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。

STE32CubeEX Untitled*: STE32F103RBTx						
File Project Window Help	File Project Window Help					
i 🖸 😂 🐘 🖯 🤐 🤌 💁 i 🔶 💻 🕐 🖗 🗳						
Pinout Clock Configuration Configuration Power Consumption Calculator						
Configuration WiddleTares FATFS User-defined		-	Middle	ewares		
FREERTOS		FREERIOS				
Peripherals						
Activated		Control	Analog	Connectivity	System	

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。

FREERIOS Configuration						
🖋 Config parameters 🖋 Include parameters 🖋 User Constants 🖋 Iasks and Queues 🖋 Iimers and Semaphores						
Tasks						
Name	Priority	Stack Size (words)	Entry Function	Code Generation Option		
defaultTask 🖉 🖉	osPriorityNormal	128	StartDefaultTask	Default		
	Edit Task					
	Name	MsgProduce	r			
	Priority	osPriority	Normal 🛛 👻			
	Stack Siz	e (words) 128				
	Entry Fun	ction MsgProduce	rTask —			
	Code Gene	ration Option Default	~	Add Delete		
Queues		OK Cancel				
Name						

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

♥ FREERIOS Configuration							
🖋 Config parameters 🖋 Include parameters 🖋 User Constants 🖋 Iasks and Queues 🖋 Timers and Semaphores							
Tasks							
Name	Priority	Stack Size (w	vords) l	Entry Funct	tion	Code Generation	Option
MsgProducer	osPriorityNormal	128	Mis	sgProducerI	ask	Default	
	New Task Name		MsgConsumer				
Priority		r –	osPriorityNo	rmal	×		
Stack Size		ze (words)	128				
Entry Function		unction	MsgConsumerT	ask			
	Code Ger	meration Option	Default		~	Add De	Lete
Queues		OK	Cancel				
Name							

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。

💿 FREERTOS Configu	ration			×		
🖋 Config parameters 🛛 🖋 Include parameters 🛛 🖋 User Constants 📝 Tasks and Queues 🚀 Timers and Semaphores						
Name	Priority	Stack Size (words)	Entry Function	Code Generation Option		
MsgProducer	osPriorityNormal	128	MsgProducerTask	Default		
MsgConsumer	osPriorityNormal	128	MsgConsumerTask	Default		
PQueues ITame		Queue Name myQ01 Queue Size 16 Item Size uint16_t OK Cancel	Item Size	Add Delete		
			-11-7			

注: 其他的都使用默认参数。 Step8.生成代码。

STE32CubeEX Untitled*:	Project Settings	
File Project Window Help	o i roject bettings	
📭 📂 🖿 🖶 🖳 🚺 🚺	Project Code Generator Advanced Settings	_
Pinout Clock Configuration Cor	Project Settings	
Configuration	Project Name	
□ MiddleWares	FreeRIOS Mse0	1
FATFS		-
User-defined	Project Location	
E- I FREERTOS	F:\SIM32CubeMI_Proj\ Browse	
- Enabled		
- Peripherals	Icolchain Folder Location	
E CRC	F:\SIM32CubeMI_Proj\FreeRIOS_MsgQ\	
Activated	Tealchain / TDF	
🖨 💿 INDG		
Activated	Generate Under Koot	
High Speed Clock (H	Clinker Settings	
	Minimum Heap Size 0x200	
Timebase Source:IIM	Minimum Stack Size 0x400	
🖨 💿 TYDG		
Activated		
	Mcu and Firmware Package	
	Mcu Reference	
	SIM32F103RBTx	
	Firmware Package Name and Version	
	STM32Cube FW_F1 V1.3.1	
MCUs Selection Output		
Series		
STM32F1	0k Cancel	
SIM32F1		_

等完成后直接打开工程。

© Code Generation	
Ihe Code is successfully generated under F:/SIM32CubeMX_Pro	j/FreeRIOS_MsgQ
Open Folder Open Project Close	

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。

```
41 /* Private variables -
42
   osThreadId MsgProducerHandle;
43 osThreadId MsgConsumerHandle;
44 osMessageQId myQ01Handle;
45
46
   /* USER CODE BEGIN PV */
   /* Private variables -
47
48
49
   /* USER CODE END PV */
50
   /* Private function prototypes
51
52
   void SystemClock_Config(void);
53
   static void MX_GPIO_Init(void);
   void MsgProducerTask(void const * argument);
54
55 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。



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。

```
170 /* MsgProducerTask function */
171
    void MsgProducerTask(void const * argument)
172 🕀 {
173
174
      /* USER CODE BEGIN 5 */
      /* Infinite loop */
175
176
      for(;;)
177 E
      {
178
          osDelav(1000);
179
          osMessagePut(myQ01Handle, 1, osWaitForever); // 发送消息到队列
180
          osDelav(1000);
          osMessagePut(myQ01Handle, 3, osWaitForever); // 发送消息到队列
181
182
          osDelay(1000);
183
          osMessagePut(myQ01Handle, 5, osWaitForever); // 发送消息到队列
184
          osDelay(2000);
185
      3
186
       /* USER CODE END 5 */
187
```

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。

```
189 /* MsgConsumerTask function */
190 void MsgConsumerTask(void const * argument)
191 🕀 {
       /* USER CODE BEGIN MsgConsumerTask */
192
193
      osEvent event;
194
       /* Infinite loop */
195
     for(;;)
196 ¢
     {
197
           event = osMessageGet(myQ01Handle, osWaitForever);
198
           if(event.status == osEventMessage)
199 E
           {
200
                while(event.value.v--)
201 🗄
                {
202
                    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
203
204
                    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
205
206
207
                    osDelay(100);
208
                3
209
           }
210
       /* USER CODE END MsgConsumerTask */
211
212 }
```

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