FreeRTOS 学习之四：消息队列

前提：默认已经装好 MDK V5 和 STM32CubeMX，并安装了 STM32F1xx 系列的支持包。

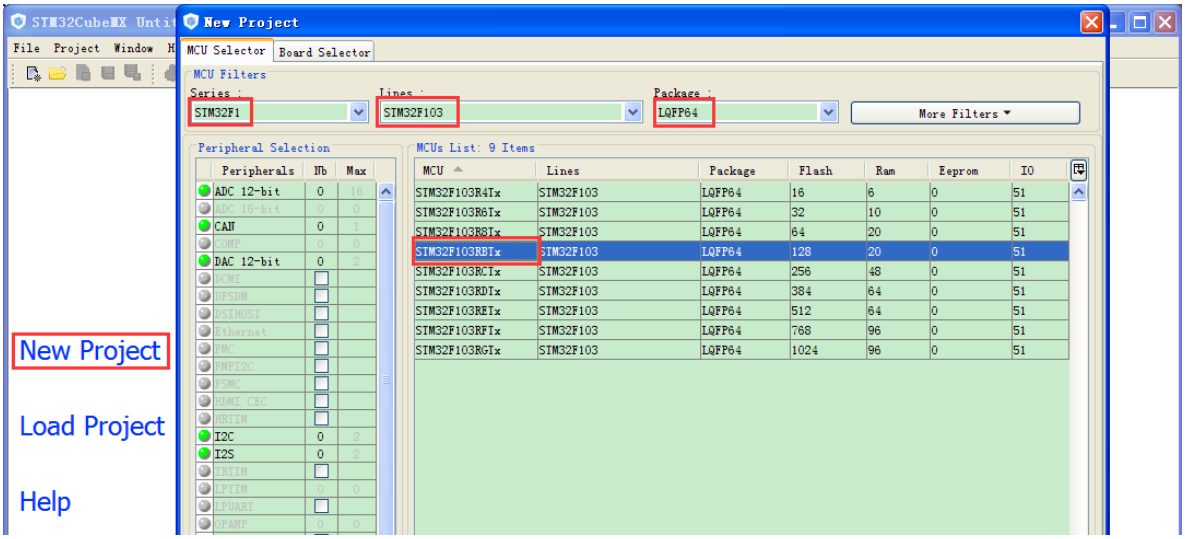
硬件平台：STM32F1xx 系列。

目的：学习消息队列的使用。

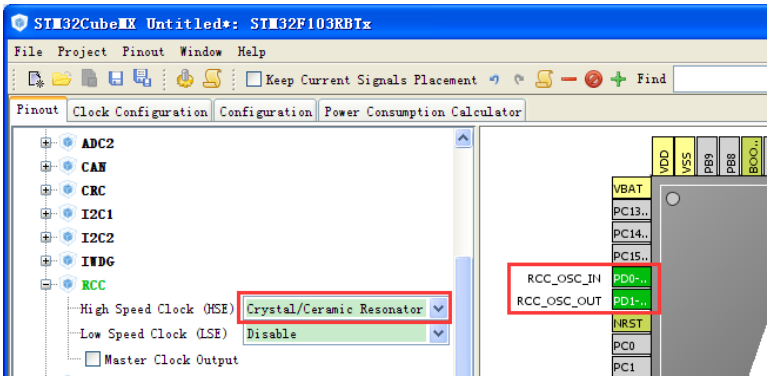
队列可以保存有限个具有确定长度的数据单元。通常情况下，队列被作为 FIFO(先进先出)使用，即数据由队列尾写入，从队列首读出。

本文例子使用 STM32CubeMX 配置创建两个任务，一个任务每隔一定时间发送一个消息到队列，另一个等待消息并根据消息的内容控制 LED 的闪烁次数。

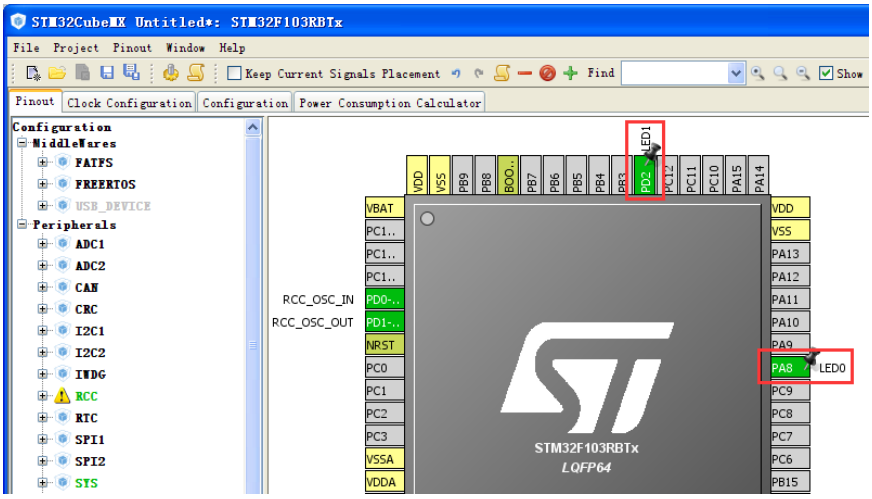
Step1.打开 STM32CubeMX，点击“New Project”，选择芯片型号，STM32F103RBTx。



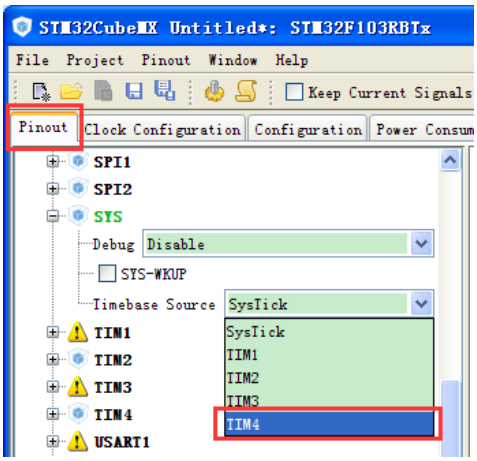
Step2.配置时钟引脚。



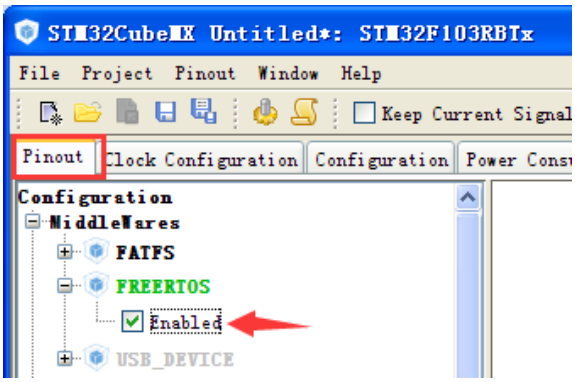
Step3.配置 PA8 和 PD2 为 Output，并把用户标签分别改为 LED0，LED1。



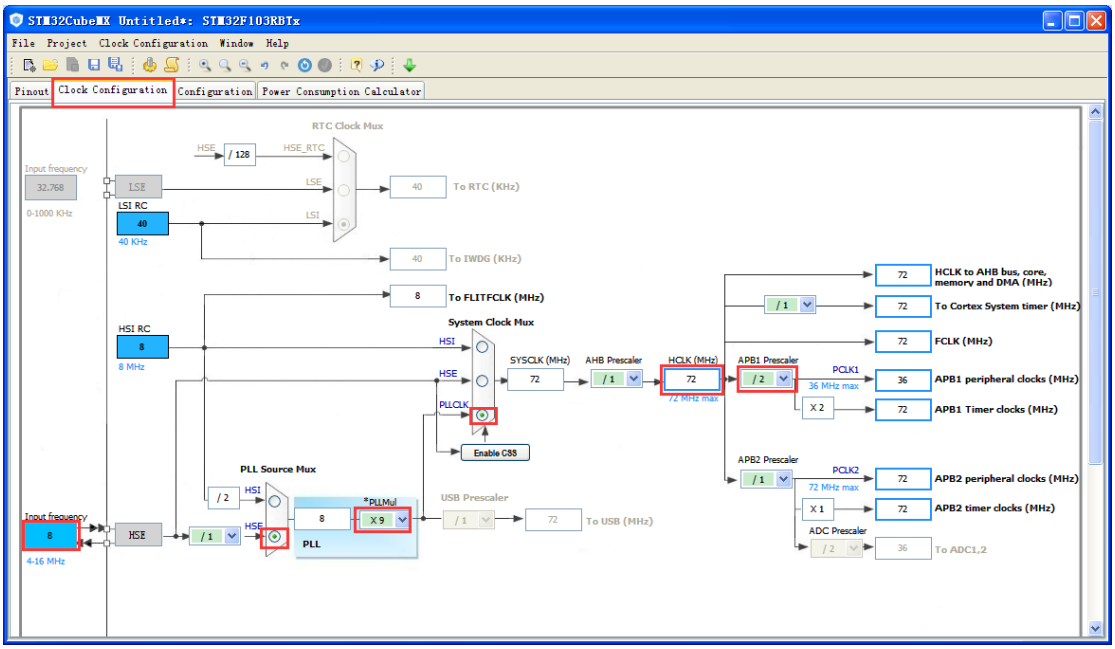
Step4.将系统时基源改为 TIM4。



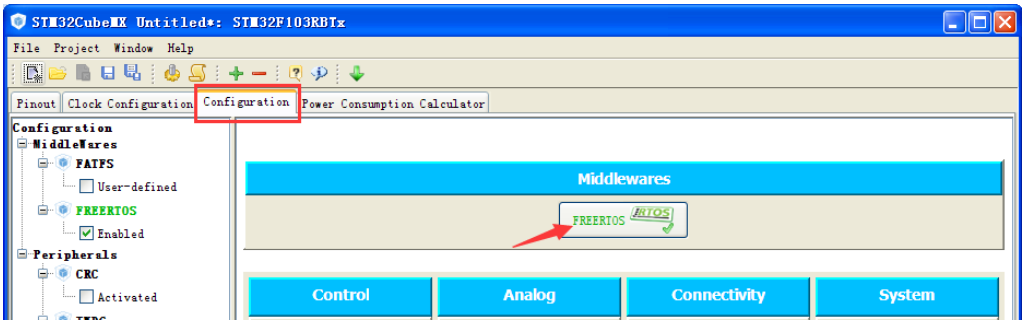
Step5.使能 FreeRTOS。



Step6.配置时钟树。8M 输入时，通过 PLL 得到 72M 内部时钟。

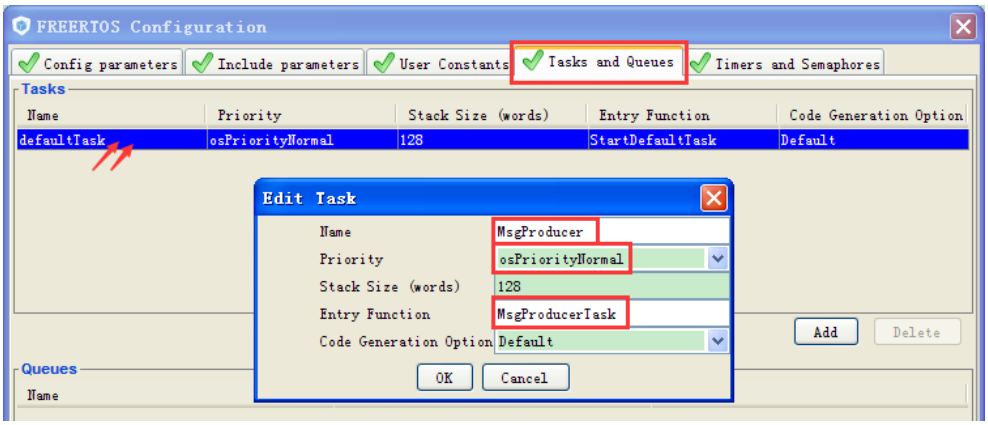


Step7.配置 FreeRTOS。

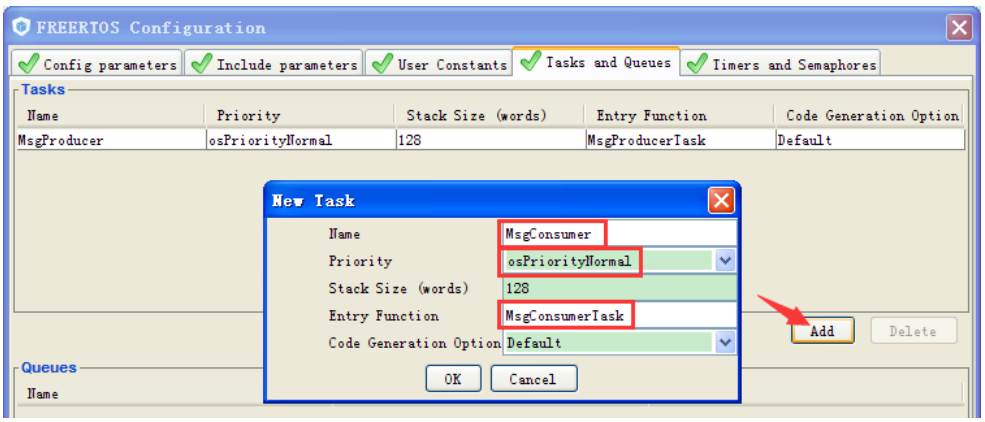


在 Tasks and Queues 选项卡，默认配置了一个名为 defaultTask 的任务，其优先级为普通，任务堆栈大小为 128 字，任务函数名为 StartDefaultTask。

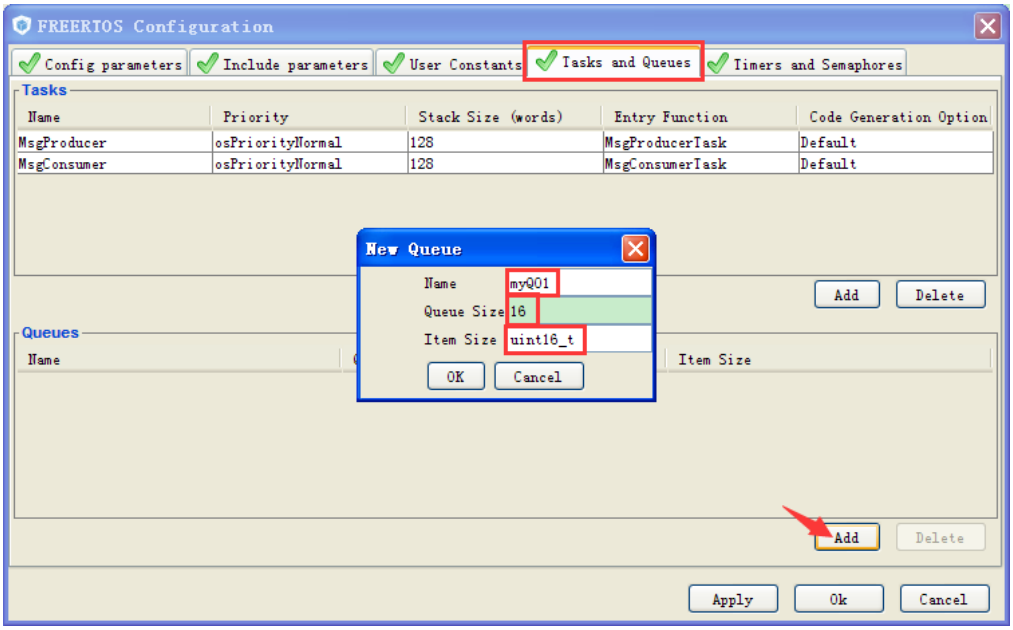
双击蓝色的地方，弹出对话框，将任务名修改为 MsgProducer，将任务函数名修改为 MsgProducerTask。



点击 Add 按钮，增加一个任务 MsgConsumer，优先级设置为 Normal，函数名为 MsgConsumerTask。

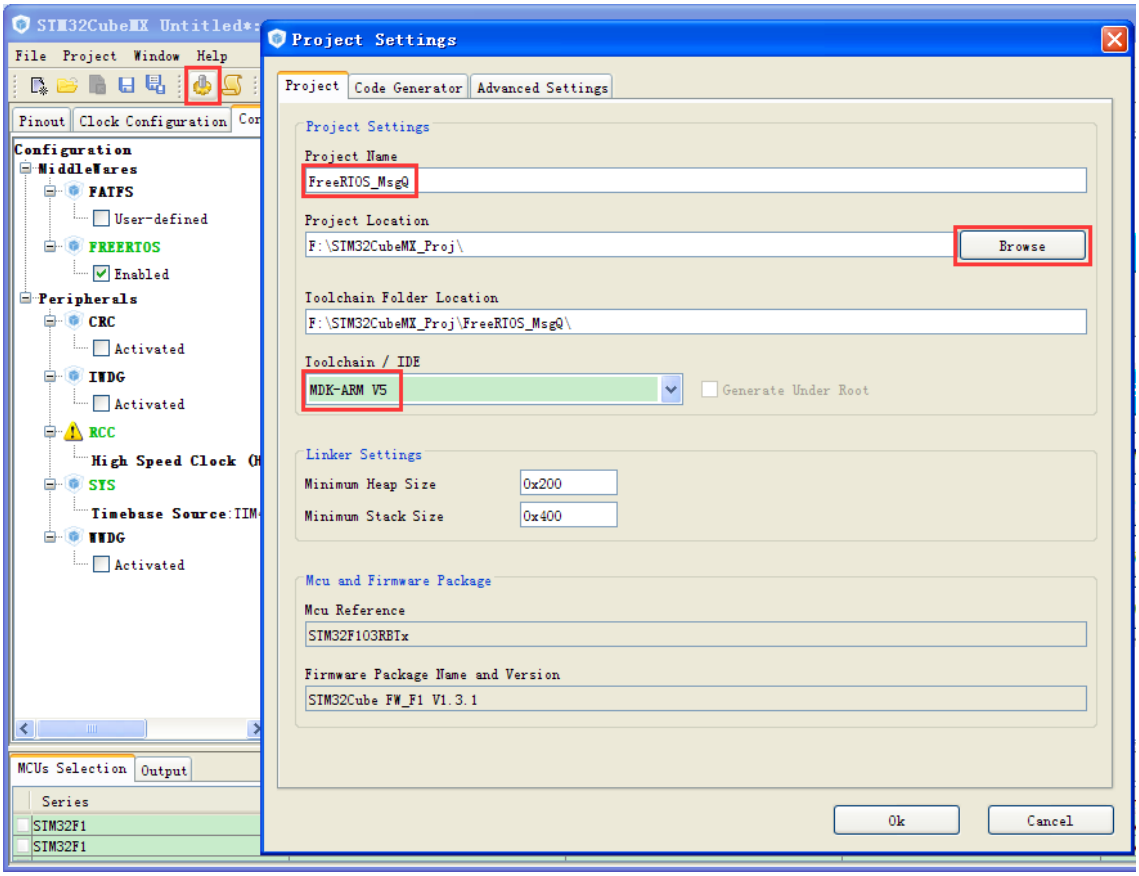


点击 Queues 栏的 Add 按钮，增加一个任务队列 myQ01，深度为 16，每个单元数据类型是 uint16_t。

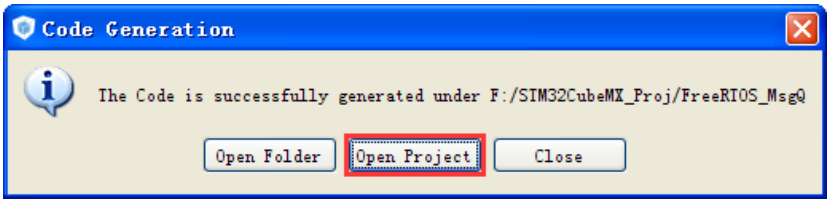


注：其他的都使用默认参数。

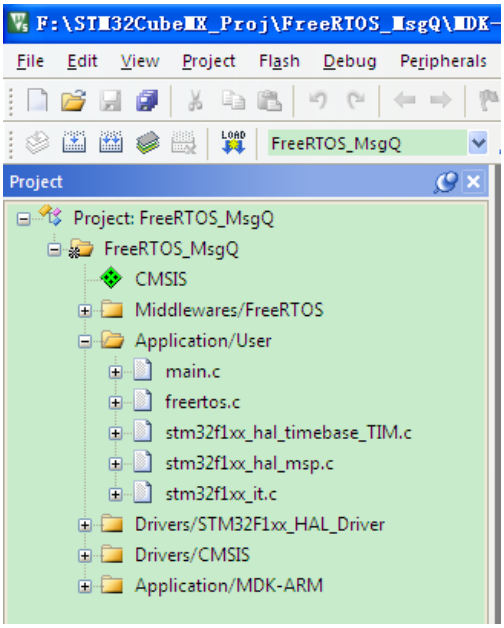
Step8.生成代码。



等完成后直接打开工程。



工程基本组织结构如下图，其中 Application/User 组中的文件是用户可以修改的，而其他组中的文件一般不进行修改。



Step9.分析程序结构。

在进入 main 函数之前，先定义了几个变量，声明了几个函数。

```
/* Private variables -----------------------
osThreadId MsgProducerHandle;
osThreadId MsgConsumerHandle;
osMessageQId myQ01Handle;

/* USER CODE BEGIN PV */
/* Private variables -----------------------

/* USER CODE END PV */

/* Private function prototypes -------------
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
void MsgProducerTask(void const * argument);
void MsgConsumerTask(void const * argument);
```

再看 main 函数。将 main 函数整理，删除很多注释之后，得到下图所示内容。

```
int main(void)
{
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();
  /* Configure the system clock */          ①
  SystemClock_Config();
  /* Initialize all configured peripherals */
  MX_GPIO_Init();

  /* Create the thread(s) */
  /* definition and creation of MsgProducer */
  osThreadDef(MsgProducer, MsgProducerTask, osPriorityNormal, 0, 128);
  MsgProducerHandle = osThreadCreate(osThread(MsgProducer), NULL);
  /* definition and creation of MsgConsumer */
  osThreadDef(MsgConsumer, MsgConsumerTask, osPriorityNormal, 0, 128);
  MsgConsumerHandle = osThreadCreate(osThread(MsgConsumer), NULL);
  /* Create the queue(s) */
  /* definition and creation of myQ01 */                              ②
  osMessageQDef(myQ01, 16, uint16_t);
  myQ01Handle = osMessageCreate(osMessageQ(myQ01), NULL);
  /* Start scheduler */
  osKernelStart();          ③

  /* We should never get here as control is now taken by the scheduler */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  }
}
```

其中第①部分，是硬件配置；第②部分，创建两个任务和一个消息队列；第③部分，启动调度器。

Step10.添加代码。

在 main.c 文件中，找到前面配置添加的两个任务函数，并在其中分别添加代码。

MsgProducerTask 的功能是，发送 1 次消息，间隔一秒后发送 1 次，再间隔一秒发送 1 次，然后等待 2 秒。

```
/* MsgProducerTask function */
void MsgProducerTask(void const * argument)
{

  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
      osDelay(1000);
      osMessagePut(myQ01Handle, 1, osWaitForever);  // 发送消息到队列
      osDelay(1000);
      osMessagePut(myQ01Handle, 3, osWaitForever);  // 发送消息到队列
      osDelay(1000);
      osMessagePut(myQ01Handle, 5, osWaitForever);  // 发送消息到队列
      osDelay(2000);
  }
  /* USER CODE END 5 */
}
```

MsgConsumerTask 的功能的，等待消息，然后根据消息的内容控制 LED0 和 LED1 闪烁次数。

```
/* MsgConsumerTask function */
void MsgConsumerTask(void const * argument)
{
  /* USER CODE BEGIN MsgConsumerTask */
  osEvent event;
  /* Infinite loop */
  for(;;)
  {
      event = osMessageGet(myQ01Handle, osWaitForever);
      if(event.status == osEventMessage)
      {
          while(event.value.v--)
          {
              HAL_GPIO_WritePin(LED0_GPIO_Port, LED0_Pin, GPIO_PIN_RESET);  //LED0亮
              HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);  //LED1亮
              osDelay(100);
              HAL_GPIO_WritePin(LED0_GPIO_Port, LED0_Pin, GPIO_PIN_SET);    //LED0灭
              HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_SET);    //LED1灭
              osDelay(100);
          }
      }
  }
  /* USER CODE END MsgConsumerTask */
}
```

Step11.编译下载运行。现象是，LED 闪 1 次，一秒后闪 3 次，再过一秒闪 5 次，再等三秒，LED 闪 1 次…如此循环。

程序分析：

消息队列和计数信号量相比，队列可以传递更多的信息。如果把消息的内容忽略，那么队列实现的实际上就是计数信号量的功能。

S.D.Lu 于 深圳
2016 年 8 月