

# 浙江大学

## 本科实验报告

课程名称: 嵌入式系统

姓 名: Jessie Peng

学 院: 计算机科学与技术学院

系:

专 业:

学 号:

指导教师: 王总辉

2019 年 4 月 20 日

# 浙江大学实验报告

课程名称: 嵌入式系统 实验类型: 综合

实验项目名称: 实验 3: 前后台模式码表

学生姓名: Jessie Peng 专业:                  学号:                 

实验地点: 曹西 501 实验日期: 2019 年 4 月 20 日 指导老师: 王总辉

## 一、实验目的和要求

- 理解 MCU 上电启动过程
- 掌握使用 Cube 库来编写 STM32 裸机程序的方法
- 掌握使用 Cube 库来编写 GPIO 和 UART 程序的方法
- 掌握使用 Cube 库来编写中断响应程序的方法
- 理解前后台程序模式
- 掌握在 STM32 上编写裸机程序并下载运行的方法

## 二、实验内容和原理

### 【实验器材——硬件】

- STM32F103 核心板
- microUSB 线（供电）
- STLink 板

### 【实验器材——软件】

- MDK 软件及扩展
  - mdk\_513
  - Keil.STM32F1xx\_DFP.2.0.0 pack
  - stm32cubemx

### 【实验器材——其它资料】

- STM32 基础
  - Keil/GCC 生成 HEX 文件

○ st-link v2 下载程序

- STM32CubeMX 设置
- lab3 步骤

#### 【实验内容——编写输出"Hello, World"程序】

- 编写 Cube 程序，配置 UART0 为 9600, 8n1，上电后向串口输出"Hello, World!"
- 在 PC 上通过串口软件观察结果

#### 【实验内容——编写循环检测输入信号程序】

- 通过面包板在 PA11 和 PA12 各连接一个按钮开关到地（若无按钮，可手动触碰在一起模拟按钮）
- 编写 Cube 程序，配置 PA11 和 PA12 为内部上拉到输入模式，在 main() 函数循环检测 PA11 按钮按下，并在按钮按下时在串口输出“Pressed”。

#### 【实验内容——编写中断处理程序】

- 编写 Cube 程序，配置 PA12 下降沿触发中断，程序中设置两个全局变量，一个为计数器，一个为标识
- 当中断触发时，计数器加 1，并设置标识
- 在主循环中判断标识，如果标识置位则清除标识并通过串口输出计数值
- 编写 Cube 程序，开启定时器为 200ms 中断一次，中断触发计数器自增，主循环根据计数器值来做串口输出（取消上一步的串口输出）

#### 【实验内容——编写完整的码表程序】

- 编写完整的码表程序，PA12 的按钮表示车轮转了一圈，通过计数器可以得到里程，通过定时器中断得到的时间可以计算出速度
- PA11 的按钮切换模式，模式一在串口输出里程，模式二在串口输出速度

### 三、实验步骤和结果记录

#### 【Keil MDK-ARM5 准备】

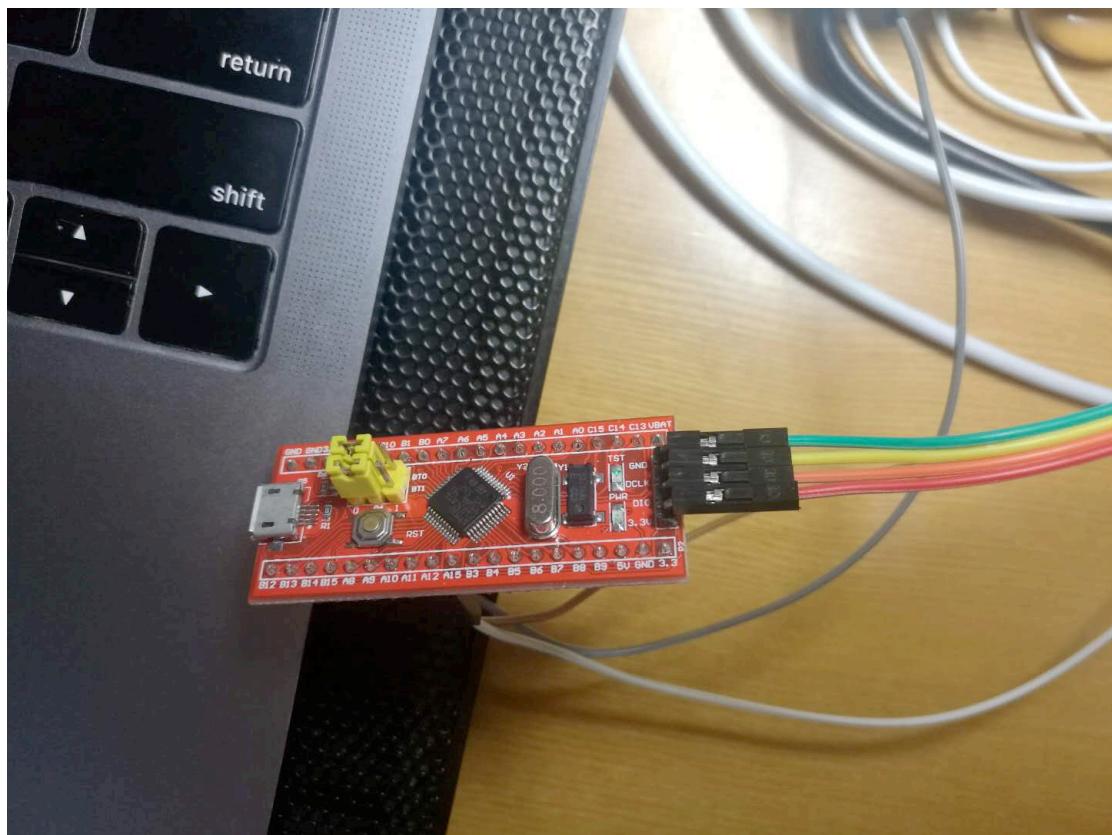
安装 Keil MDK-ARM 5.13，导入示例工程 blink 进行测试。

点击 Build 生成 HEX 文件。

按照如下方式连接 STLink 与 STM32:

STLink	STM32
3.3V	3.3V
GND	GND
SWDIO	DIO
SWCLK	DCLK

连线实物图如下:





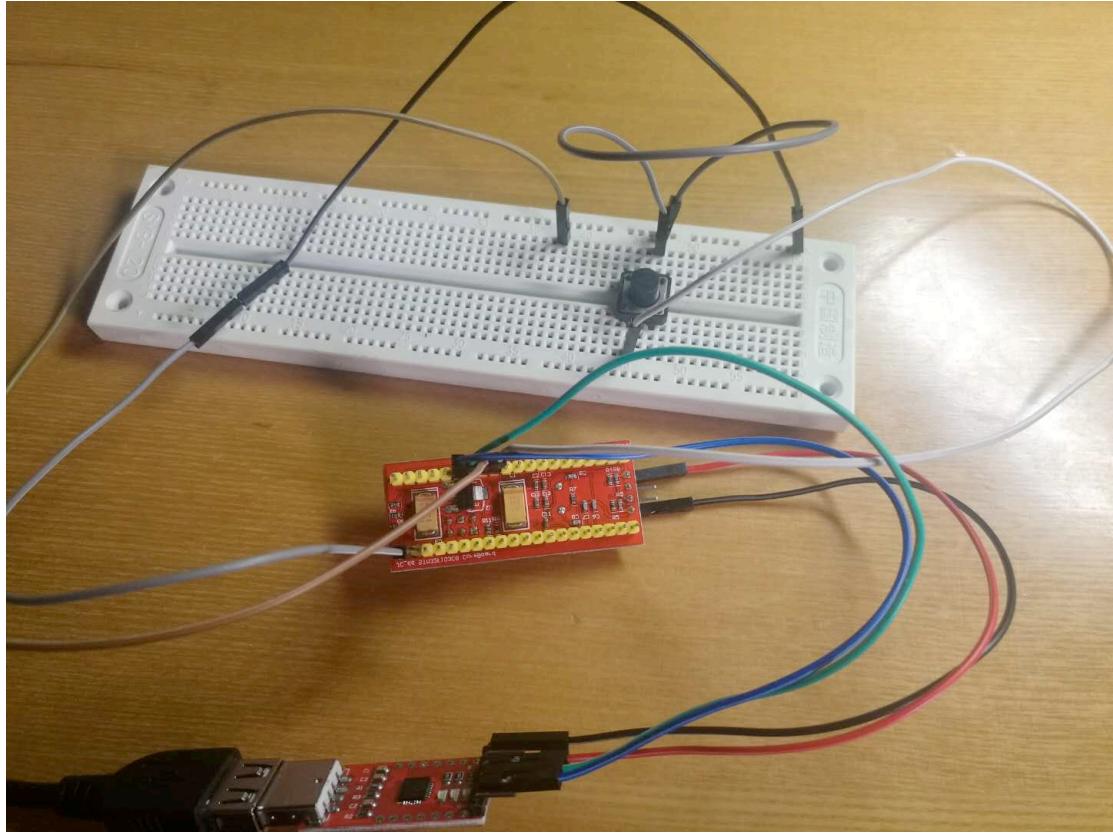
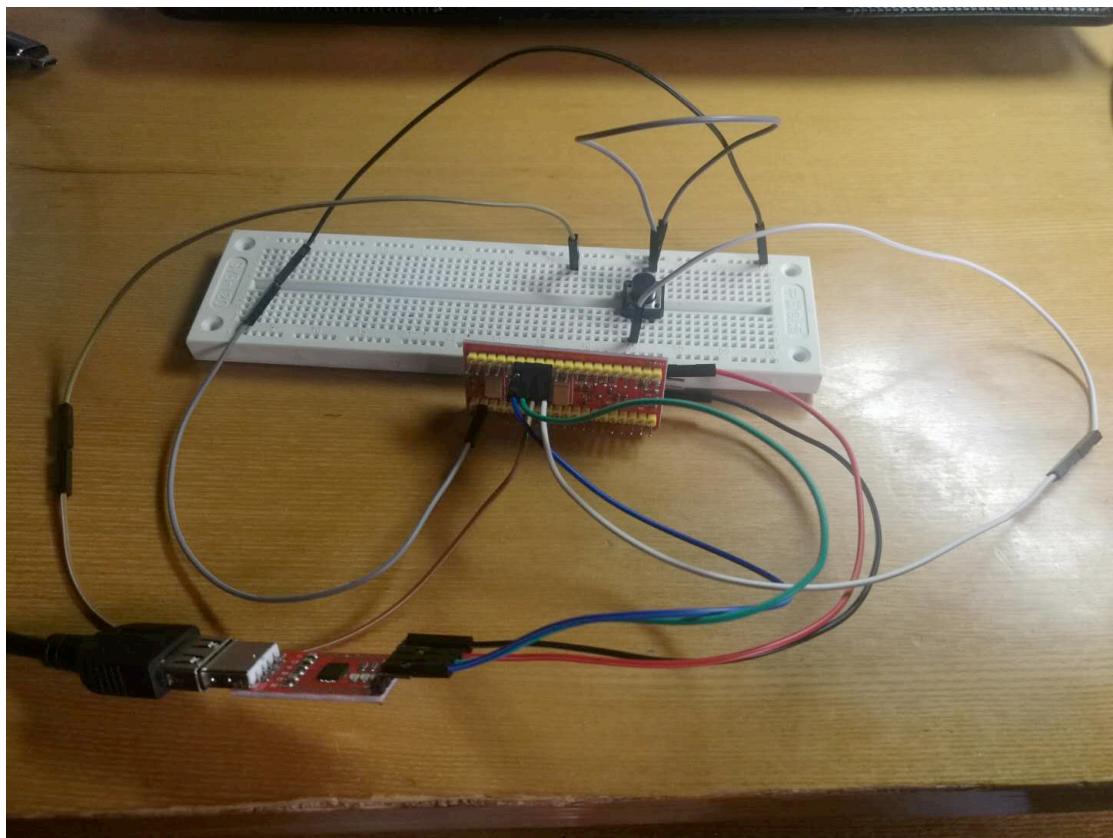
点击 Download 下载程序到开发板，可以观察到 LED 闪烁。

#### 【连接 USBtoTTL 与 STM32 开发板】

USB to TTL	STM32
3.3V	3.3V
GND	GND
TXD	A10
RXD	A9

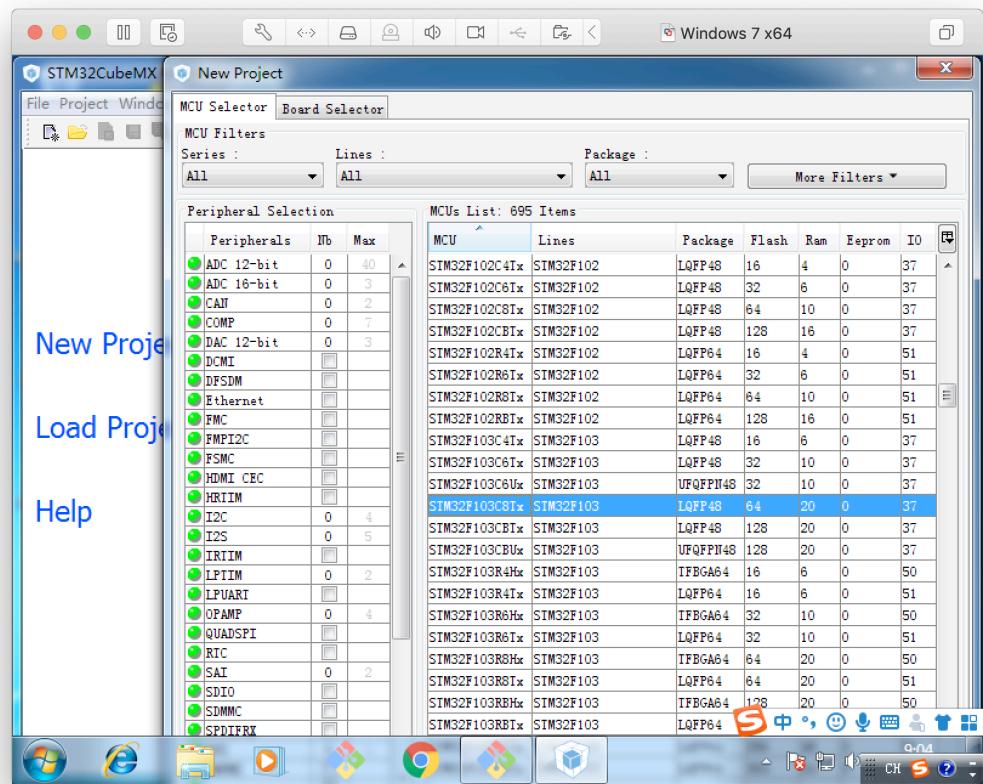
注：如果同时连了 USBtoTTL 和 STLink，则供电（3.3V/GND）只需接一个。

再将 A11 和 A12 分别通过按钮连接到 GND（由于只有一个按钮，则 A11 先不接，之后人工连接到 GND 来模拟按钮操作）。连线实物图如下：

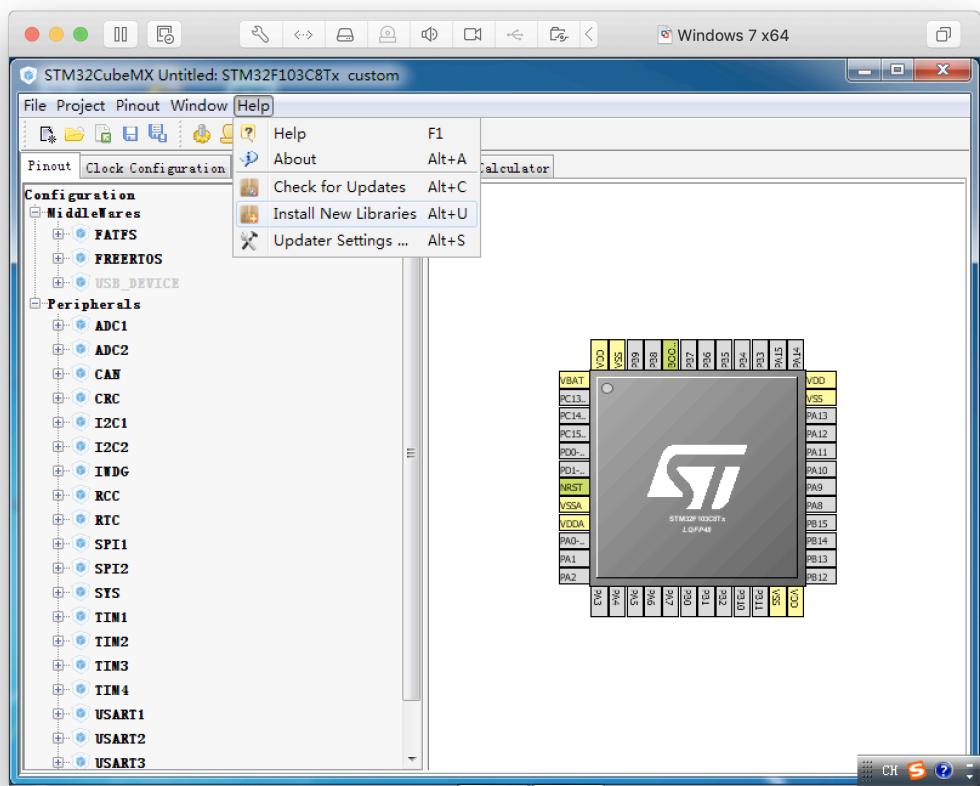


## 【STM32CubeMX 准备】

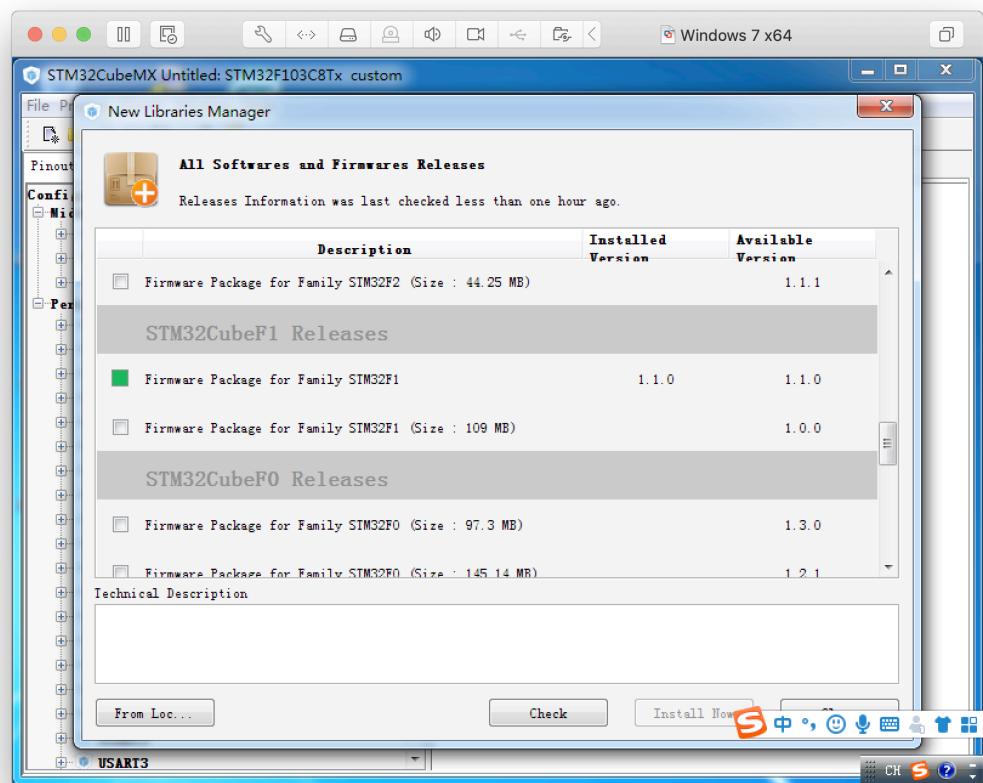
安装 STM32CubeMX，新建工程，选择核心板型号 STM32F103C8Tx：



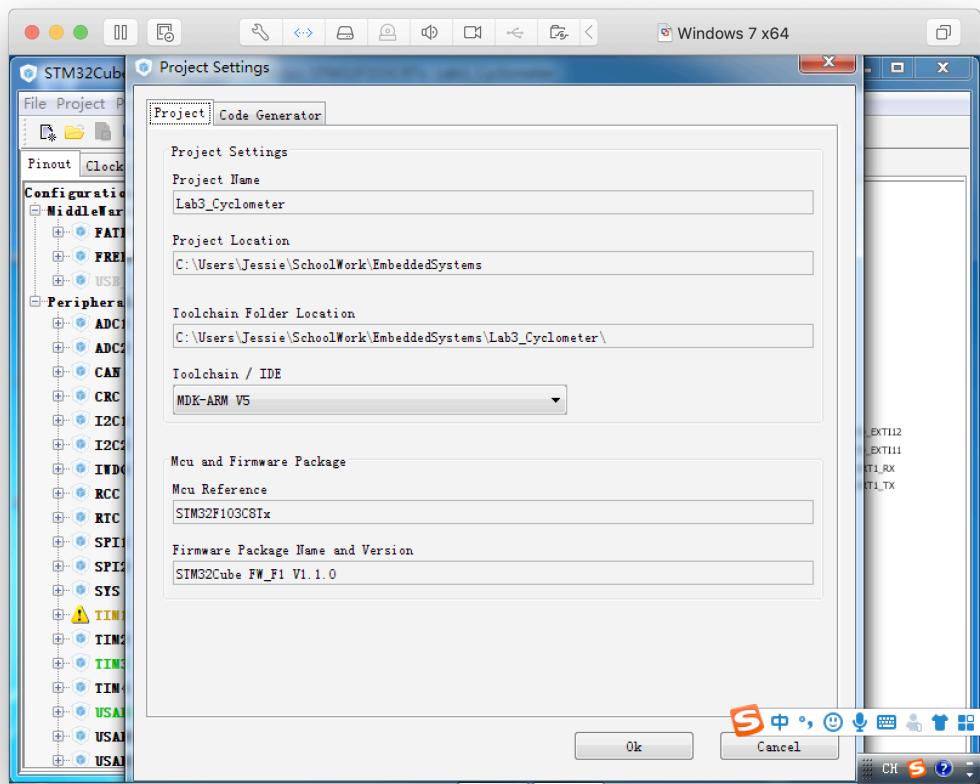
点击 help->install new libraries:



将 Cube 库文件解压到 repository folder 的路径下，检查 Cube 库已经导入了：

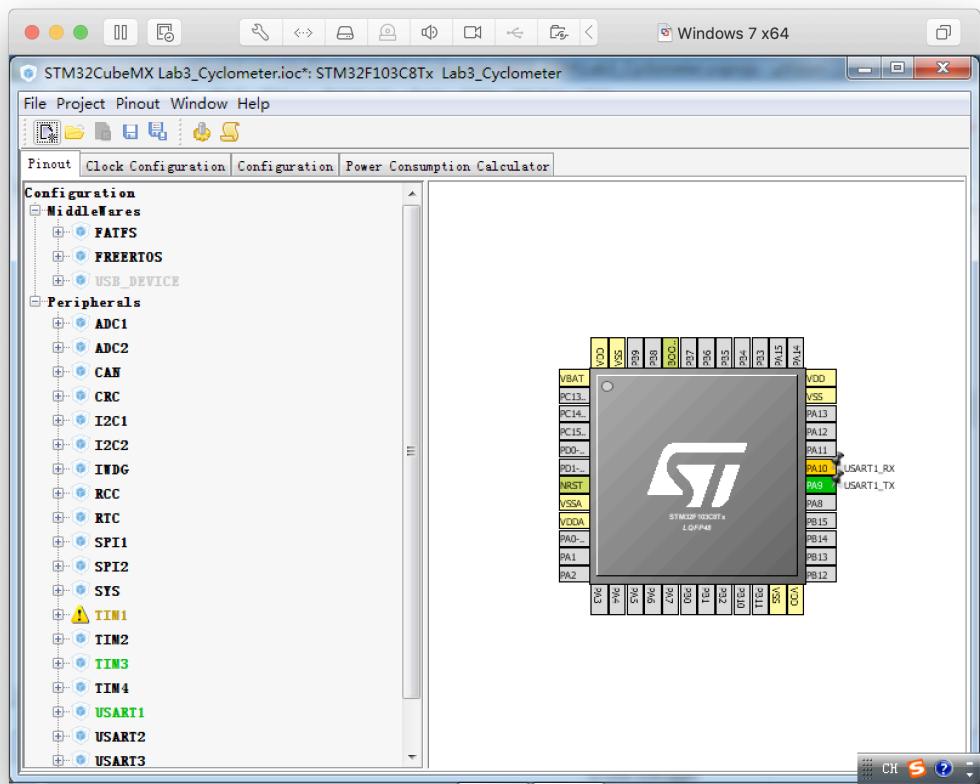


设置项目属性（名称、位置、工具链）：

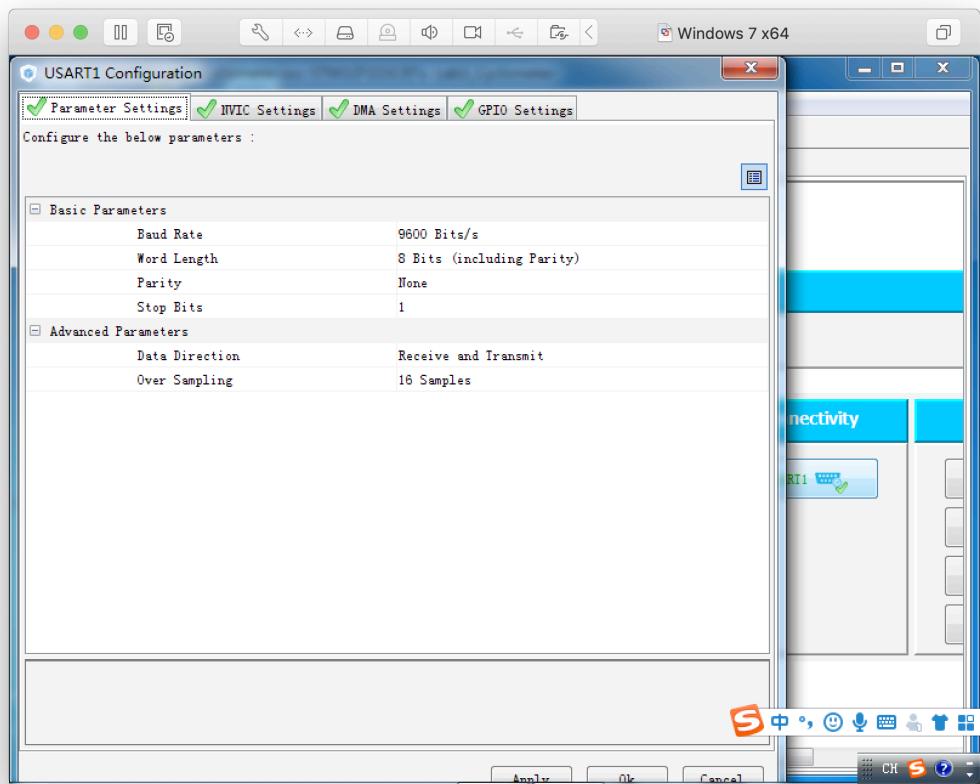


## 【编写输出 Hello World 程序】

设置 PA9、PA10 引脚为串口通信：



配置 UART (波特率 9600, 字长 8 比特, 停止位 1 比特, 无校验):



点击 generate source code based on user settings 生成代码。

用 Keil 打开刚才 CubeMX 生成的工程文件编写代码。

注：写代码时写在 USER CODE BEGIN 和 USER CODE END 之间，以后在 CubeMX 里面修改了配置，重新生成代码后，这些用户代码不会被改动。

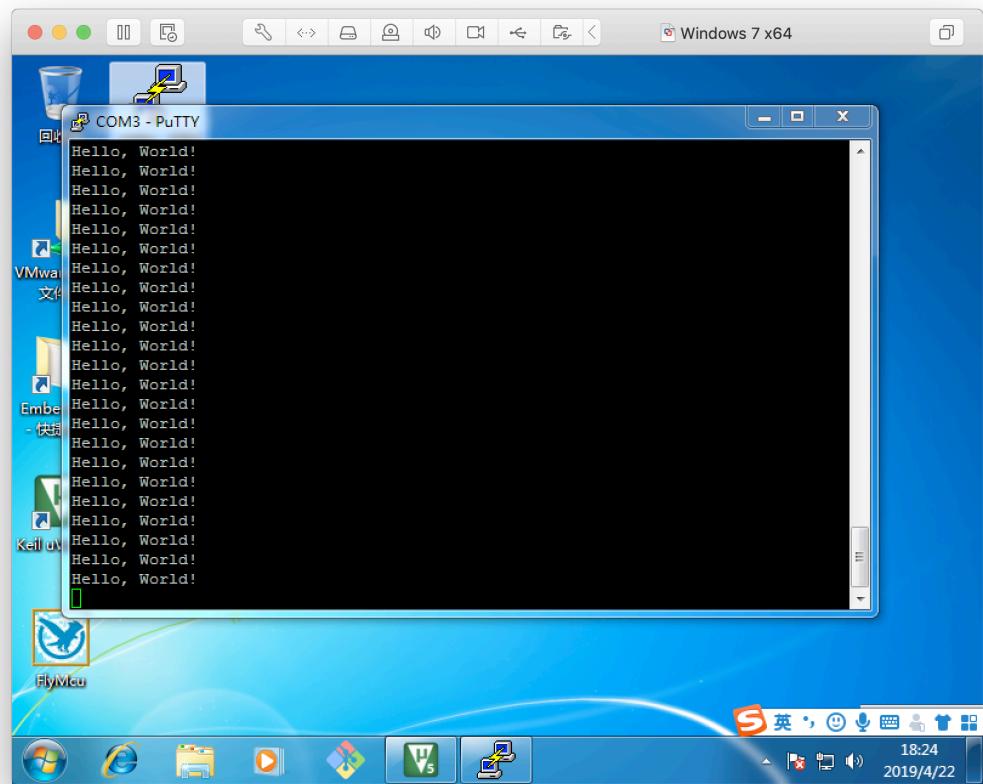
在主函数的 while 循环中向串口每 100ms 发送一次字符串"Hello, World!\r\n":

The screenshot shows the MDK-ARM IDE interface. The project 'Lab3\_Cyclometer' is open, and the main.c file is the active editor. The code contains a main loop with a printf statement:

```
90 MX_TIM3_Init();
91 MX_USART1_UART_Init();
92
93 /* USER CODE BEGIN 2 */
94
95 /* Infinite loop */
96 /* USER CODE BEGIN WHILE */
97 while (1)
98 {
99     /* USER CODE END WHILE */
100
101     /* USER CODE BEGIN 3 */
102     printf("Hello, World!\r\n");
103     HAL_Delay(100);
104 }
105 /* USER CODE END 3 */
106
107
108
109
110 /**
111  * @brief System Clock Configuration
112 */
113 void SystemClock_Config(void)
114 {
115 }
```

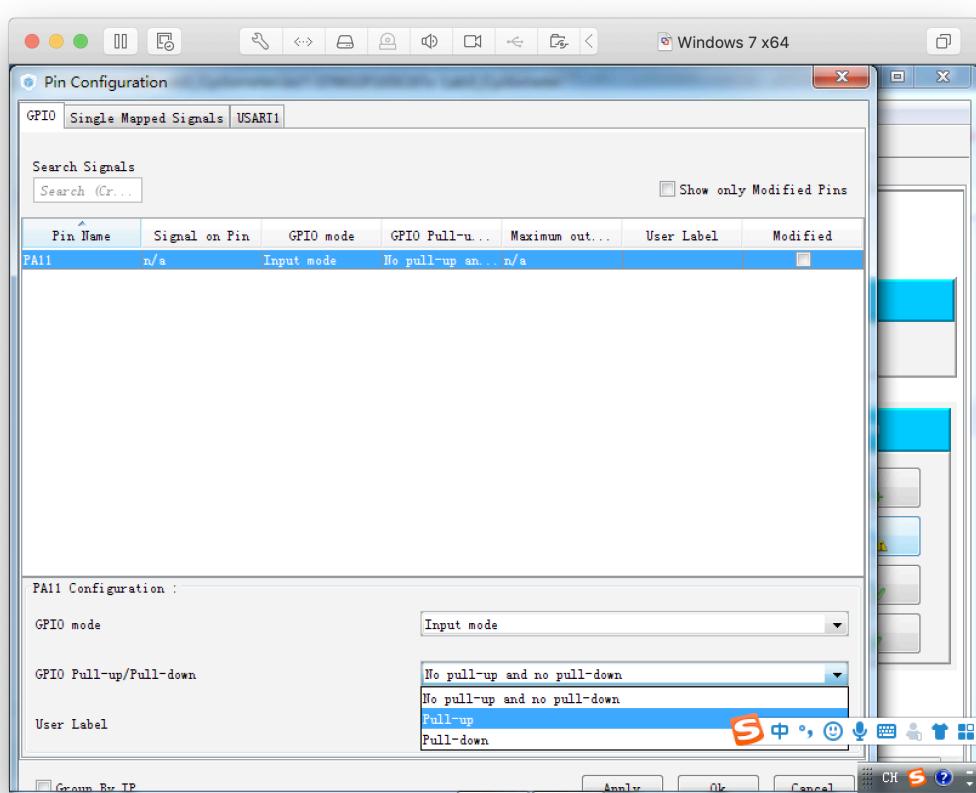
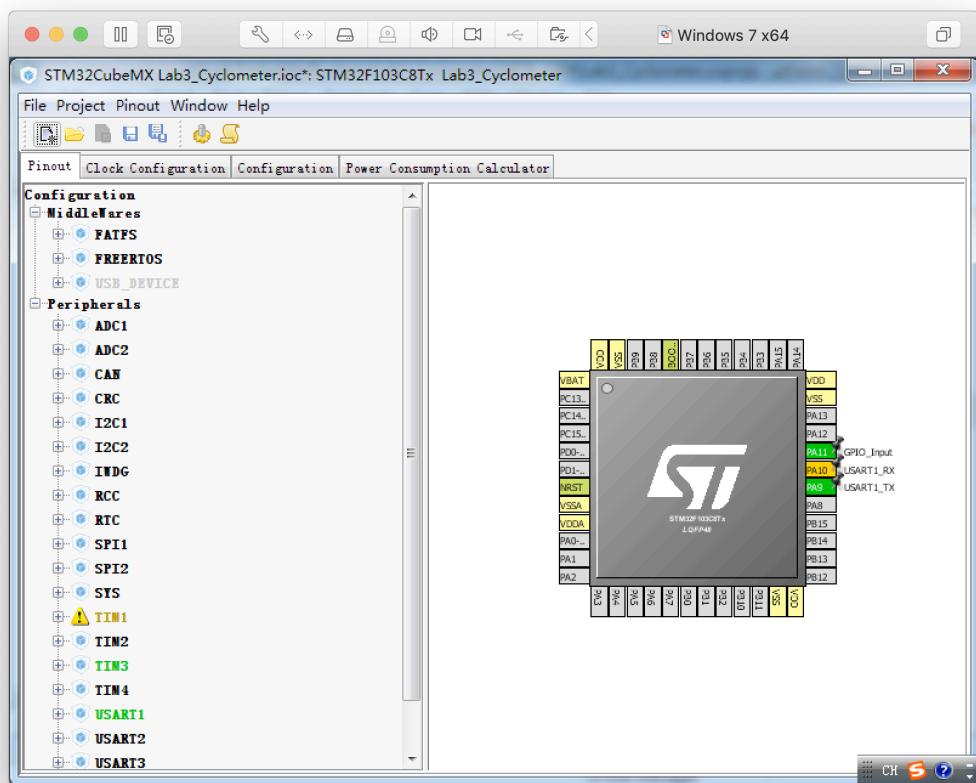
注：为了方便串口输出，我们重定向了标准库的 printf 函数。

生成 HEX 文件，用 STLink 烧录至 STM32 开发板。烧录成功后，用 PuTTY 在 Windows 主机上查看串口输出：

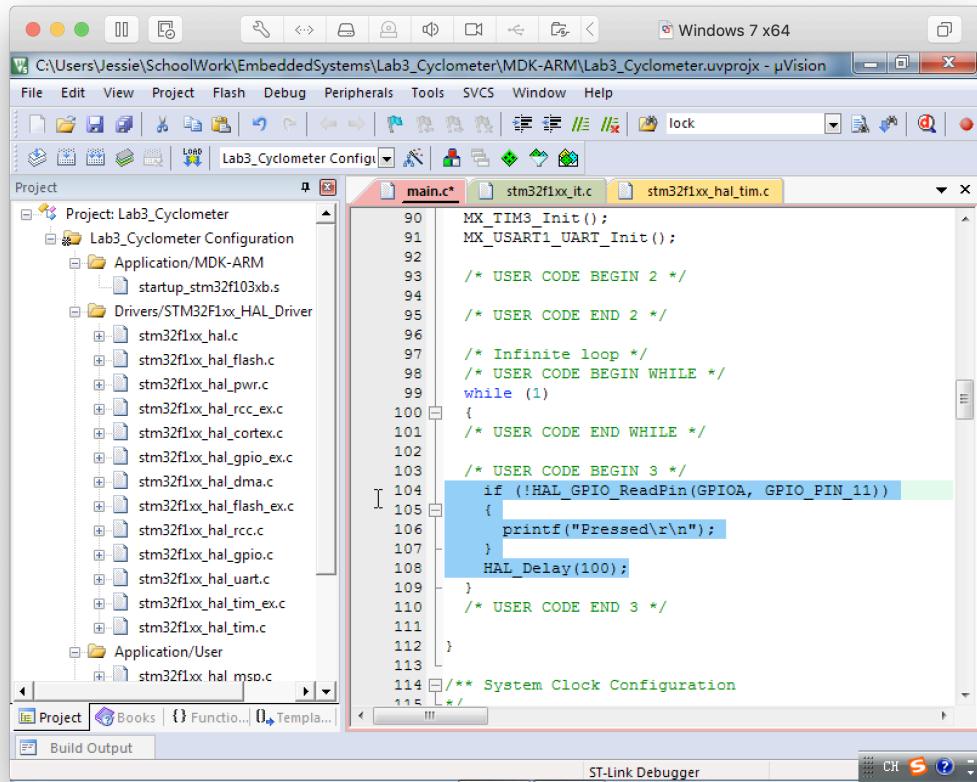


### 【编写循环检测输入信号程序】

修改配置，将 PA11 初始化为上拉输入模式：



main 函数中修改为按下 PA11 的按钮输出"Pressed\r\n":



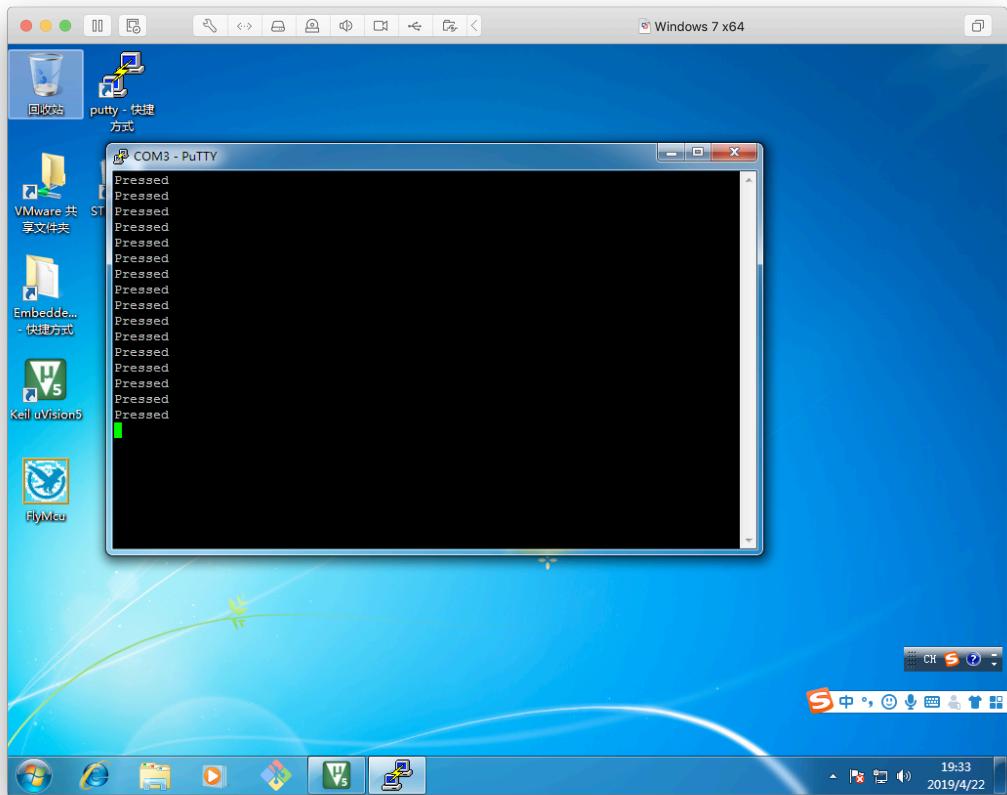
The screenshot shows the MDK-ARM uVision IDE interface. The project is named 'Lab3\_Cyclometer'. The main.c file is open in the editor. A specific line of code has been highlighted with a blue selection:

```
90    MX_TIM3_Init();
91    MX_USART1_UART_Init();
92
93    /* USER CODE BEGIN 2 */
94
95    /* USER CODE END 2 */
96
97    /* Infinite loop */
98    /* USER CODE BEGIN WHILE */
99    while (1)
100    {
101        /* USER CODE END WHILE */
102
103        /* USER CODE BEGIN 3 */
104        if (!HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_11))
105        {
106            printf("Pressed\r\n");
107        }
108        HAL_Delay(100);
109    }
110    /* USER CODE END 3 */
111
112}
113
114 /** System Clock Configuration
115 */

The highlighted line is:
```

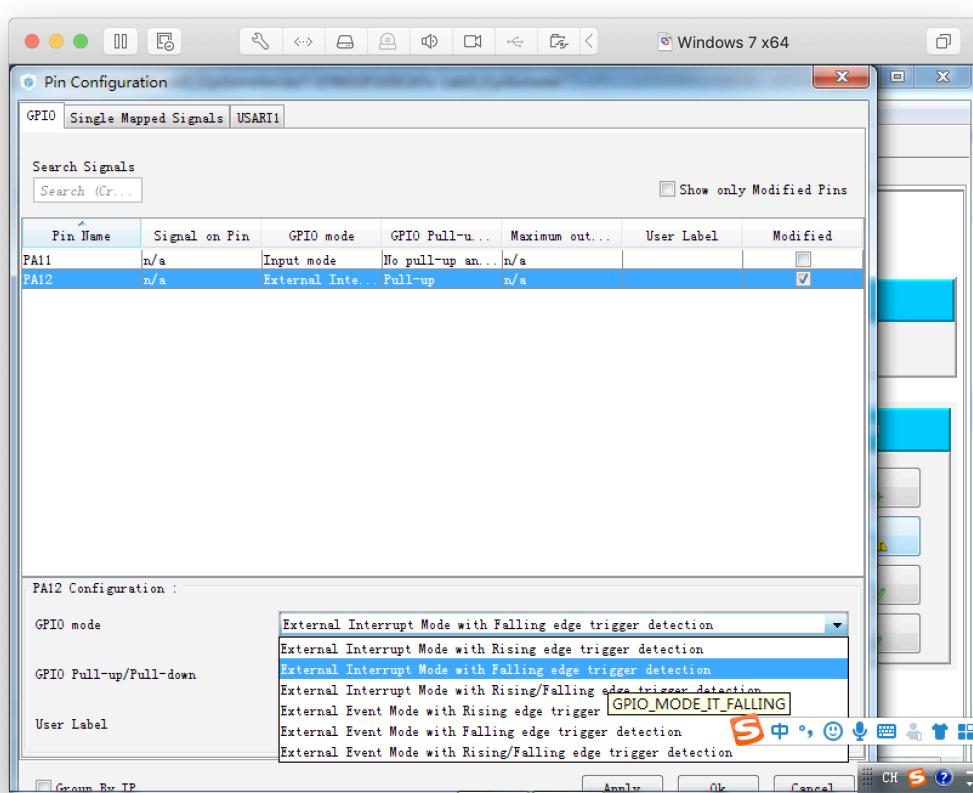
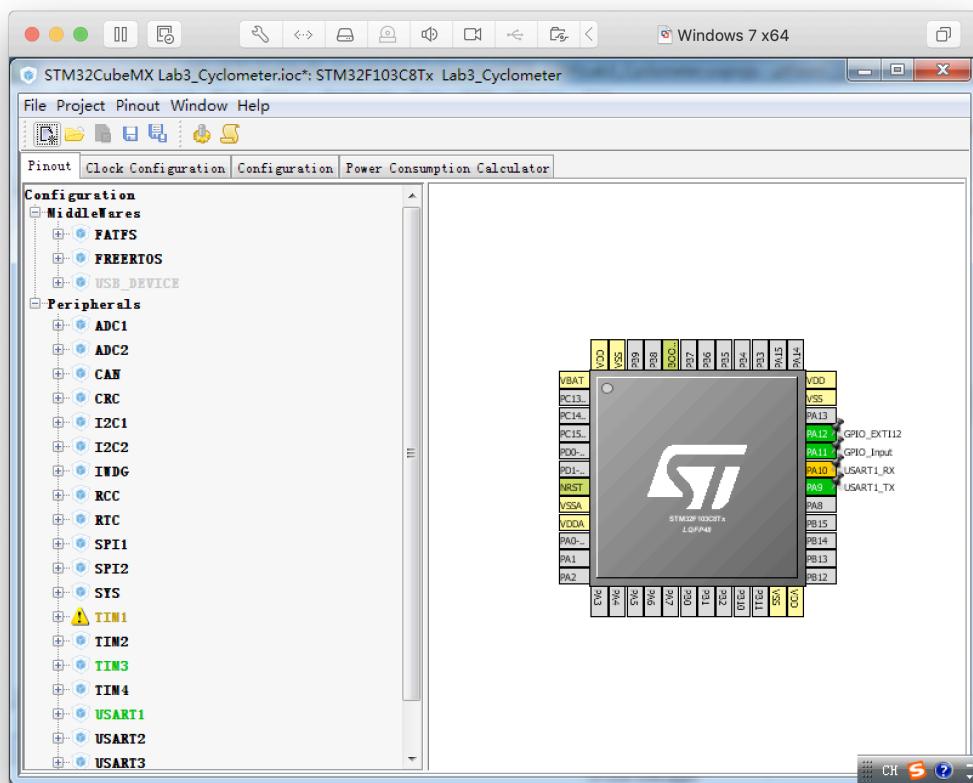
if (!HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_11))

编译并下载，查看串口输出：

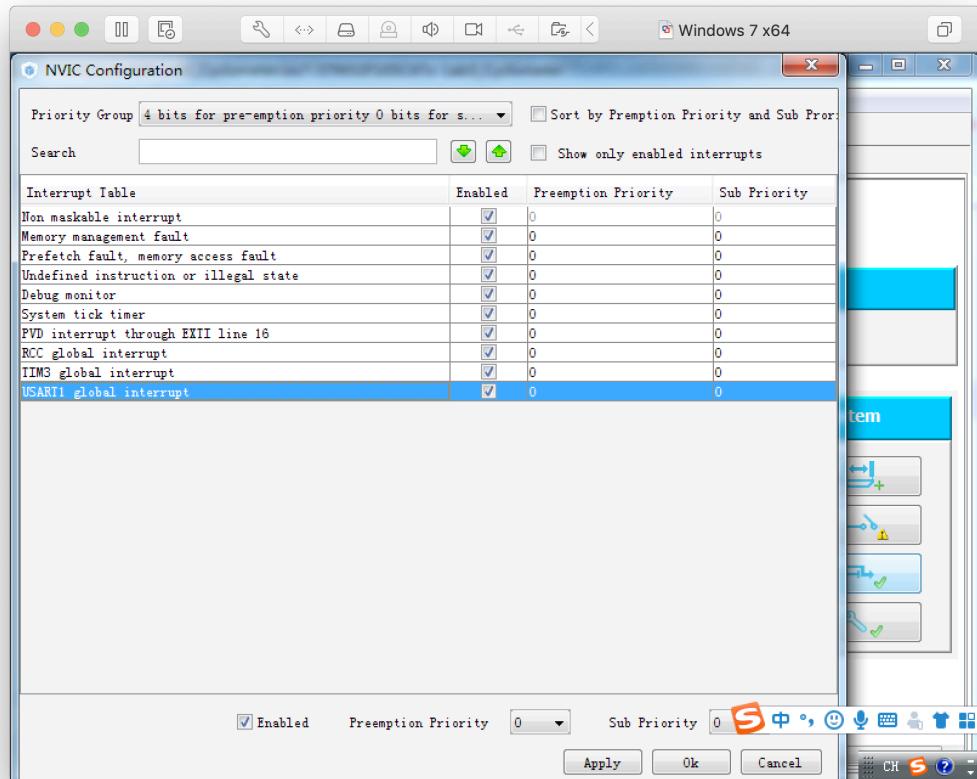


## 【编写中断处理程序】

编写在 PA12 下降沿触发中断的程序，首先配置 PA12 为上拉模式并且在下降沿触发中断：



使能中断向量表并设置中断优先级：



当该中断触发时会从 `stm32f1xx_it.c` 中调用对应的中断处理程序，即 `EXTI15_10_IRQHandler` 函数：

```
198 HAL_UART_IRQHandler(&huart1);
199 /* USER CODE BEGIN USART1_IRQHandler */
200
201 /* USER CODE END USART1_IRQHandler */
202 }
203
204 /**
205 * @brief This function handles EXTI line[15:10] interrupt.
206 */
207 void EXTI15_10_IRQHandler(void)
208 {
209     /* USER CODE BEGIN EXTI15_10_IRQHandler */
210
211     /* USER CODE END EXTI15_10_IRQHandler */
212     HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_12); // Cursor here
213     /* USER CODE BEGIN EXTI15_10_IRQHandler */
214
215     /* USER CODE END EXTI15_10_IRQHandler */
216 }
217
218 /* USER CODE BEGIN 1 */
219
220 /* USER CODE END 1 */
221 //***** (C) COPYRIGHT STMicroelectronics *****
```

函数里面又调用了 HAL 库中的 HAL\_GPIO\_EXTI\_IRQHandler 函数，它封装了判断中断标志位、调用中断回调函数、清除标志位等过程：

```
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project Lab3_Cyclometer Config stm32f1xx_hal_gpio.c main.c stm32f1xx_it.c
stm32f1xx_hal_flash.c
stm32f1xx_hal_pwr.c
stm32f1xx_hal_rcc_ex.c
stm32f1xx_hal_cortex.c
stm32f1xx_hal_gpio_ex.c
stm32f1xx_hal_dma.c
stm32f1xx_hal_flash_ex.c
stm32f1xx_hal_rcc.c
stm32f1xx_hal_gpio.c
stm32f1xx_hal_uart.c
stm32f1xx_hal_tim_ex.c
stm32f1xx_hal_tim.c
Application/User
stm32f1xx_hal_msp.c
stm32f1xx_it.c
main.c
Drivers/CMSIS
system_stm32f1xx.c

552 * @brief This function handles EXTI interrupt request
553 * @param GPIO_Pin: Specifies the pins connected EXTI :
554 * @retval None
555 */
556 void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
557 {
558     /* EXTI line interrupt detected */
559     if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
560     {
561         __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
562         HAL_GPIO_EXTI_Callback(GPIO_Pin);
563     }
564 }
565 /**
566 * @brief EXTI line detection callback
567 * @param GPIO_Pin: Specifies the pins connected EXTI :
568 * @retval None
569 */
570 __weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
571 {
572     /* NOTE : This function Should not be modified, when
573      * the HAL_GPIO_EXTI_Callback could be implemented
574      */
575 }
576
577 */

Build Output ST-Link Debugger
```

默认的中断回调函数 HAL\_GPIO\_EXTI\_Callback 是由 `_weak` 关键字修饰的，因此接下来在 main.c 里面重写这个函数：

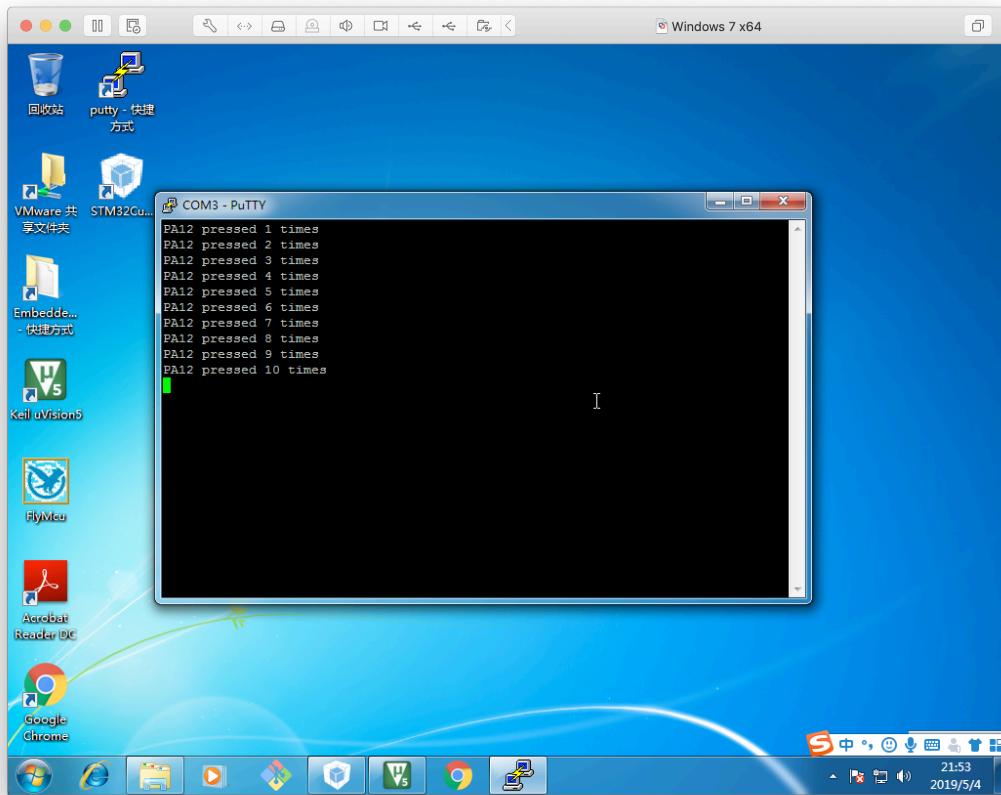
```
202  /* EXTI interrupt init*/
203  HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
204  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
205
206 }
207
208 /* USER CODE BEGIN 4 */
209 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
210 {
211     if (GPIO_Pin == GPIO_PIN_12 && PA12_Flag == 0)
212     {
213         PA12_Flag = 1;
214         PA12_Count++;
215     }
216     else
217     {
218         UNUSED(GPIO_Pin);
219     }
220 }
221
222 /**
223 * @brief Retargets the C library printf function to
224 * @param None
225 * @retval None
226 */
227 #include "printf.h"
```

main 函数也相应修改：

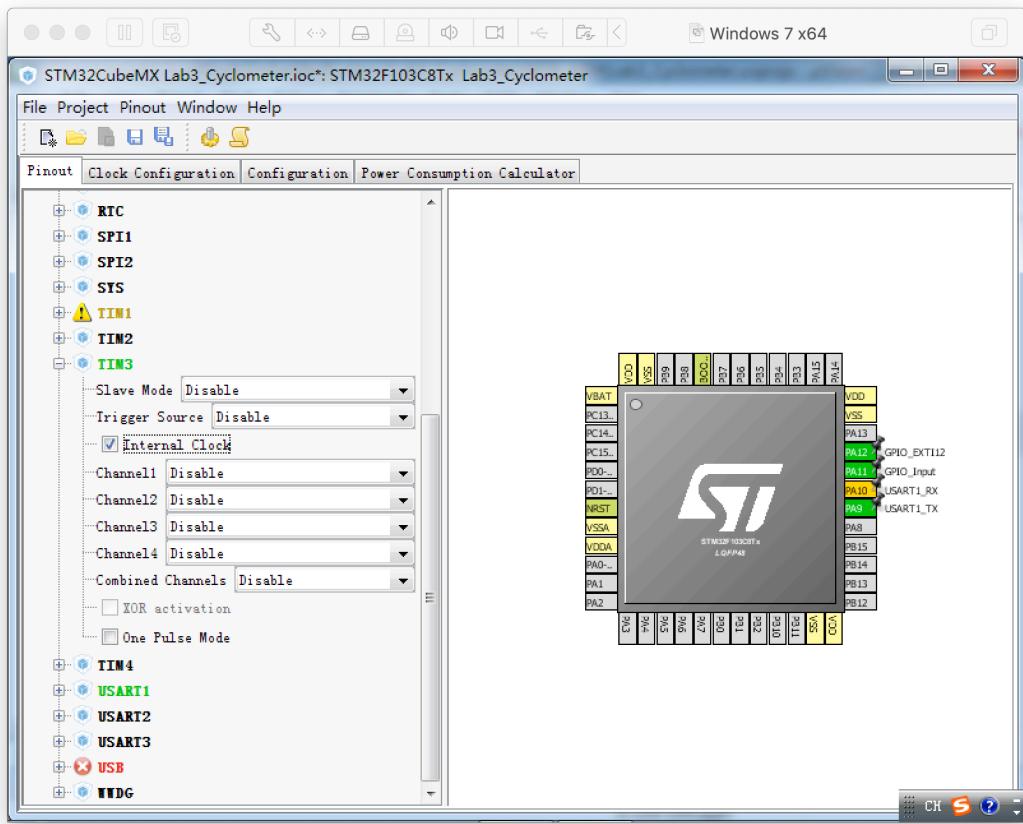
The screenshot shows the MDK-ARM IDE interface on Windows 7 x64. The project is named 'Lab3\_Cyclometer' and the file being edited is 'main.c'. The code implements a simple button press counter:

```
94  /* USER CODE BEGIN 2 */
95
96  /* USER CODE END 2 */
97
98  /* Infinite loop */
99  /* USER CODE BEGIN WHILE */
100 while (1)
101 {
102     /* USER CODE END WHILE */
103
104     /* USER CODE BEGIN 3 */
105     if (PA12_Flag == 1)
106     {
107         PA12_Flag = 0;
108         printf("PA12 pressed %d times\r\n", PA12_Count);
109     }
110     /* USER CODE END 3 */
111
112 }
113
114
115 /** System Clock Configuration
116 */
117 void SystemClock_Config(void)
118 {
119 }
```

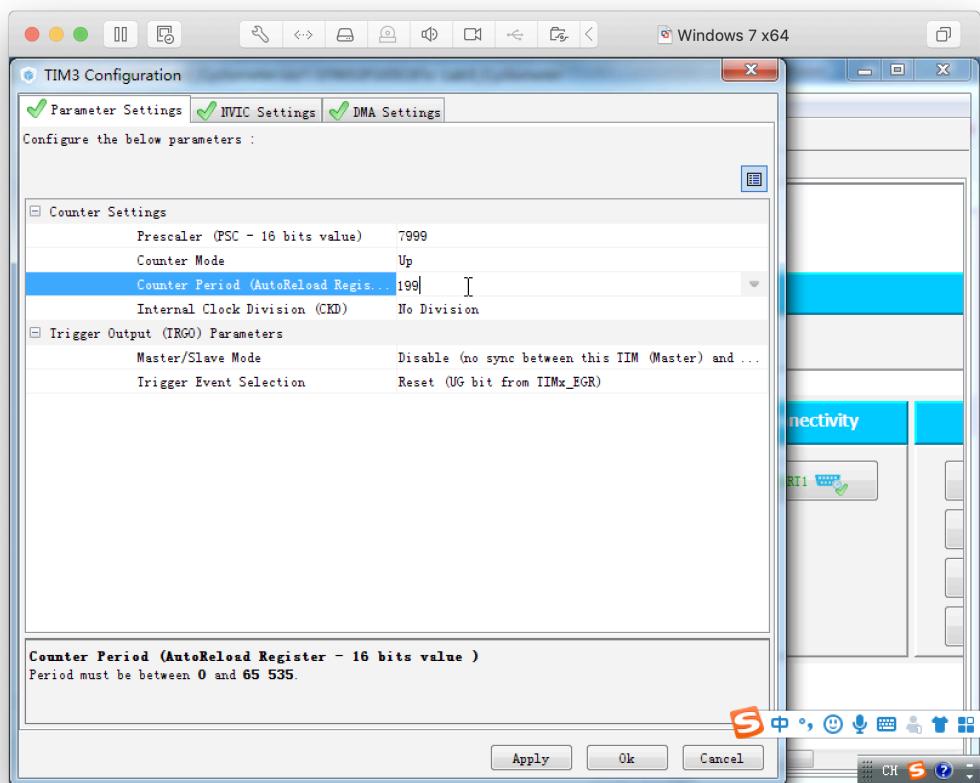
编译下载，运行结果：



下面改为定时器中断，使用 TIM3 计数，首先将 TIM3 设置为内部时钟：



设置预分频数和定时器周期，使一个周期为 200ms:



类似地，在 main.c 里面编写对应的回调函数，当计数周期结束时引发中断，进入该回调函数：

The screenshot shows the MDK-ARM IDE interface with the project 'Lab3\_Cyclometer' open. The 'main.c' file is the active editor window. The code has been modified to include a printf statement at line 233:

```
214     PA12_Flag = 1;
215     PA12_Count++;
216 }
217 else
218 {
219     UNUSED(GPIO_Pin);
220 }
221 }
222
223 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *h)
224 {
225     TIM3_Flag = 1;
226 }
227
228 /**
229 * @brief Retargets the C library printf function to 1
230 * @param None
231 * @retval None
232 */
233 PUTCHAR_PROTOTYPE
234 {
235     HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 0xffff);
236     return ch;
237 }
238
239 /* USER CODE END 4 */
240
```

主循环修改为对应的输出：

The screenshot shows the MDK-ARM IDE interface with the project 'Lab3\_Cyclometer' open. The main window displays the 'main.c' file, which contains C code for a STM32F1xx microcontroller. The code includes comments indicating sections for user code and system clock configuration. A specific section of the code is highlighted in blue, showing an if-statement that increments a counter and prints its value.

```
94  /* USER CODE BEGIN 2 */
95
96  /* USER CODE END 2 */
97
98
99  /* Infinite loop */
100 /* USER CODE BEGIN WHILE */
101 while (1)
102 {
103     /* USER CODE END WHILE */
104
105     /* USER CODE BEGIN 3 */
106     if (TIM3_Flag == 1)
107     {
108         TIM3_Count++;
109         TIM3_Flag = 0;
110         printf("counter: %d times\r\n", TIM3_Count);
111     }
112     /* USER CODE END 3 */
113
114
115 }
116
117 /** System Clock Configuration
118 */
119 void SystemClock_Config(void)
```

注意主函数里面还需要这一个初始化语句：

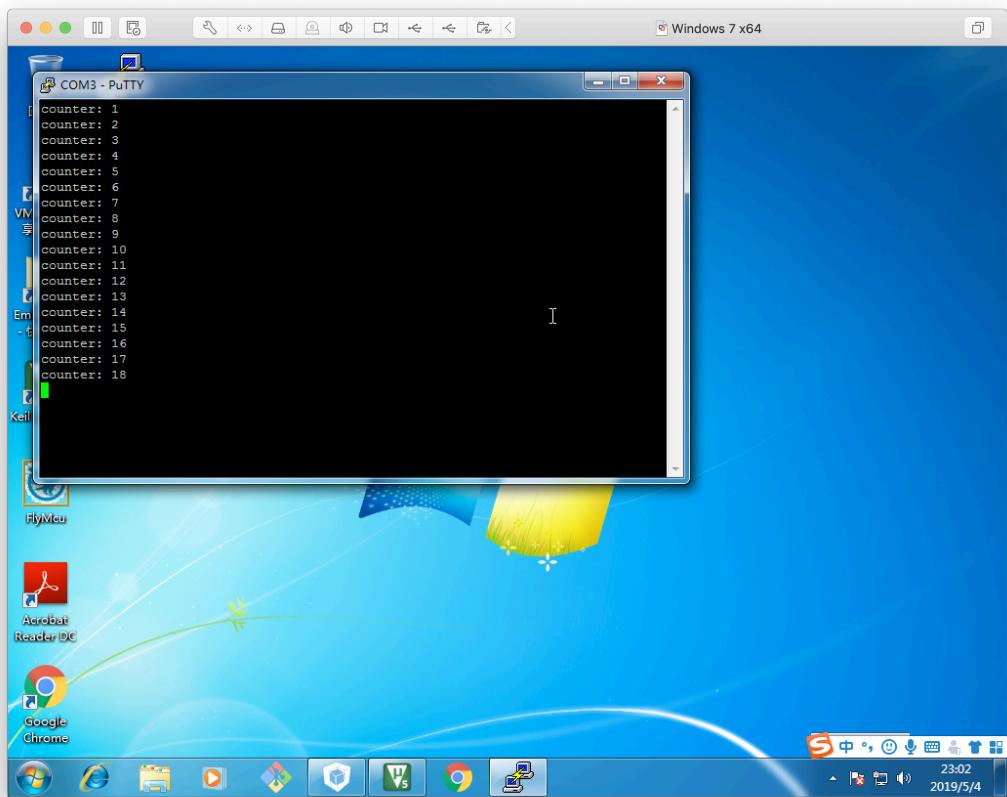
The screenshot shows the MDK-ARM IDE interface with the following details:

- Title Bar:** C:\Users\Jessie\SchoolWork\EmbeddedSystems\Lab3\_Cyclometer\MDK-ARM\Lab3\_Cyclometer.uvprojx - µVision
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbars:** Standard, Project, Build, Run, Debug, Peripherals, Tools, SVCS, Window, Help.
- Project Explorer:** Shows the project structure under "Lab3\_Cyclometer Config".
- Code Editor:** The main.c file is open, displaying the following C code:

```
75 int main(void)
76 {
77     /* USER CODE BEGIN 1 */
78
79     /* USER CODE END 1 */
80
81     /* MCU Configuration-----*/
82
83     /* Reset of all peripherals, Initializes the Flash interface of the
HAL_Init();
84
85     /* Configure the system clock */
SystemClock_Config();
86
87     /* Initialize all configured peripherals */
MX_GPIO_Init();
88     MX_TIM3_Init();
89     MX_USART1_UART_Init();
90
91     /* USER CODE BEGIN 2 */
92     HAL_TIM_Base_Start_IT(&htim3);
93     /* USER CODE END 2 */
94
95     /* Infinite loop */
96     /* USER CODE BEGIN WHILE */
97
98     /* USER CODE END WHILE */
99
100    /* USER CODE BEGIN 3 */
101}
```

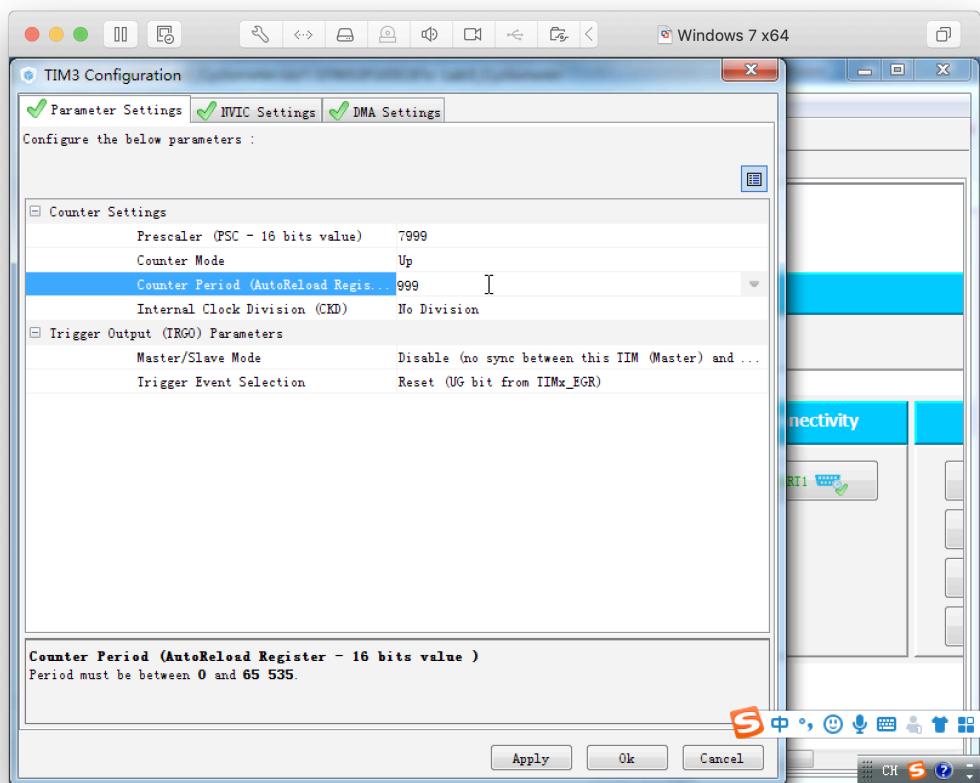
- Status Bar:** ST-Link Debugger

运行结果:



### 【编写完整的码表程序】

修改 TIM3 的设置使周期为 1 秒，以便于计算速度：



将 PA11 设置为和 PA12 一样的上拉并且下降沿触发中断。

PA11 和 PA12 触发的外部中断共用一个中断信号量，因此要在中断回调函数中判断是哪个按钮按下，修改回调函数如下：

The screenshot shows the MDK-ARM IDE interface with the project 'Lab3\_Cyclometer' open. The main window displays the 'main.c' file, which contains C code for handling GPIO EXTI callbacks. The code snippet is as follows:

```
228 }
229 /* USER CODE BEGIN 4 */
230 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
231 {
232     if (GPIO_Pin == GPIO_PIN_12)
233     {
234         PA12_Flag = 1;
235     }
236     else if (GPIO_Pin == GPIO_PIN_11)
237     {
238         PA11_Flag = 1;
239     }
240     else
241     {
242         UNUSED(GPIO_Pin);
243     }
244 }
245
246 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
247 {
248     TIM3_Count++;
249 }
250
251 /**
252 * Brief Description: the C library printf function to print
253 * !!!
```

主循环中增加一个延时 10ms 的去抖动处理，代码如下：

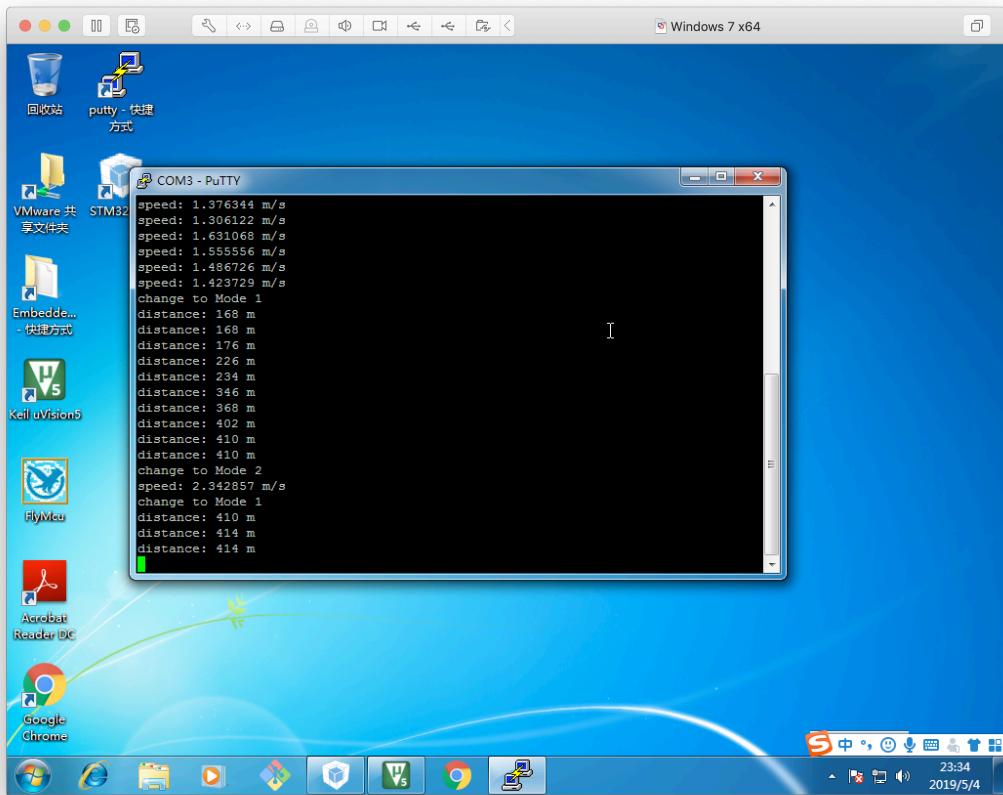
The screenshot shows the MDK-ARM IDE interface with the following details:

- Title Bar:** C:\Users\Jessie\SchoolWork\EmbeddedSystems\Lab3\_Cyclometer\MDK-ARM\Lab3\_Cyclometer.uvproj - μVision
- File Menu:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for Open, Save, Build, Run, and Debug.
- Project Explorer:** Shows the project structure:
  - Project: Lab3\_Cyclometer
  - Lab3\_Cyclometer Configuration
  - Application/MDK-ARM
    - startup\_stm32f103xb.s
  - Drivers/STM32F1xx\_HAL\_Driver
    - stm32f1xx\_hal.c
    - stm32f1xx\_hal\_flash.c
    - stm32f1xx\_hal\_pwr.c
    - stm32f1xx\_hal\_rcc\_ex.c
    - stm32f1xx\_hal\_cortex.c
    - stm32f1xx\_hal\_gpio\_ex.c
    - stm32f1xx\_hal\_dma.c
    - stm32f1xx\_hal\_flash\_ex.c
    - stm32f1xx\_hal\_rcc.c
    - stm32f1xx\_hal\_gpio.c
    - stm32f1xx\_hal\_uart.c
    - stm32f1xx\_hal\_tim\_ex.c
    - stm32f1xx\_hal\_tim.c
  - Application/User
    - stm32f1xx\_hal\_msp.c
    - stm32f1xx\_it.c
    - main.c
  - Drivers/CMSIS
    - system\_stm32f1xx.c
- Code Editor:** The main.c file is open, showing the following code:

```
99     while (1)
100    {
101        /* USER CODE END WHILE */
102
103        /* USER CODE BEGIN 3 */
104        //printf("Hello, World!\r\n");
105        if (PA11_Flag == 1)
106        {
107            HAL_Delay(10); // anti-jitter
108            if (!HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_11))
109            {
110                mode ^= 1;
111                printf("change to Mode %d\r\n", mode+1);
112            }
113            PA11_Flag = 0;
114        }
115        if (PA12_Flag == 1)
116        {
117            HAL_Delay(10); // anti-jitter
118            if (!HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_12))
119            {
120                rounds++;
121                if (mode == 0) // distance
122                {
123                    printf("distance: %d m\r\n", rounds*2);
124                }
125                else
126                {
127                    printf("speed: %f m/s\r\n", rounds*2/(double)TIM3_Count);
128                }
129            }
130            PA12_Flag = 0;
131        }
132    }
133    /* USER CODE END 3 */
134 }
```

- Toolbars:** ST-Link Debugger, CAP NUM SCR L/OVR R/W.

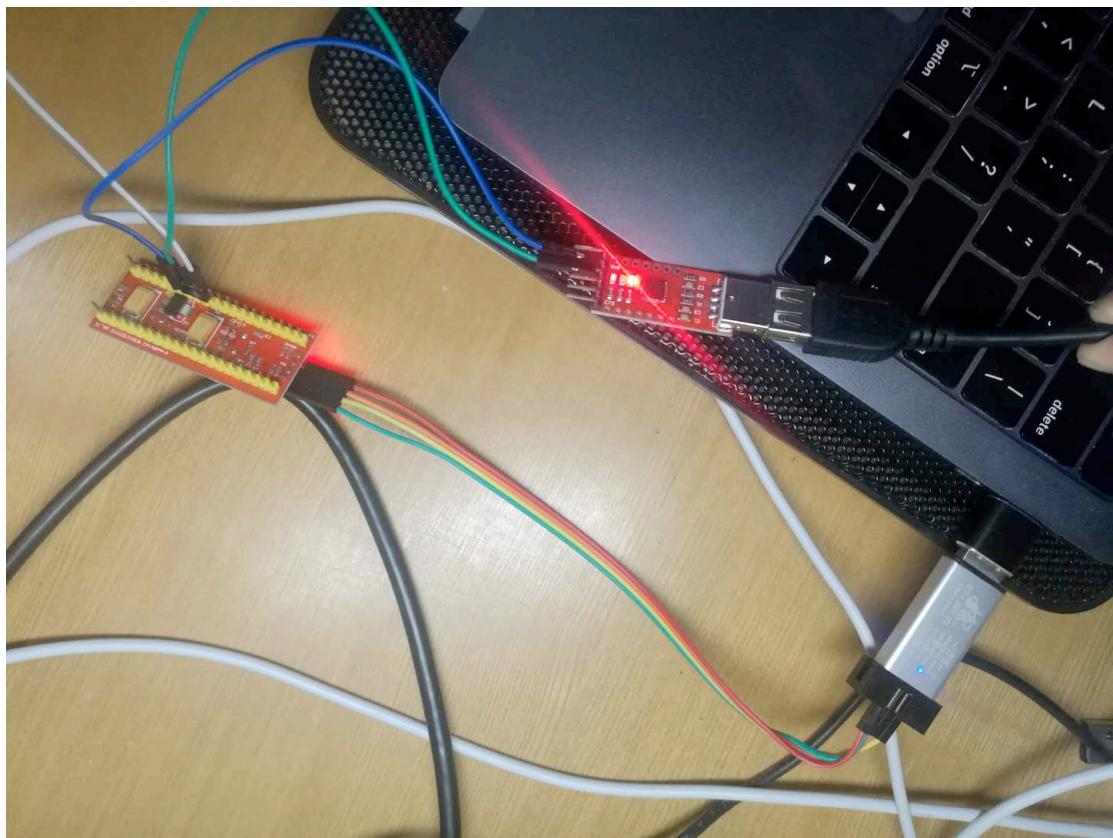
结果如下：



#### 四、实验结果分析

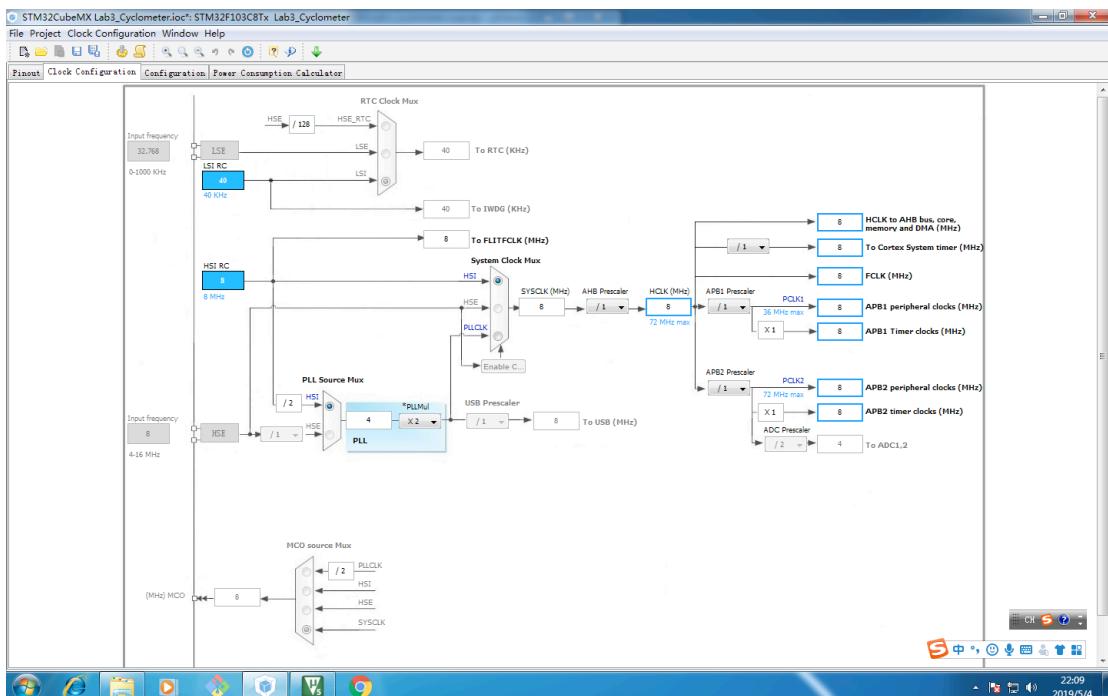
### 【实物连线图】

将转串口芯片和 STLink 同时连接 STM32:



## 【计数器参数计算】

如下图是单片机的时钟设置，系统时钟频率为 8MHz，外设的预分频参数都为 1：

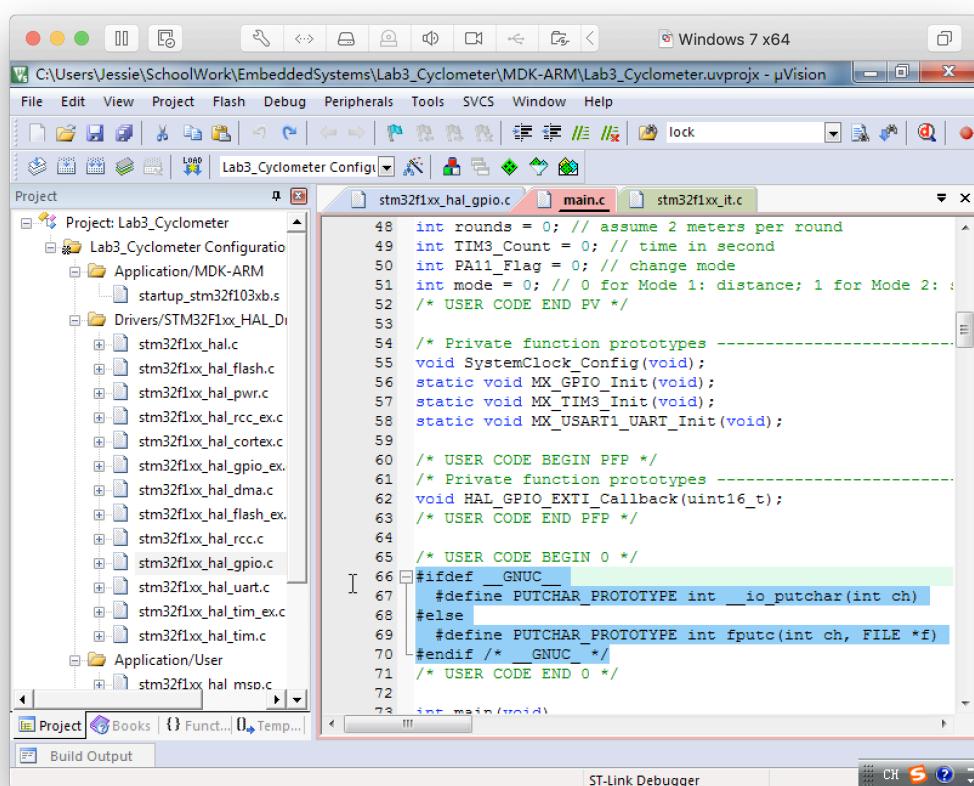


如果需要计数器是 200ms 每周期，则可以是每次 tick 相隔 1ms，每个周期 tick 次数为 200：

1. 每次 tick 时间 1ms，则需要预分频 8000 次，即频率为  $8000000/8000=1000\text{Hz}$ 。在 TIM3 设置里面的 Prescaler 表示从 0 开始直到多少复位，因此设置为 7999。
2. 每个周期 200 个 ticks，则从 0 开始直到 199 复位，因此 Counter Period 设置为 199。

### 【重定向 printf 函数】

为了方便编写串口输出语句，将 printf 函数重定向。首先要包含头文件 stdio.h，然后加入如下宏：



The screenshot shows the MDK-ARM IDE interface with the project 'Lab3\_Cyclometer' open. The left pane displays the project structure, and the right pane shows the 'main.c' source code editor. The code contains the following redirection logic:

```
48 int rounds = 0; // assume 2 meters per round
49 int TIM3_Count = 0; // time in second
50 int PA11_Flag = 0; // change mode
51 int mode = 0; // 0 for Mode 1: distance; 1 for Mode 2: :
52 /* USER CODE END PV */
53
54 /* Private function prototypes -----
55 void SystemClock_Config(void);
56 static void MX_GPIO_Init(void);
57 static void MX_TIM3_Init(void);
58 static void MX_USART1_UART_Init(void);
59
60 /* USER CODE BEGIN PFP */
61 /* Private function prototypes -----
62 void HAL_GPIO_EXTI_Callback(uint16_t);
63 /* USER CODE END PFP */
64
65 /* USER CODE BEGIN 0 */
66 #ifdef __GNUC__
67 #define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
68 #else
69 #define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
70#endif /* __GNUC__ */
71 /* USER CODE END 0 */
72
73 main()
74 {
75     /* USER CODE BEGIN 1 */
76 }
```

重定向：

The screenshot shows the MDK-ARM IDE interface on Windows 7 x64. The project is named 'Lab3\_Cyclometer'. The main.c file is open, displaying the following code:

```
243     UNUSED(GPIO_Pin);
244 }
245
246 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *h)
247 {
248     TIM3_Count++;
249 }
250
251 /**
252  * @brief Retargets the C library printf function to 1
253  * @param None
254  * @retval None
255 */
256
257 PUTCHAR_PROTOTYPE
258 {
259     HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 0xffff);
260     return ch;
261 }
262
263 /* USER CODE END 4 */
264
265 #ifndef USE_FULL_ASSERT
266
267 /**
268  * @brief Reports the name of the source file and the
269  *       line number if assert fails
270 */
271
272 /**
273  * @brief Prints error message using the C library printf
274  *       function
275 */
276
277 /**
278  * @brief Prints error message using the C library printf
279  *       function
280 */
281
282 /**
283  * @brief Prints error message using the C library printf
284  *       function
285 */
286
287 /**
288  * @brief Prints error message using the C library printf
289  *       function
290 */
291
292 /**
293  * @brief Prints error message using the C library printf
294  *       function
295 */
296
297 /**
298  * @brief Prints error message using the C library printf
299  *       function
300 */
301
302 /**
303  * @brief Prints error message using the C library printf
304  *       function
305 */
306
307 /**
308  * @brief Prints error message using the C library printf
309  *       function
310 */
311
312 /**
313  * @brief Prints error message using the C library printf
314  *       function
315 */
316
317 /**
318  * @brief Prints error message using the C library printf
319  *       function
320 */
321
322 /**
323  * @brief Prints error message using the C library printf
324  *       function
325 */
326
327 /**
328  * @brief Prints error message using the C library printf
329  *       function
330 */
331
332 /**
333  * @brief Prints error message using the C library printf
334  *       function
335 */
336
337 /**
338  * @brief Prints error message using the C library printf
339  *       function
340 */
341
342 /**
343  * @brief Prints error message using the C library printf
344  *       function
345 */
346
347 /**
348  * @brief Prints error message using the C library printf
349  *       function
350 */
351
352 /**
353  * @brief Prints error message using the C library printf
354  *       function
355 */
356
357 /**
358  * @brief Prints error message using the C library printf
359  *       function
360 */
361
362 /**
363  * @brief Prints error message using the C library printf
364  *       function
365 */
366
367 /**
368  * @brief Prints error message using the C library printf
369  *       function
370 */
371
372 /**
373  * @brief Prints error message using the C library printf
374  *       function
375 */
376
377 /**
378  * @brief Prints error message using the C library printf
379  *       function
380 */
381
382 /**
383  * @brief Prints error message using the C library printf
384  *       function
385 */
386
387 /**
388  * @brief Prints error message using the C library printf
389  *       function
390 */
391
392 /**
393  * @brief Prints error message using the C library printf
394  *       function
395 */
396
397 /**
398  * @brief Prints error message using the C library printf
399  *       function
400 */
401
402 /**
403  * @brief Prints error message using the C library printf
404  *       function
405 */
406
407 /**
408  * @brief Prints error message using the C library printf
409  *       function
410 */
411
412 /**
413  * @brief Prints error message using the C library printf
414  *       function
415 */
416
417 /**
418  * @brief Prints error message using the C library printf
419  *       function
420 */
421
422 /**
423  * @brief Prints error message using the C library printf
424  *       function
425 */
426
427 /**
428  * @brief Prints error message using the C library printf
429  *       function
430 */
431
432 /**
433  * @brief Prints error message using the C library printf
434  *       function
435 */
436
437 /**
438  * @brief Prints error message using the C library printf
439  *       function
440 */
441
442 /**
443  * @brief Prints error message using the C library printf
444  *       function
445 */
446
447 /**
448  * @brief Prints error message using the C library printf
449  *       function
450 */
451
```

## 五、讨论与心得

本次实验使用 STM32 核心板开发，由于我对单片机没有什么了解，也缺乏电子电路相关知识，这次实验完成得比较费力。好在参考资料丰富，HAL 库和配套的开发软件也很好用，虽然实验过程中遇到了各种奇怪的 bugs，但是最终通过查阅大量资料还是解决了问题。