

```

1  #include "stm32f4xx_conf.h"
2  #include "stdio.h"
3  #include "delay.h"
4  #include "stm32f4xx_rng.h"
5  #include "stm32_ub_rng.h"
6  #include <stdbool.h>
7
8
9  void Delay (int ms);
10 void GPIO_Ini (void);
11 void TIM_Ini1(void);
12 void TIM_Ini2(void);
13
14 int main(void)
15 {
16
17     int Vcounter = 0, countold = 0, var = 0;
18     int returntimer;
19
20     SystemInit();
21     GPIO_Ini();
22     TIM_Ini1();
23     TIM_Ini2();
24
25     returntimer = SysTick_Config(SystemCoreClock / 10000);
26
27     if (returntimer != 0)
28     {
29         GPIO_SetBits(GPIOD, GPIO_Pin_14);
30     }
31
32
33 while(1)
34 {
35
36     while(GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_3) == 1)
37     {
38
39         if(GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_0) == 1)
40         {
41             delay_100us(1);
42
43             if(GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_0) == 0)
44             {
45                 TIM_SetCounter(TIM4, 0);
46             }
47
48             if(GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_1) == 0)
49             {
50                 delay_100us(1);
51

```

```

//Initialisierung des STM32-Board
//Initialisierung der GPIO-Ports
//Initialisierung Timer 4
//Initialisierung Timer 2

//Wartezeit auf definieren

//Prüfen ob returntimer == 0

//Falls Fehler bei der Initialisierung
//der Wartezeit aufgetreten

//Main-Schleife

//Abfrage von /Attract

//Abfrage /VBLANK = high?

//Wartezeit für Taktflankenermittlung

//Bei negativer Taktflanke /VBLANK

//Counter für die Position zurücksetzen

//Abfrage /VBALL = low?

//Wartezeit für Taktflankenermittlung

```

```

52         if(GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_1) == 1)           //Positive Taktflanke /VBALL?
53         {
54             Vcounter = TIM_GetCounter(TIM4);                         //Aktuelle Position in Variable speichern
55             var = Vcounter - countold;                                //Prüfung ob nach oben oder unten fliegt: < 0
56             -> Ball fliegt nach oben
57
58             if(var < 0)                                               //Ausgleich für das Paddle "Ball fliegt nach
59             {                                                         oben"
60                 Vcounter = Vcounter - 3;
61             }
62
63             if(var > 0) Vcounter++;                                    //Ausgleich für das Paddle "Ball fliegt nach
64             countold = Vcounter;                                       unten"
65             Durchlauf speichern                                       //Aktueller Counterwert für nächsten
66         }
67     }
68     TIM_SetCompare1(TIM2, Vcounter - 30);                             //Anpassung und Ausgabe der Positionsvariable
69     als PWM-Signal
70
71     if((GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_2) == 0))             //Rücksetzen des PWM Counter
72     {
73         TIM_SetCounter(TIM2, 0);
74     }
75
76     if (TIM_GetFlagStatus(TIM4, TIM_FLAG_Update) == SET)             //Timerflag bei Überlauf zurücksetzen (TIM4)
77     {
78         TIM_ClearFlag(TIM4, TIM_IT_Update);
79         GPIO_ToggleBits(GPIOD, GPIO_Pin_14);
80     }
81
82     if (TIM_GetFlagStatus(TIM2, TIM_FLAG_Update) == SET)             //Timerflag bei Überlauf zurücksetzen (TIM2)
83     {
84         TIM_ClearFlag(TIM2, TIM_IT_Update);
85         GPIO_ToggleBits(GPIOD, GPIO_Pin_15);
86     }
87 }
88 }
89 }
90
91
92
93 void GPIO_Ini (void)
94 {
95     RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);          //Aktivierung Clocksignal AHB1
96     RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
97     RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

```

```

98
99     GPIO_InitTypeDef GPIO_InitStructure;
100
101     GPIO_InitStructure.GPIO_Pin =                               //Pins für die LEDs auf der STM-Platine
102     GPIO_Pin_12 |
103     GPIO_Pin_13 |
104     GPIO_Pin_14 |
105     GPIO_Pin_15;
106     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
107     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
108     GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
109     GPIO_Init (GPIOB, &GPIO_InitStructure);
110
111     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0                    //Definition der Input-Pins
112     | GPIO_Pin_1                                                //Pin: /VBLANK
113     | GPIO_Pin_2                                                //Pin: /VBALL
114     | GPIO_Pin_3;                                              //Abfrage Attract-Mode
115     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;              //Rücksetzsignal für das PWM-Signal
116     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
117     GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_DOWN;
118     GPIO_Init (GPIOB, &GPIO_InitStructure);
119 }
120
121 void TIM_Ini1(void)                                           //Timer 4 - Erfassung der vertikalen Position
122 {
123     TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStructure;
124
125     RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);      //Timer am Bus freigeben
126
127     TIM_TimeBaseStructInit (&TIM_TimeBaseInitStructure);
128     TIM_TimeBaseInitStructure.TIM_Prescaler = 4999;           //Vorteiler für den Takt einstellen
129     TIM_TimeBaseInitStructure.TIM_Period = 300;               //Zähle von 0 bis 300
130     TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up; //Aufwärtszähler
131     TIM_TimeBaseInit (TIM4, &TIM_TimeBaseInitStructure);
132     TIM_Cmd(TIM4, ENABLE);                                     //Zähler starten
133 }
134
135 void TIM_Ini2(void)                                           //Timer 2 - Erzeugung des PWM-Signal
136 {
137     TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStructure;
138     GPIO_InitTypeDef      GPIO_InitStructure;
139     TIM_OCInitTypeDef      TIM_OCInitStructure;
140
141     RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);      //Timer am Bus freigeben
142     RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA , ENABLE);    //Timer an Bus freigeben
143
144     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;              //GPIO_Mode: Alternate Function
145     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;                 //Ausgabe über GPIO_Pin_0
146     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
147     GPIO_Init (GPIOA, &GPIO_InitStructure);                   //Hinzufügen der Konfiguration zu GPIOA
148

```

```
149     GPIO_PinAFConfig(GPIOA, 0, GPIO_AF_TIM2); //GPIOA_Pin_0 mit Timer verbinden
150
151     TIM_TimeBaseStructInit(&TIM_TimeBaseInitStructure);
152     TIM_TimeBaseInitStructure.TIM_Prescaler = 5249; //Vorteiler für den Takt einstellen
153     TIM_TimeBaseInitStructure.TIM_Period = 262; //Definierter Bereich für die PWM
154     TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up;
155     TIM_TimeBaseInit(TIM2, &TIM_TimeBaseInitStructure);
156
157     TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1; //Timer Output-Compare aktivieren
158     TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable; //Ausgang aktivieren
159     TIM_OCInitStructure.TIM_Pulse = 130; //130 //
160     TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
161     TIM_OC1Init(TIM2, &TIM_OCInitStructure);
162     TIM_Cmd(TIM2, ENABLE); //Timer 2 starten
163 }
164
165
166
167
```