

DM tutoriel Timers

EI2I3-IIA

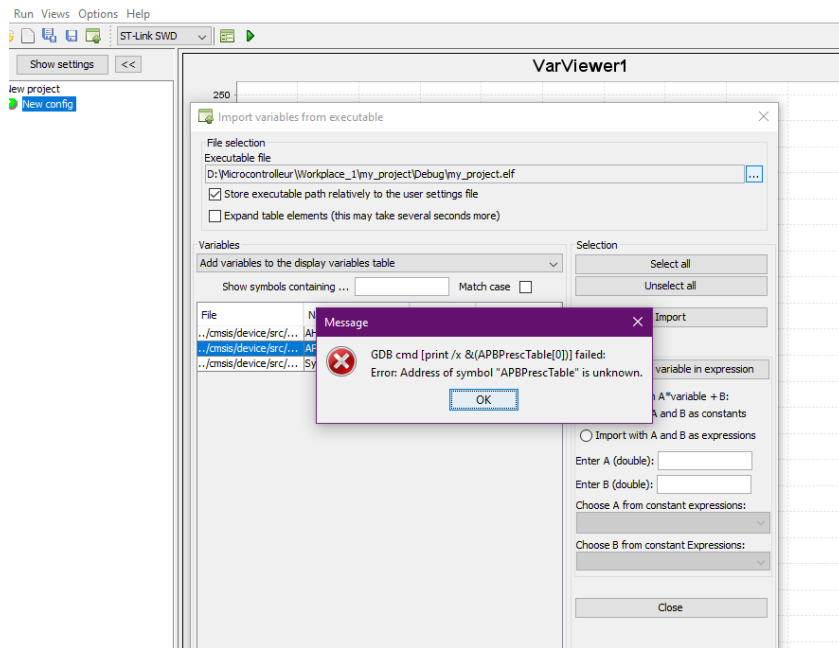
Julian BARKOUDEH

Note 1 : Tous les codes de BSP.c relatifs à ce tutoriel se trouvent dans la partie Timer, à partir de la ligne 340.

Note 2 : La fonction de printf est appelée j_printf dans mes codes. Or dans le rendu de chaque main.c je les avait appelés mon_printf

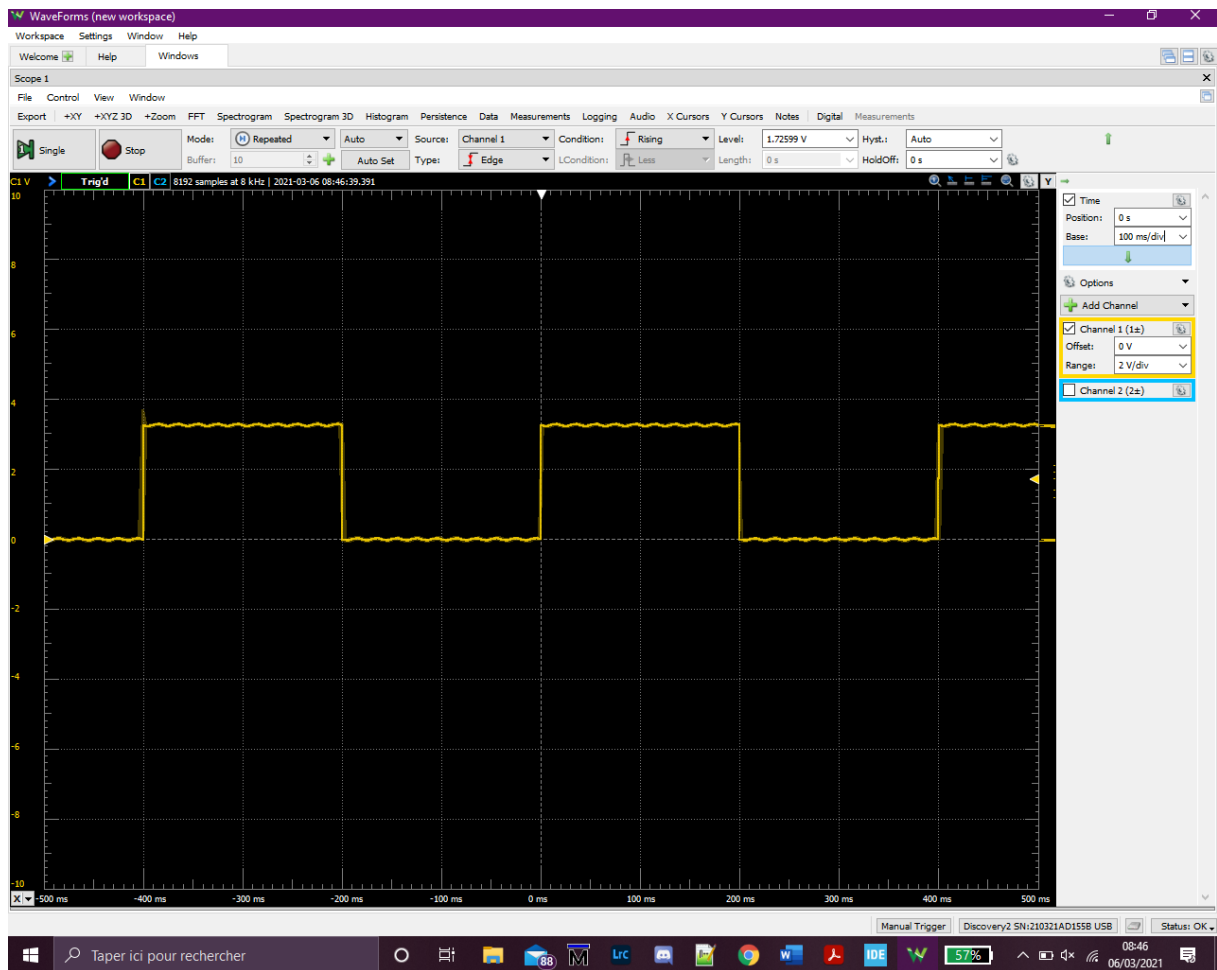
1.1 Base de temps simple : Mutineries

```
1 /*
2  * main.c
3  * Created on: Feb 3, 2021
4  * Author: BARKOUDEH */
5
6 #include "stm32f0xx.h"
7 #include "bsp.h"
8 #include "delay.h"
9 #include "main.h"
10 static void SystemClock_Config(void);
11
12 // Main program
13 int main()
14 {
15     // Configuration horloge du systeme
16     SystemClock_Config();
17     //Initialisation signal PWM
18     BSP_LED_Init();
19     BSP_TIMER_Timebase_Init();
20     uint8_t TIM6_counter;
21     uint8_t TIM6_UIF;
22     while(1){
23         TIM6_counter = TIM6->CNT;
24         if(TIM6->SR == 1){
25             if(TIM6_UIF == 0){
26                 TIM6_UIF = 1024;
27             }
28             else if(TIM6_UIF == 1024){
29                 TIM6_UIF = 0;
30             }
31         }
32         TIM6->SR = 0;
33     }
34 }
```



Le code ci-dessus permet de générer le signal en question. Or l'erreur suivant empêche de visualiser les variables sur STMstudio.

1.2) Utilisation du Timer pour gérer une fonction de temporisation robuste



On branche l'oscilloscope sur le pin PA5 lié au LED afin de visualiser le basculement avec les fonctions delay introduites.

2.1 Mode Input Capture : Single edge – détection d'un seul front

The screenshot displays an IDE environment with the following components:

- Source Code (main.c):**

```
1 /*  
2  * main.c  
3  * Created on: Feb 3, 2021  
4  * Author: darga */  
5  
6 #include "stm32f0xx.h"  
7 #include "bsp.h"  
8 #include "delay.h"  
9 #include "main.h"  
10 static void SystemClock_Config(void);  
11  
12 // Main program  
13 int main()  
14 {  
15     // Configuration horloge du  
16     SystemClock_Config();  
17     // Initialisation Pin pour L  
18     BSP_LED_Init();  
19     // Initialisation de la cons  
20     BSP_Console_Init();  
21     j_printf("La Console est Rea  
22     // Initialisation et lancem  
23     // pour faire un delai  
24     BSP_DELAY_TIM_init();  
25     // Initialisation du TIM3 en  
26     // la patte PB4 est egalement  
27     BSP_TIMER_IC_Init();  
28     // Boucle infinie  
29     while(1) {  
30         // Affichage du status de  
31         j_printf("CNT->%05d Fall  
32         // Attente 100 ms avec  
33         // Timer TIM6  
34         BSP_DELAY_TIM_ms(100);  
35     }  
36 }  
37 */  
38 * Clock configuration for the N  
39 * HSE input Bypass Mode -> 8MHz  
40 * SYSCLK, AHB, APB1 -> 48MHz  
41 * D4R as MCO with /16 prescaler
```
- COM3 - PuTTY:** A terminal window showing the output of the program:

```
La Console est Ready!  
La Console est Ready!  
La Console est Ready!  
CNT->53920 Fall->51741
```
- Debugger Console:** Shows the status of the program execution:

```
Console  
project [STM32 Cortex-M C/C++ Application]  
  
ifing ...  
  
mload verified successfully
```

The bottom of the image shows the Windows taskbar with the system clock at 08:53 on 06/03/2021 and a battery level of 53%.

2.2 Mode Input Capture : Dual edges captures – détection de deux fronts

The screenshot displays an IDE environment with the following components:

- Code Editor:** Contains C code for an STM32F0xx application. The code includes headers for the STM32F0xx, BSP, delay, and main. It defines a `SystemClock_Config` function and a `main` function. The `main` function initializes the system clock, LED, console, and timer, then enters an infinite loop where it prints the TIM3 status (CNT, CCR1, and CCR2) and delays for 100ms.
- COM3 - PuTTY:** A terminal window showing the output of the program. It displays "La Console est Ready!" followed by several "La Console est Ready!0" and "La Console est Ready!1" messages. It also shows the output of the timer status: "CNT->08649 Fall->05817 Rise->05904".
- Console:** A window showing the output of the program. It displays "Verifying ..." and "Download verified successfully".

```
1 /*  
2  * main.c  
3  * Created on: Feb 3, 2021  
4  * Author: darga */  
5  
6  #include "stm32f0xx.h"  
7  #include "bsp.h"  
8  #include "delay.h"  
9  #include "main.h"  
10 static void SystemClock_Config(void);  
11  
12 // Main program  
13 int main()  
14 {  
15     // Configuration horloge du systeme  
16     SystemClock_Config();  
17     // Initialisation Pin pour LED verte  
18     BSP_LED_Init();  
19     // initialisation de la console  
20     BSP_Console_Init();  
21     j_printf("La Console est Ready!\r\n");  
22     // Initialisation et lancement du Timer  
23     // pour faire un delai  
24     BSP_DELAY_TIM_init();  
25     // Initialisation du TIM3 en mode Input  
26     // la patte PB4 est egalement initialise  
27     BSP_TIMER_IC_Init();  
28     // Boucle infinie  
29     while(1) {  
30         // Affichage du status de TIM3 (regi  
31         // Report TIM3 status (CNT, CCR1 and  
32         j_printf("CNT->%05d Fall->%05d Rise->%05d\r\n", TIM3->CNT, TIM3->CCR1, TIM3->CCR2);  
33         // Attente 100 ms avec un delai gen  
34         // Timer TIM6  
35         BSP_DELAY_TIM_ms(100);  
36     }  
37 }  
38 */  
39 * Clock configuration for the Nucleo STM32F072RB board  
40 * HSE input Bypass Mode -> 8MHz  
41 * SYSCLK DHR DPR1 -> D8MHz
```

COM3 - PuTTY

```
La Console est Ready!  
La Console est Ready!0  
La Console est Ready!0  
La Console est Ready!1  
La Console est Ready!0  
La Console est Ready!0 Rise->00000  
CNT->08649 Fall->05817 Rise->05904
```

Console

```
Verifying ...  
  
Download verified successfully
```

Writable Smart Insert 16:1:258 08:57 06/03/2021

2.3 Mode Input Capture : Dual edges captures – détection de deux fronts avec reset automatique

The screenshot displays an IDE environment with a C source file and a terminal window.

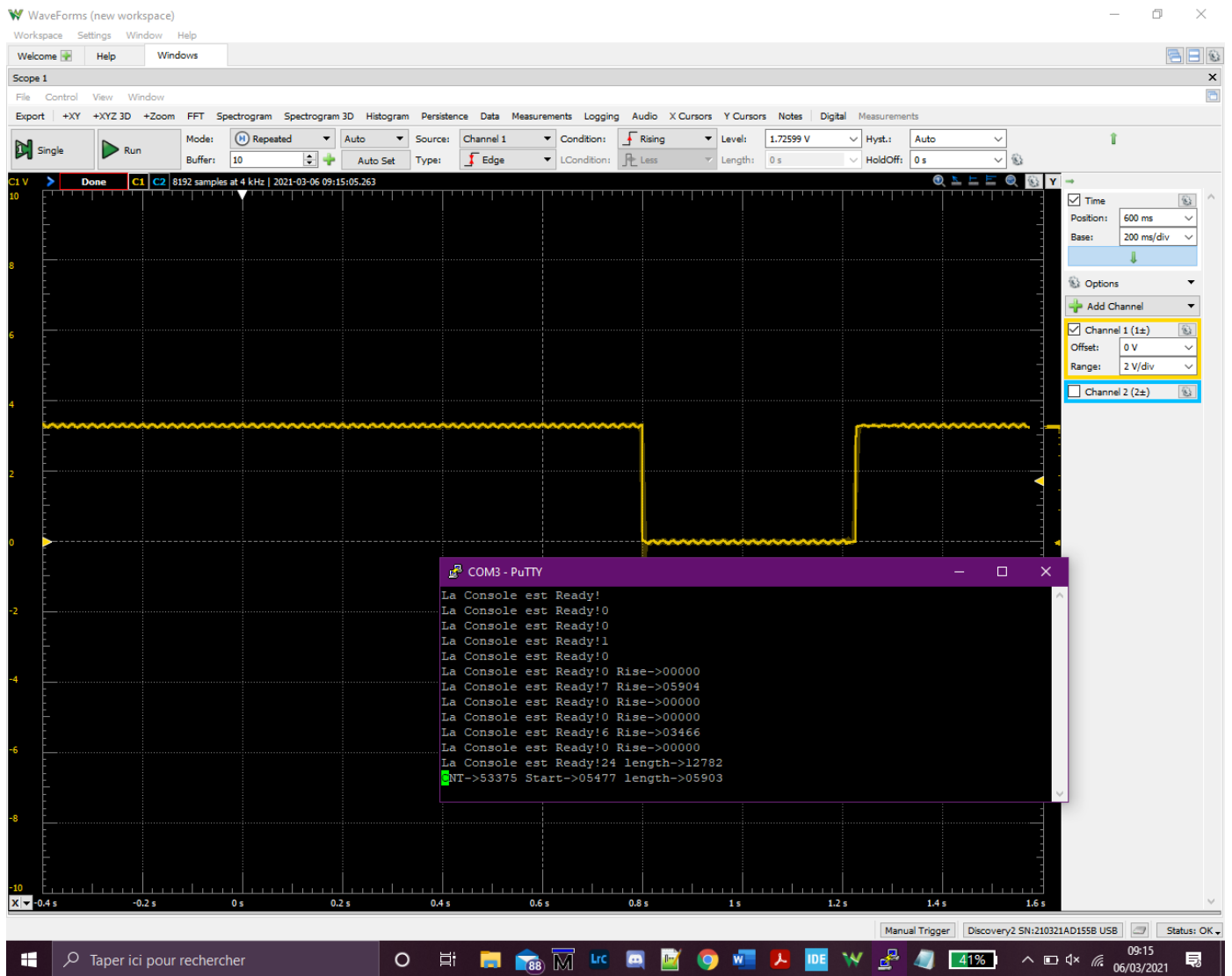
Source Code (main.c):

```
1 /*
2  * main.c
3  * Created on: Feb 3, 2021
4  * Author: darga */
5
6 #include "stm32f0xx.h"
7 #include "bsp.h"
8 #include "delay.h"
9 #include "main.h"
10 static void SystemClock_Config(void);
11
12 // Main program
13 int main()
14 {
15     // Configuration horloge du systeme
16     SystemClock_Config();
17     // Initialisation Pin pour LED verte
18     BSP_LED_Init();
19     // initialisation de la console
20     BSP_Console_Init();
21     j_printf("La Console est Ready!\r\n");
22     // Initialisation et lancement du Timer
23     // pour faire un delai
24     BSP_DELAY_TIM_init();
25     // Initialisation du TIM3 en mode Input
26     // la patte PB4 est egalement initialise
27     BSP_TIMER_IC_Init();
28     // Boucle infinie
29     while(1) {
30         // Affichage du status de TIM3 (regi
31         // Report TIM3 status (CNT, CCR1 and
32         j_printf("CNT->%05d Start->%05d leng
33         // Attente 100 ms avec un delai gen
34         // Timer TIM6
35         BSP_DELAY_TIM_ms(100);
36     }
37 }
38 */
39 * Clock configuration for the Nucleo STM32F072RB board
40 * HSE input Bypass Mode -> 8MHz
41 * SYSCLK_APR1 -> 48MHz
```

Terminal Output (COM3 - PuTTY):

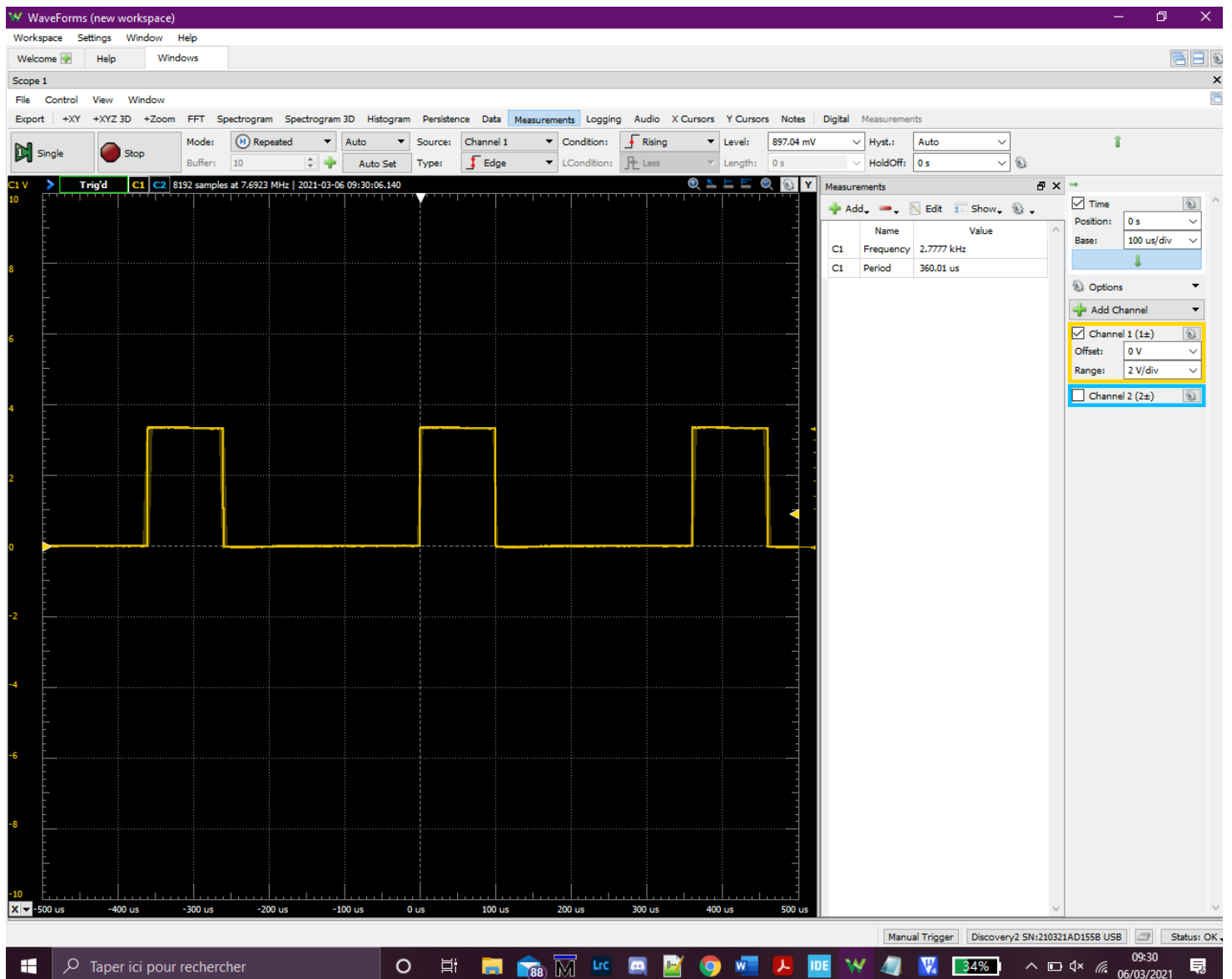
```
La Console est Ready!
La Console est Ready!0
La Console est Ready!0
La Console est Ready!1
La Console est Ready!0
La Console est Ready!0 Rise->00000
La Console est Ready!7 Rise->05904
La Console est Ready!0 Rise->00000
La Console est Ready!0 Rise->00000
La Console est Ready!0 Rise->00000
La Console est Ready!6 Rise->03466
La Console est Ready!0 Rise->00000
CNT->11458 Start->07878 length->08318
```

The IDE interface includes a Console window at the bottom showing the text "erifying ..." and "ownload verified successfully". The taskbar at the bottom shows the system clock as 09:01 on 06/03/2021 and a battery level of 50%.



Sur l’affichage de printf on mesure 0.59 s, ceci est cohérent avec la lecture graphique du temps du signal affiché sur l’oscilloscope. On mesure $t = 1.3 - 0.8 = 0.5$ s.

3.1 Qu'est-ce qu'un signal PWM



```

122 //-----Signal PWM-----
123 // BSP_PWM_Init()
124 //Fonction initialisation d'un signal PWM sur le pin PA9
125 void BSP_PWM_Init(){
126     RCC->AHBENR |= (1<<17);
127     /*
128     // Ecrire "10" sur les bits b11b10 du registre (GPIOA_MODER) (AF)
129     GPIOA->MODER |= (1<<17);
130     GPIOA->MODER &= ~(1<<16);*/
131     // Ecrire "10" sur les bits b11b10 du registre (GPIOA_MODER) (AF)
132     GPIOA->MODER |= (1<<19);
133     GPIOA->MODER &= ~(1<<18);
134     //Def de l'AF
135     //Ecrire "0010" (AF2) sur b3b2b1b0
136     GPIOA->AFR[1] &= ~(1<<3);
137     GPIOA->AFR[1] &= ~(1<<2);
138     GPIOA->AFR[1] |= (1<<1);
139     GPIOA->AFR[1] &= ~(1<<0);
140     //APB2 timer1
141     RCC->APB2ENR |= (1<<11);
142     //Configuration des caractéristiques du signal PWM
143     //Time on
144     TIM1->PSC = 47;
145     //Résolution
146     TIM1->ARR = 180;
147     // Rapport cyclique
148     TIM1->CCR1 = 50;
149     TIM1->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1PE;
150     TIM1->CCER |= TIM_CCER_CC1E;
151     TIM1->BDTR |= TIM_BDTR_MOE;
152     TIM1->CR1 |= TIM_CR1_CMS_0 | TIM_CR1_CEN;
153     TIM1->EGR |= TIM_EGR_UG;
154 }
155

```

En se basant sur la manuelle de la carte, le code ci-dessus présent dans BSP.c permet d'initialiser un signal PWM sur la pate PA8.

Tout d'abord on met la pate PA8 en mode Highspeed, ensuite on active le AF2 sur la pate PA8. On génère un signal PWM de rapport cyclique de 50%, et d'une fréquence de 2.7 KHZ. La configuration de la fréquence est faite avec le registre PSC, on le règle à 47 us qui définit la période du signal. On définit la résolution par le registre ARR, on le règle à 180 pour avoir 180 valeurs du rapport cyclique.