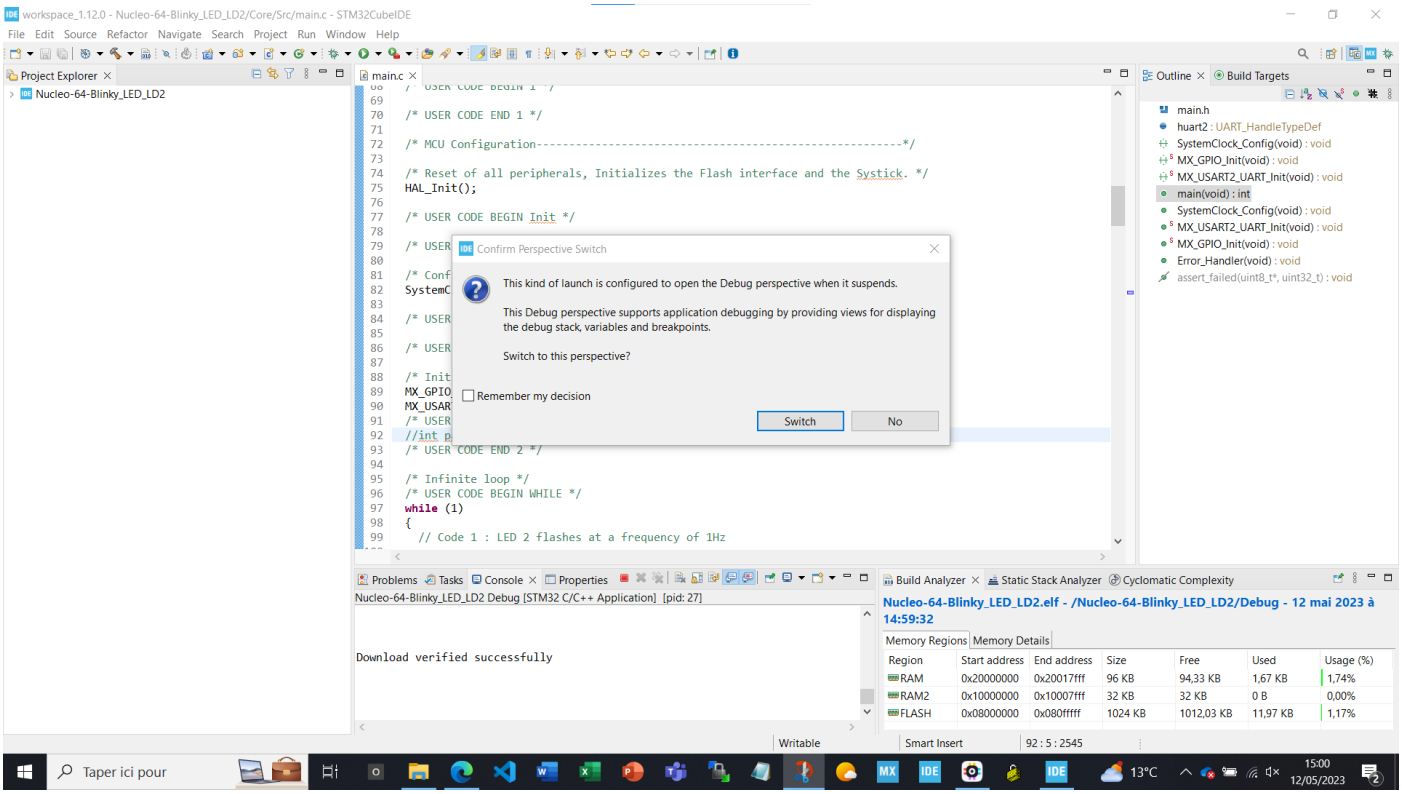


Figure 1. Pinout Nucleo-64.



Lancement du mode débugge avec un code simple :



Vérification via le Logic Analyzer :

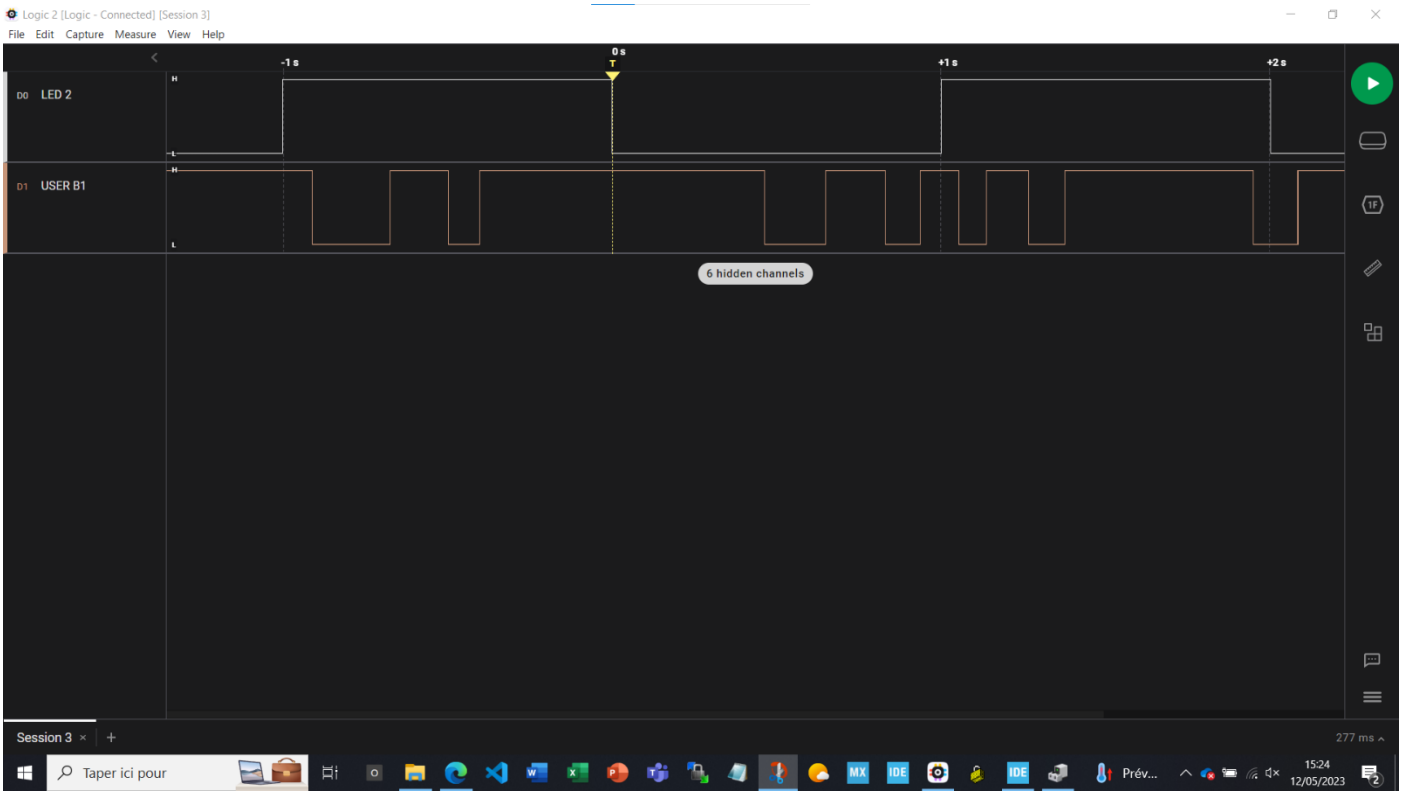


Figure 2. Analyse 1 : Test sur 3 secondes mode trigger sur front descendant.

Sur le signal LED 2, on remarque un changement d'état toutes les secondes. C'est l'objectif du code donc on est bon.

Sur le signal User B1, on remarque un front descendant quand j'appuis sur le bouton (7 appuis).

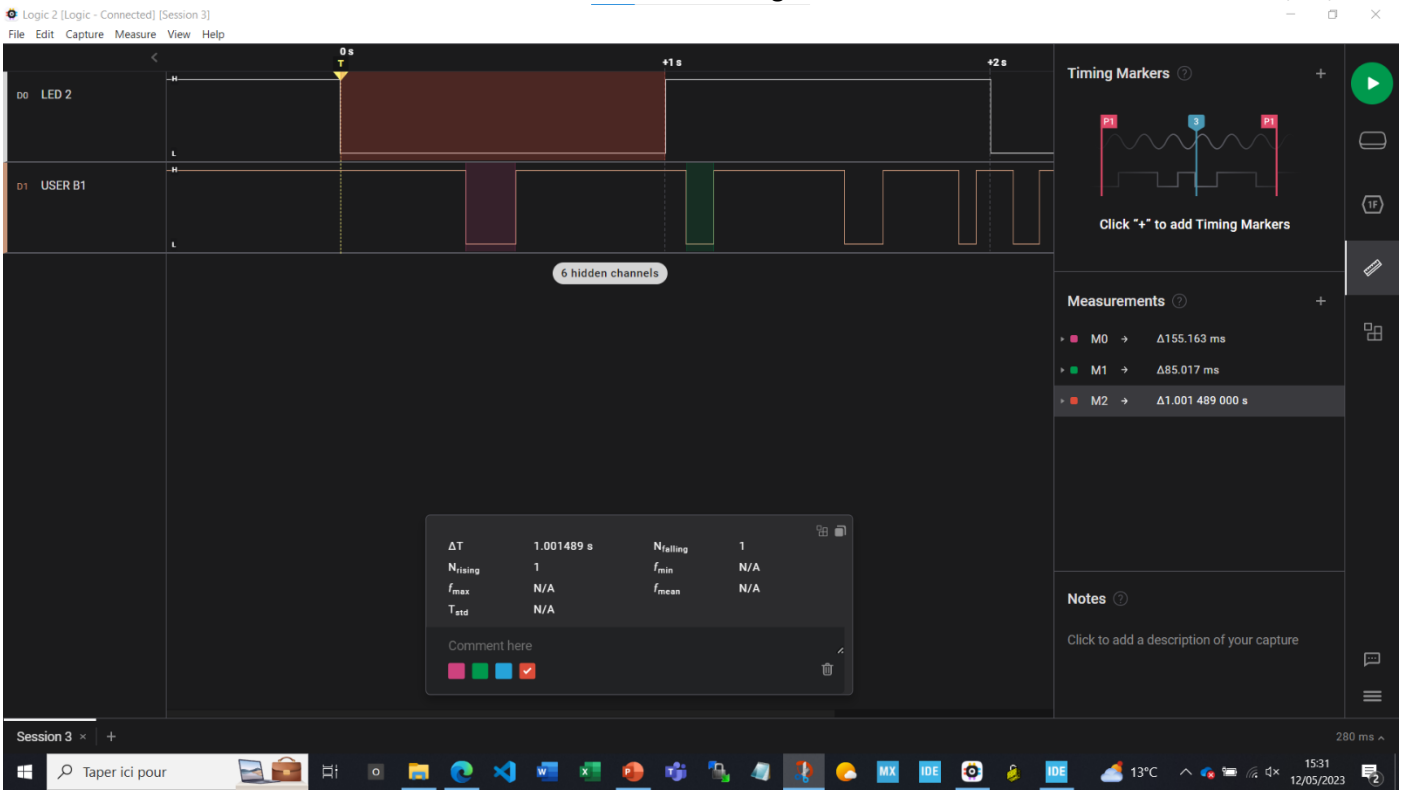


Figure 3. Analyse 2 : Test des fonctions de mesure.

D'après la mesure du Logic Analyser, la première période du signal LED 2 à durée 1,001489 s.

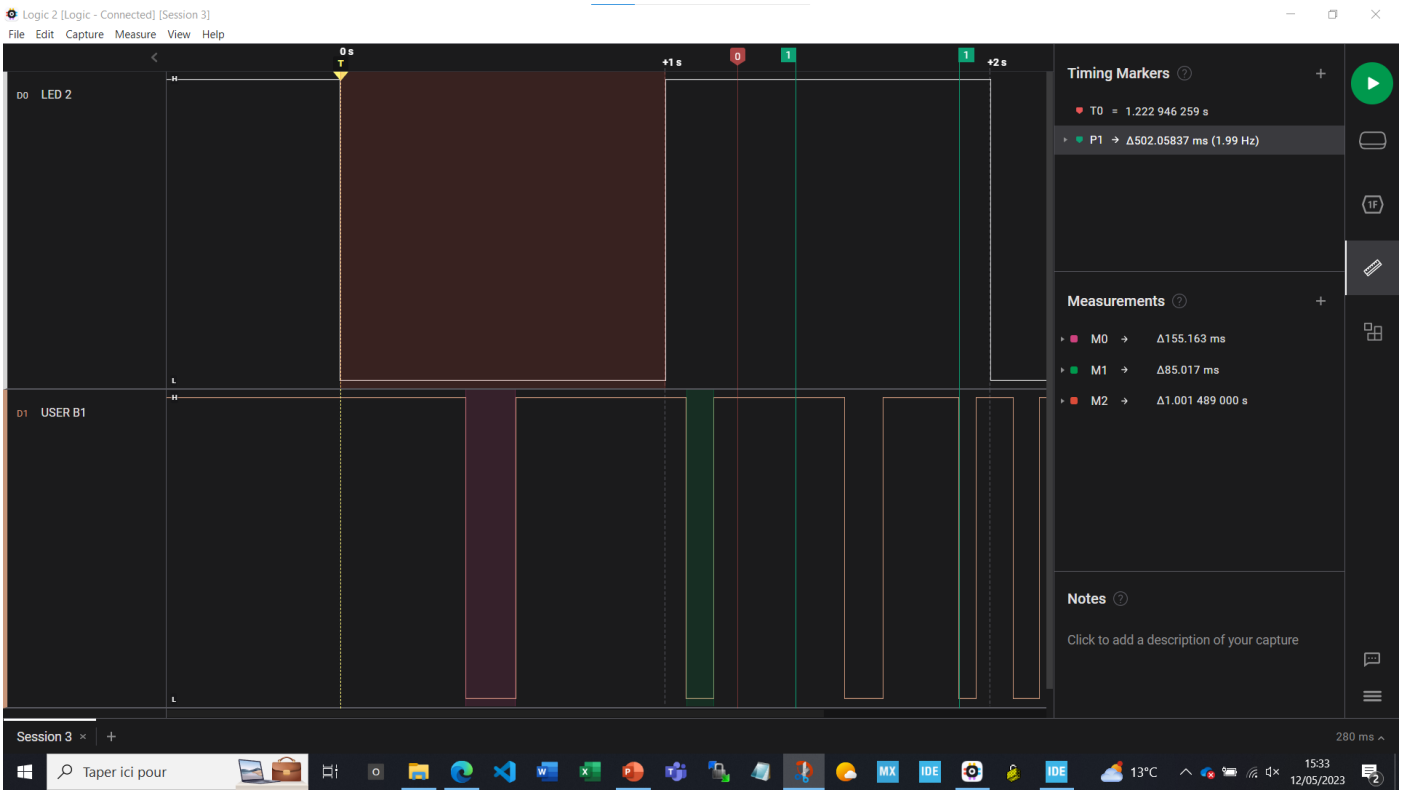


Figure 4. Analyse 3 : Test des fonctions de marqueur temporel.

J'ai placé un marqueur seul à 1,222946259 s après le T0 (trigger sur front descendant).

Il s'est écoulé 502,05837 ms entre mes deux marqueurs appairés.

Test fréquence d'exécution avec un Toggle LED2 :

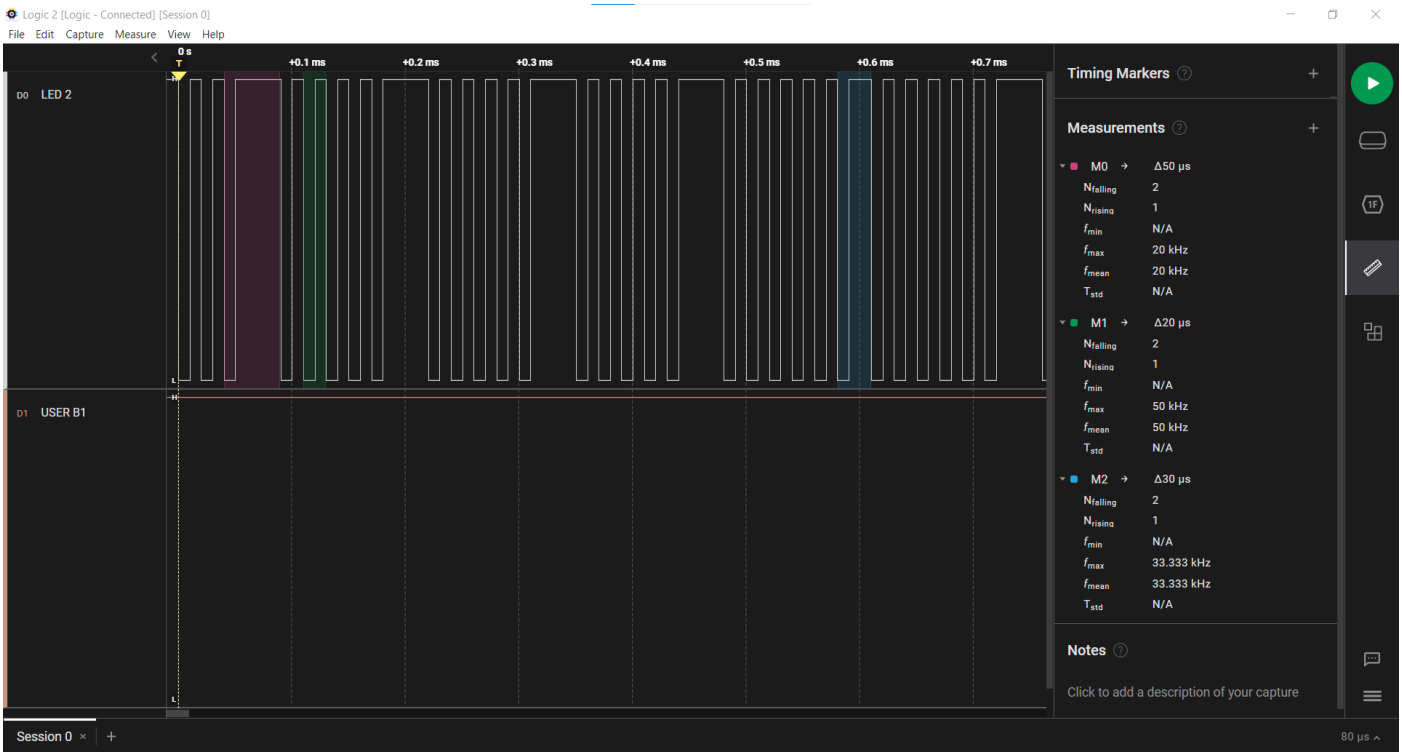


Figure 5. Mesure du GPIO PA5 LED2.

J'ai des périodes qui varient de 20µs à 50µs. C'est un signal pas très propre.

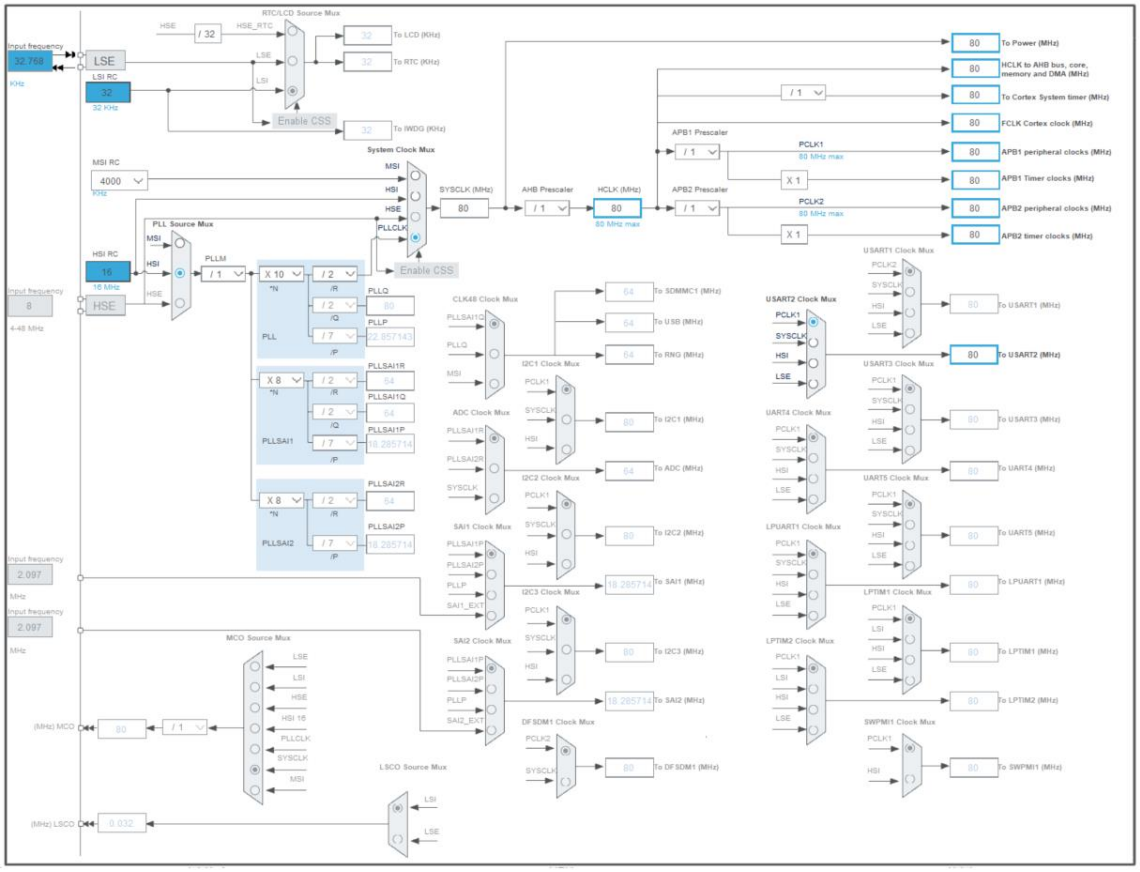


Figure 6. Programmation de l'horloge.

Test avec le Timer 16 :

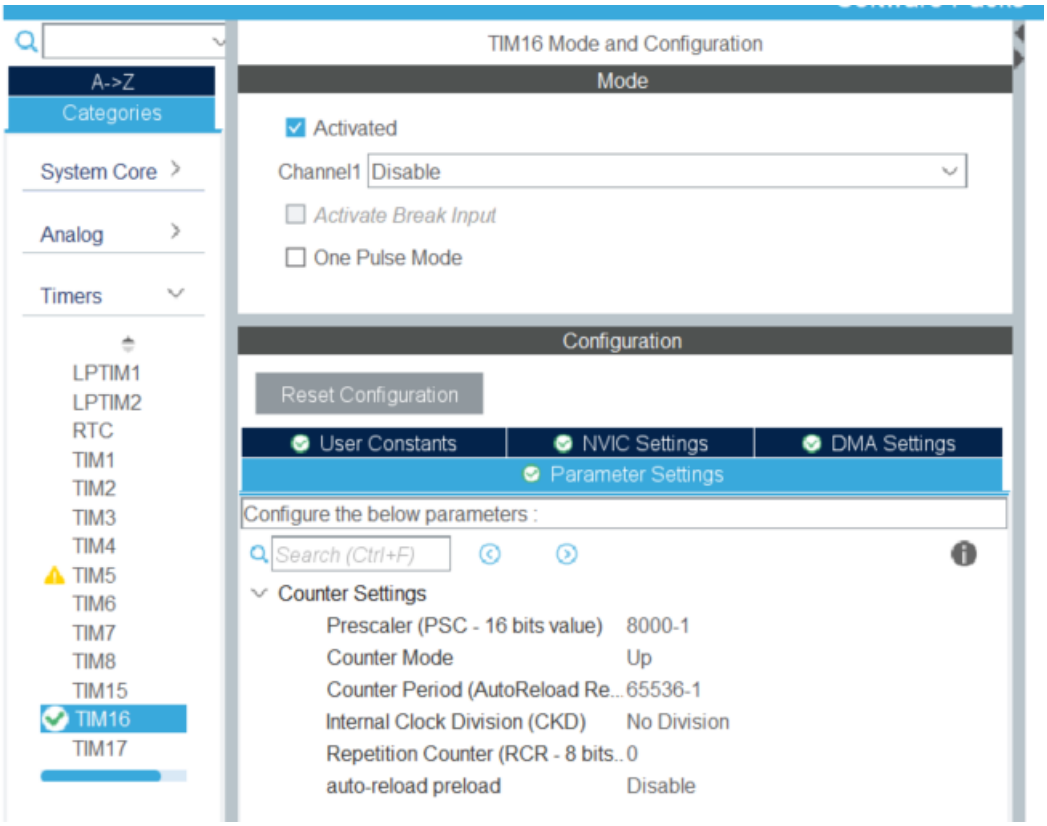


Figure 7. Configuration Timer16.

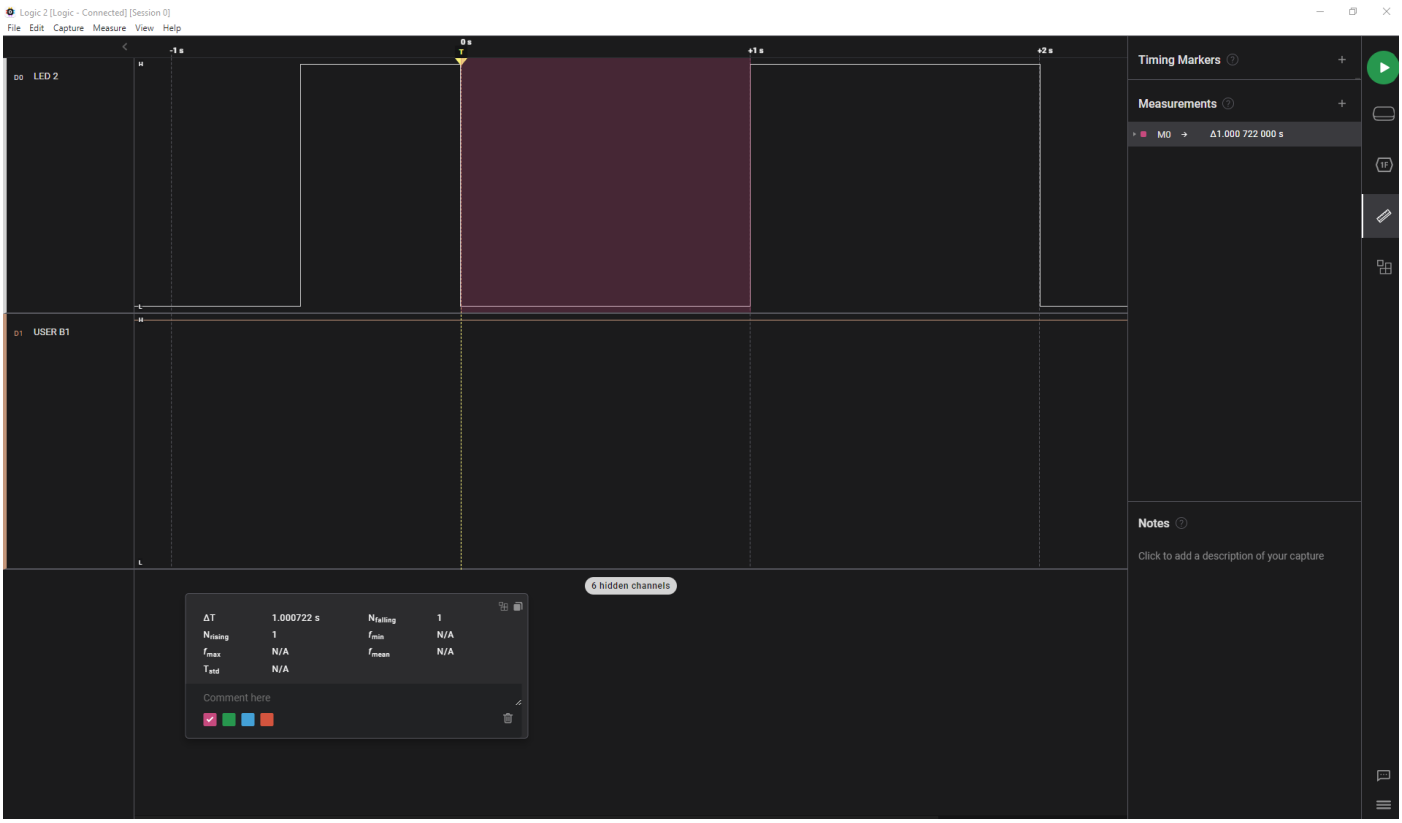


Figure 8. Mesure de la durée du comptage avant changement d'état.

Affinage des réglages du Timer16 :

```
182 static void MX_TIM16_Init(void)
183 {
184
185   /* USER CODE BEGIN TIM16_Init 0 */
186
187   /* USER CODE END TIM16_Init 0 */
188
189   /* USER CODE BEGIN TIM16_Init 1 */
190
191   /* USER CODE END TIM16_Init 1 */
192   htim16.Instance = TIM16;
193   htim16.Init.Prescaler = 4000-1;
194   htim16.Init.CounterMode = TIM_COUNTERMODE_UP;
195   htim16.Init.Period = 10000-9;
196   htim16.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
197   htim16.Init.RepetitionCounter = 0;
198   htim16.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
199   if (HAL_TIM_Base_Init(&htim16) != HAL_OK)
200   {
201     Error_Handler();
202   }
203   /* USER CODE BEGIN TIM16_Init 2 */
204
205   /* USER CODE END TIM16_Init 2 */
206
207 }
```

Figure 9. Modification de la valeur de la période (limite du compteur).

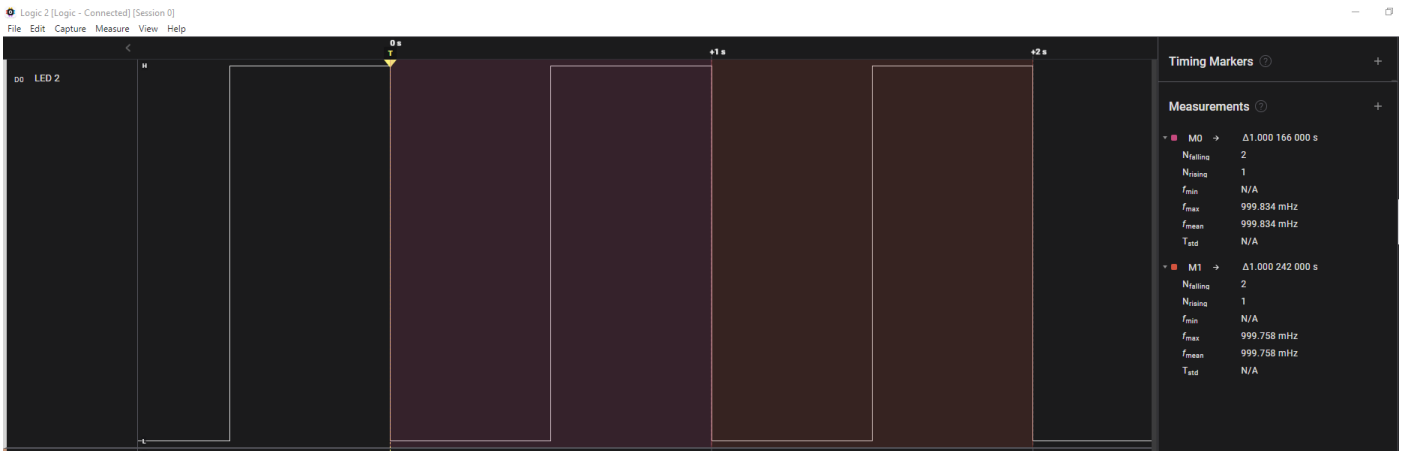


Figure 10. Mesure de la fréquence de clignotement de la LED2.

Test de l’interruption sur le Bouton B1 (Toggle LED2) :

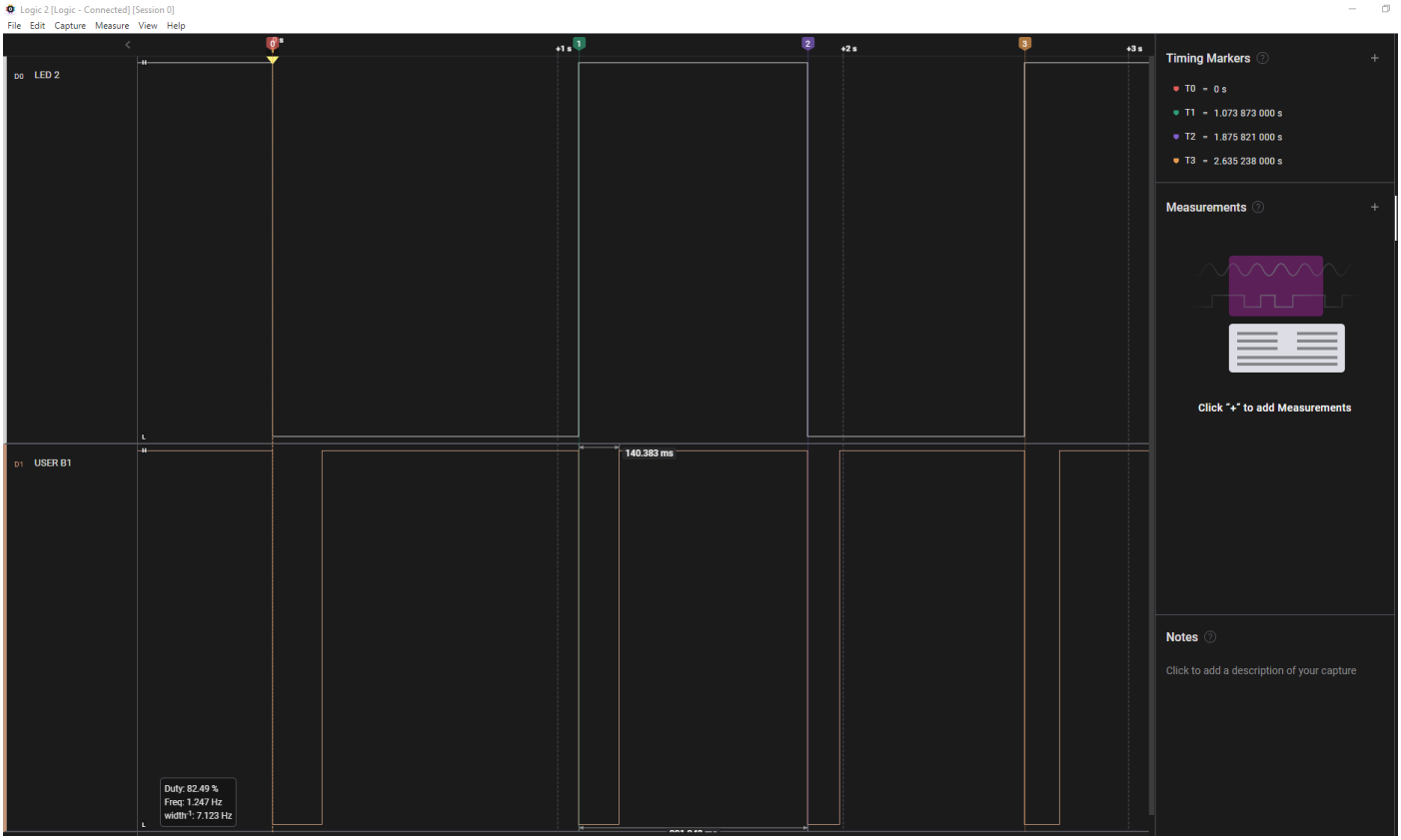


Figure 11. Mesure suite à l'utilisation du bouton B1.

A chaque fois qu’il y a un front descendant, je change la valeur du Pin de la LED2.

J’ai appuyé à 4 reprises. Il y a eu 4 changements d’état.

Test double interruption TIMER16 et Bouton :

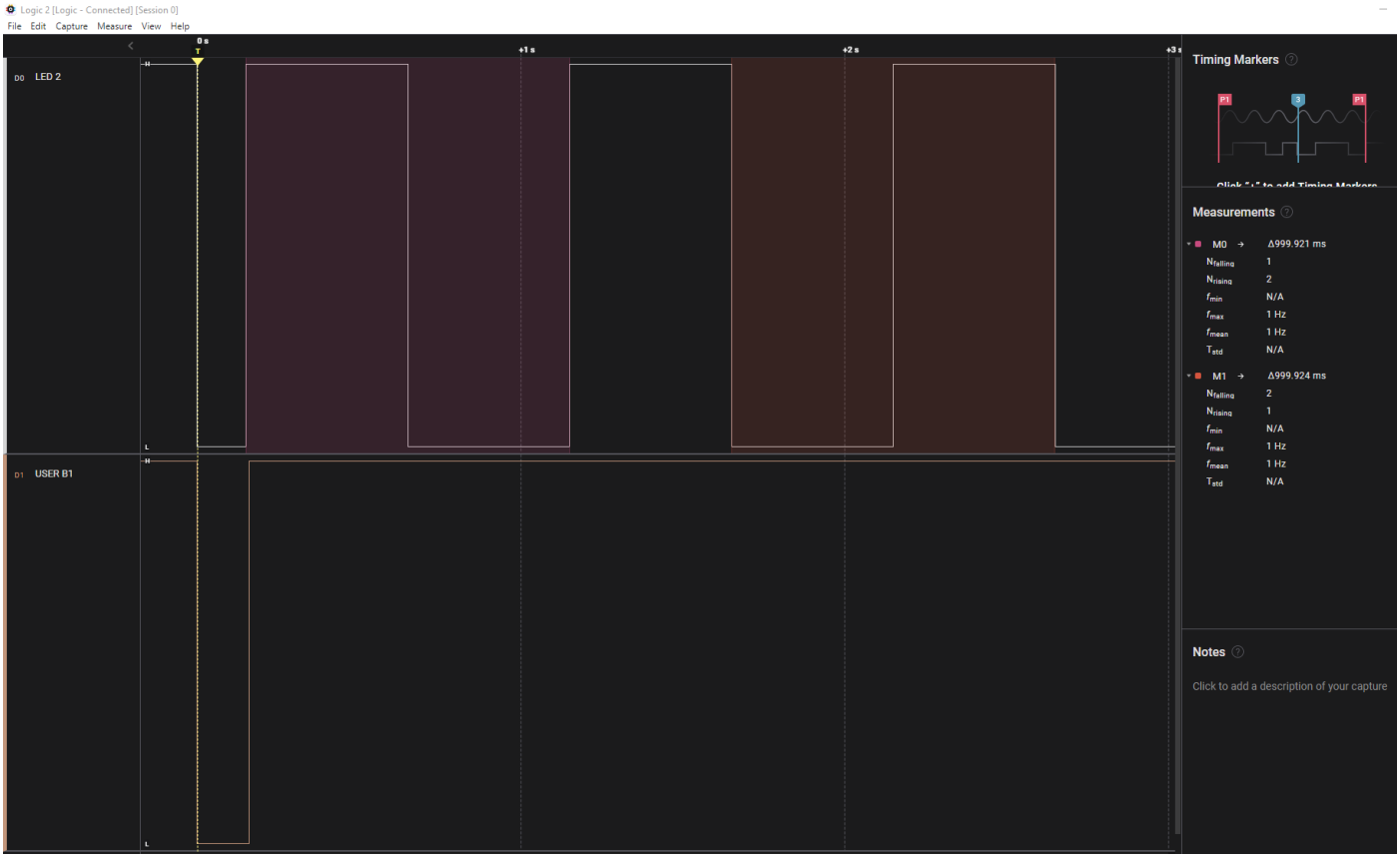
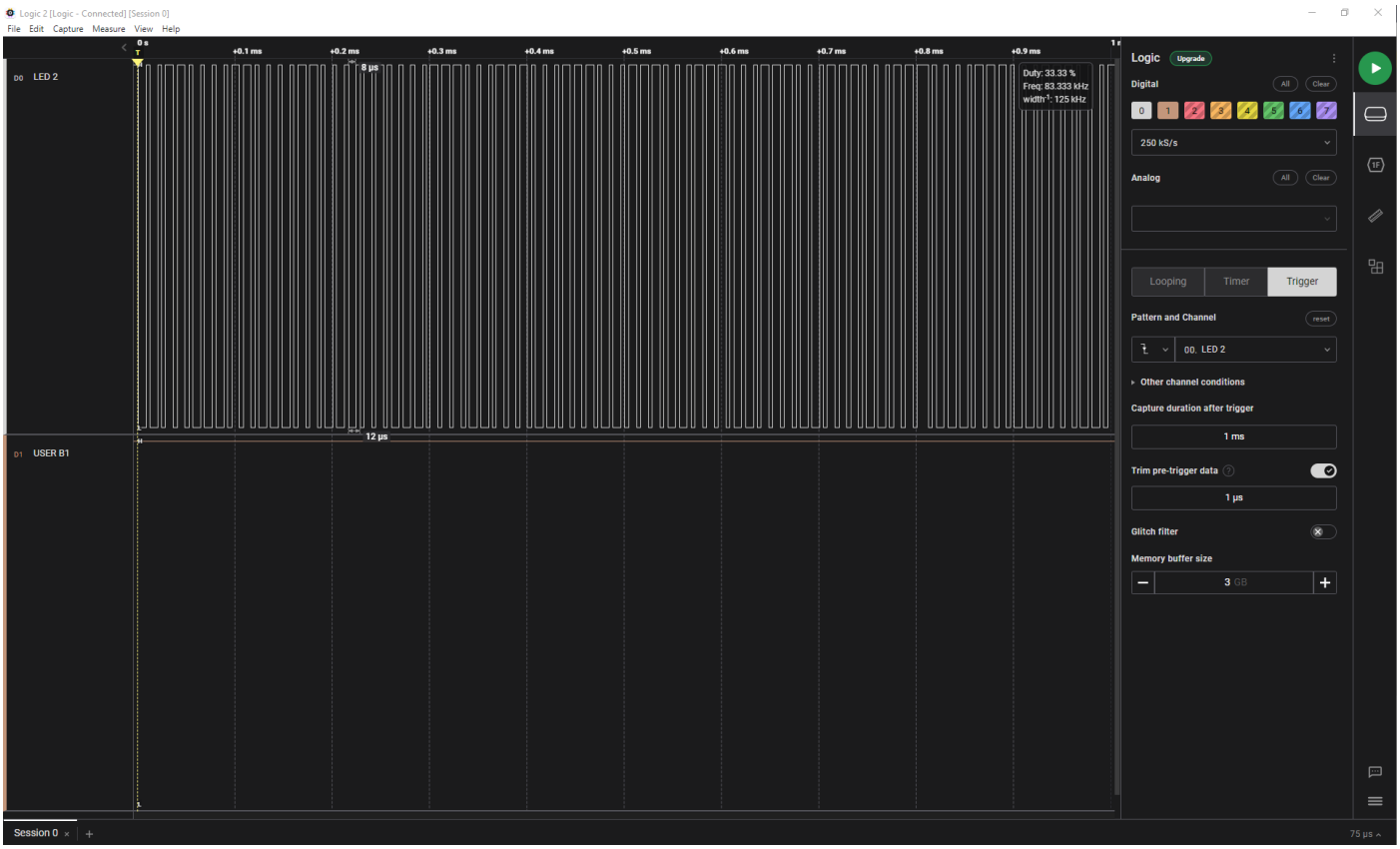


Figure 12. Mesure en sortie du Bouton B1 et de la LED2.

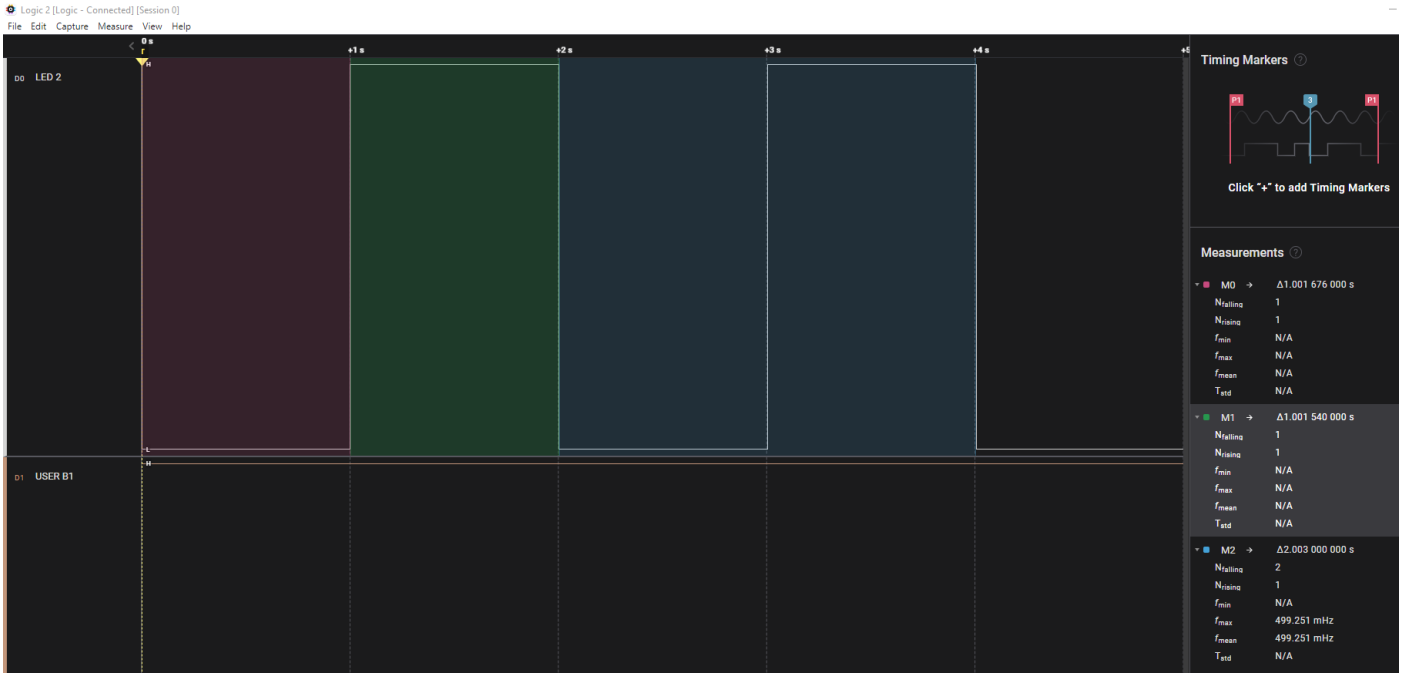
J'ai appuyé une fois sur le bouton pour passer en mode Toggle. Le principe du mode Toggle est de faire un changement d'état de la LED2 toutes les 500 ms pour avoir une période de 1 Hz. La temporisation est géré en interruption via le Timer16.

Test du Code_0 après réécriture Lib :



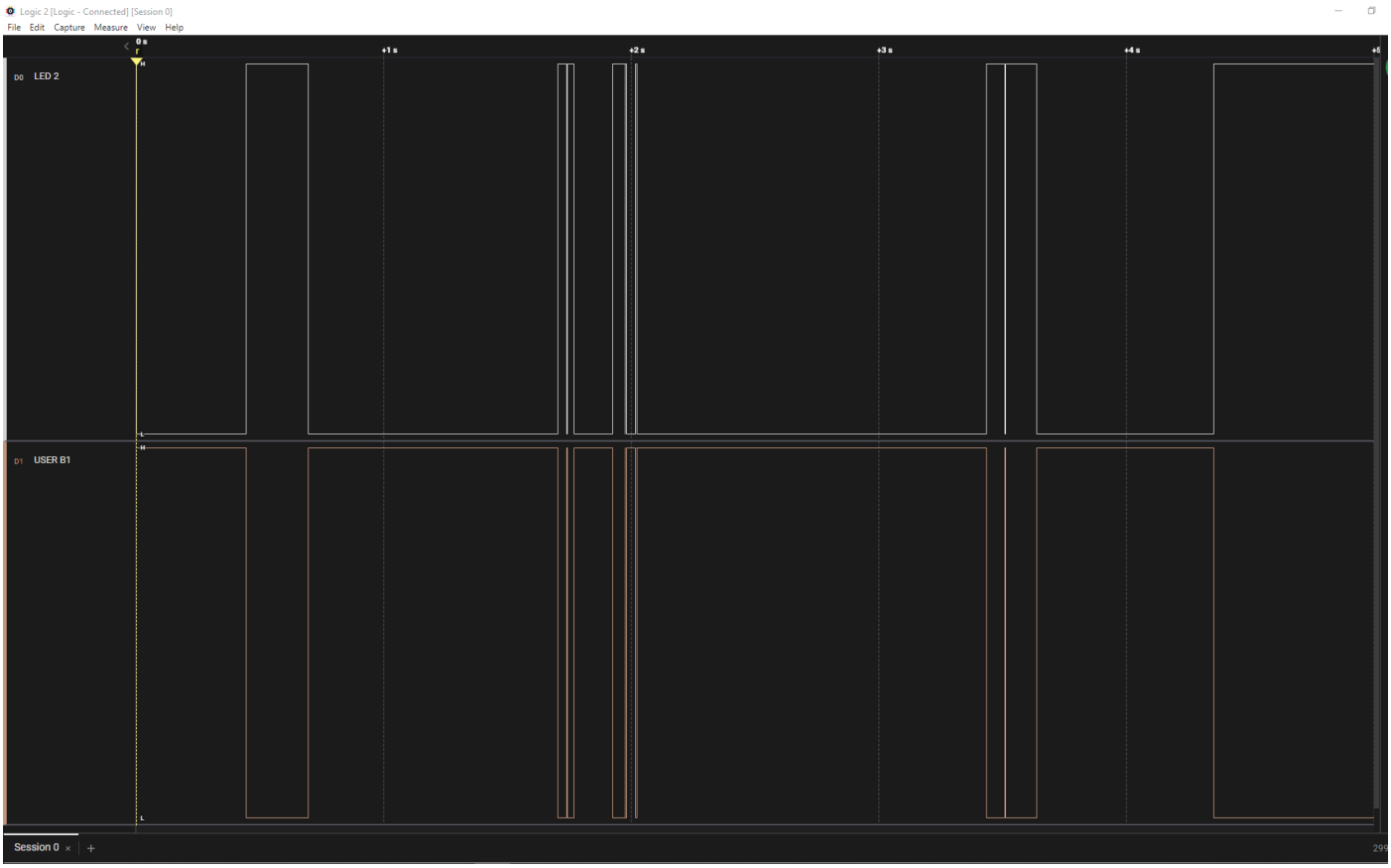
Même sur j'ai échantillonné au maximum (250 kS/s), je suis en limite de captation pour le toggle avec une fréquence de CPU de 80 MHz.

Test du Code_1 après réécriture Lib :



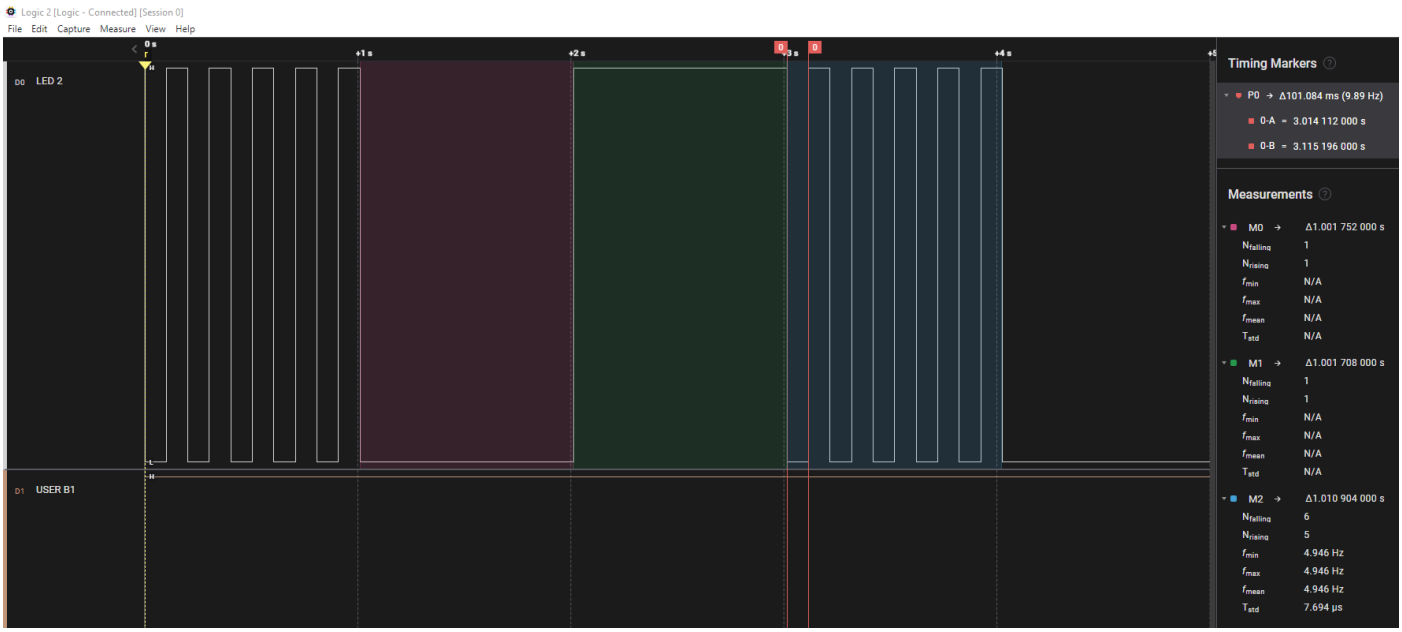
J'ai bien un intervalle d'environ 1s entre chaque changement d'état.

Test du Code_2 après réécriture Lib :



J'ai bien une réaction à chaque fois que j'appuie sur le bouton B1. Pour rappel, quand on appuie le bouton B1 renvoie un 0 et la LED2 s'allume (état Haut).

Test du Code_3 après réécriture Lib :

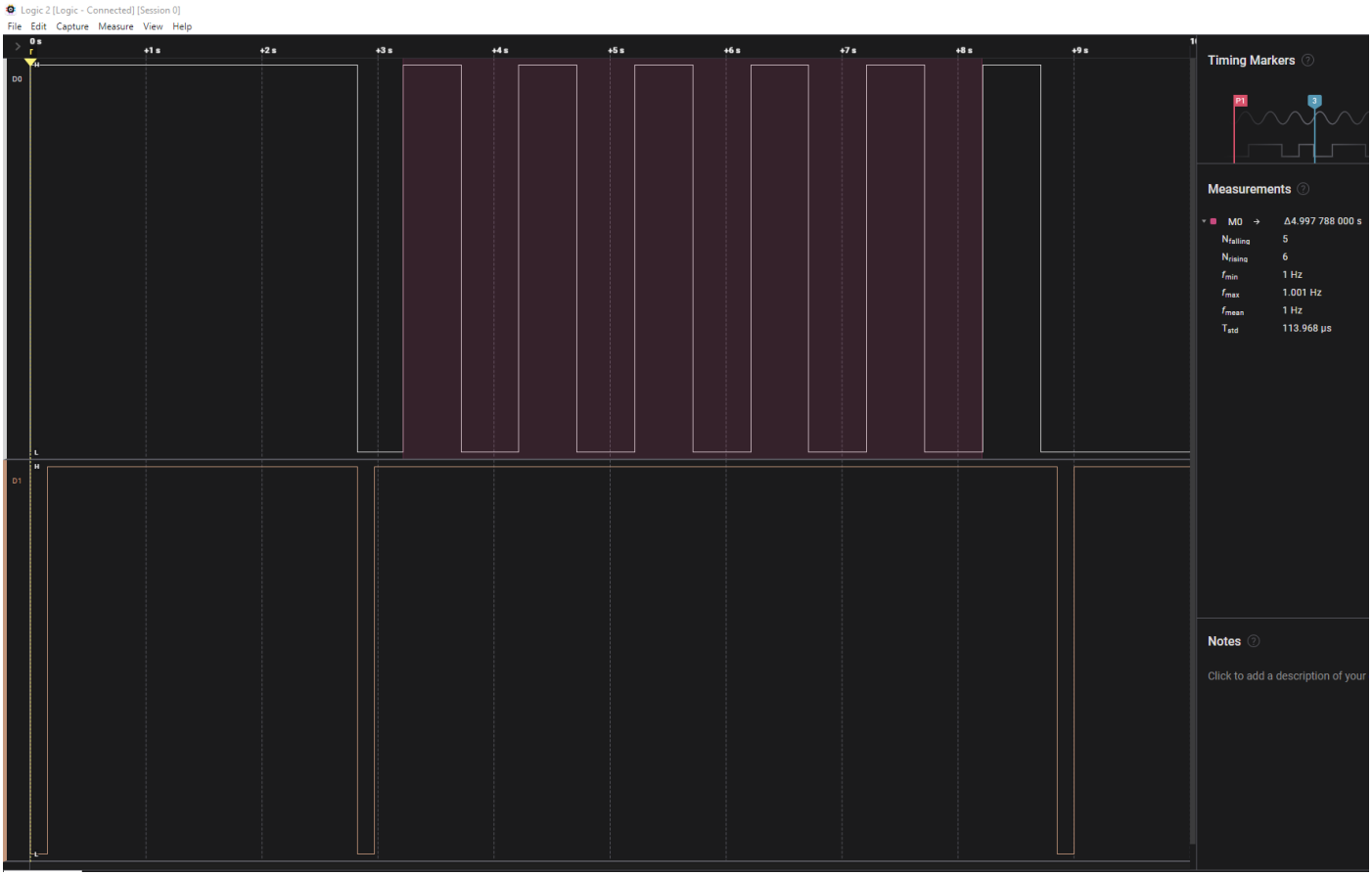


J'avais une première erreur à cause d'une mauvaise manipulation de pointer. J'incrémentais l'adresse et non la valeur. J'ai corrigé ce point-là et j'observe bien 1 second OFF, 1 second ON et 1 second d'oscillation de 5 périodes.

Test du Code_4 après réécriture Lib :

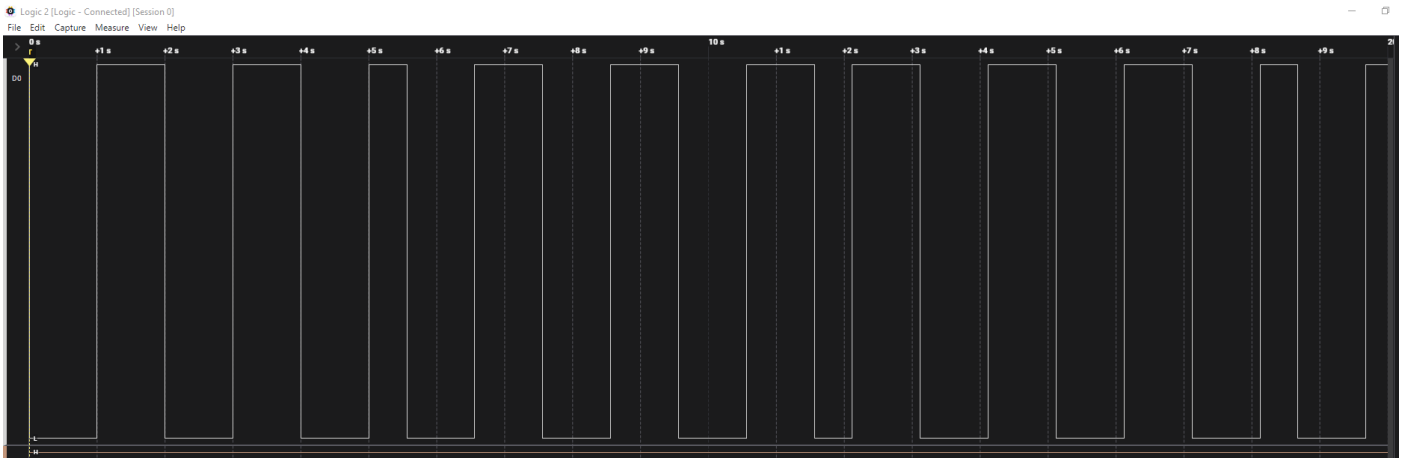
Ce code fonctionne mais il a ces limites. On le voit lors du passage de l'état 2 (toggle) à l'état 0 (off). Au 3ième appui de bouton B1, cet appuie n'a pas été lu. Il a fallu un 4ième appui surement au bon moment pour être pris en compte.

Test du Code_5 après réécriture Lib :

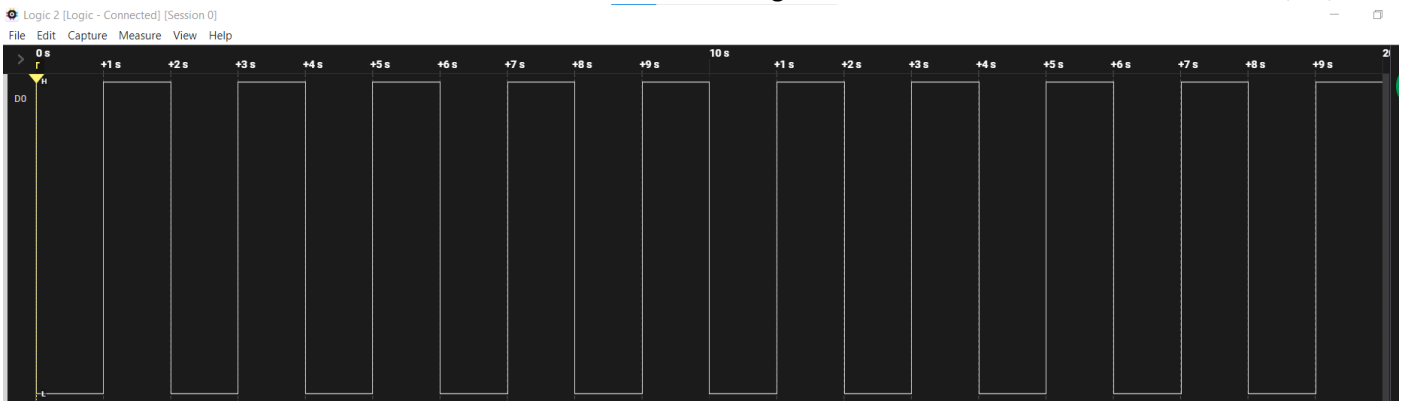


Le code 5 fonctionne correctement. Après chaque interruption, on change d'état. La fréquence de toggle est de 1 Hz comme souhaité.

Test du Code Timer après réécriture Lib :



Il y a un bug tous les 6 temps.



Après investigation, j'ai remarqué qu'il n'y avait pas du Reset du Timer. Le Timer compte jusqu'à 65535. Si on divise par 10000 on trouve 6 fois plus 5535 qui est le reste de la division euclidienne. Ces 5535 étaient mesurés car tous les 6 temps on avait une période de 553 ms. J'ai donc ajouté un reset du compteur après le Toggle pour corriger le bug et de compter seulement de 0 à 10000.