No décimo artigo da série “A Blue Pill”, daremos continuidade no estudo. A dica é acompanhar a série desde o primeiro artigo porque é um caminho estruturado, onde a sequência traz benefícios no aprendizado (linha de raciocínio). Nesse artigo basicamente concluiremos o oitavo artigo, analisando os detalhes de configuração da FreeRTOS. Será um artigo bem curto, mas, importante para concluirmos sobre as macros do FreeRTOS.

FreeRTOSConfig.h

Cada projeto, no subdiretório ~/stm32f103c8t6/rtos, possui sua própria cópia do FreeRTOSConfig.h, que configura os recursos do RTOS e outros fatores que podem variar de acordo com o projeto. Isso permite que alguns projetos deixem de fora os recursos do FreeRTOS que eles não precisam, resultando num executável menor. Em outros casos, pode haver diferenças de clock, de alocação de memória e outros recursos relacionados ao RTOS.

O arquivo de cabeçalho FreeRTOS.h, faz com que o arquivo FreeRTOSConfig.h local seja incluído. Examinaremos alguns dos elementos de configuração importantes do arquivo FreeRTOSConfig.h, mostrados a seguir.

```
#define   configUSE_PREEMPTION              1
#define   configUSE_IDLE_HOOK               0
#define   configUSE_TICK_HOOK               0
#define   configCPU_CLOCK_HZ                ((unsigned long) 72000000)
#define   configSYSTICK_CLOCK_HZ            (configCPU_CLOCK_HZ/8)
#define   configTICK_RATE_HZ                ((TickType_t) 250)
#define   configMAX_PRIORITIES              (5)
#define   configMINIMAL_STACK_SIZE          (unsigned short) 128)
#define   configTOTAL_HEAP_SIZE             ((size_t) (17*1024))
#define   configMAX_TASK_NAME_LEN           (16)
#define   configUSE_TRACE_FACILITY          0
#define   configUSE_16_BIT_TICKS            0
#define   configIDLE_SHOULD_YIELD           1
#define   configUSE_MUTEXES                 0
#define   configCHECK_FOR_STACK_OVERFLOW    1
```

A mais importante dessas macros de configuração é a macro configUSE_PREEMPTION. Quando definido como diferente de zero, indica que queremos um agendamento preemptivo no FreeRTOS. Há duas funções HOOK, que não foram usadas, portanto, configUSE_IDLE_HOOK e configUSE_TICK_HOOK estão definidas como zero.

Temos três macros que configuram o FreeRTOS para calcular o tempo correto. A declaração CPU_CLOCK_HZ configura um clock de CPU de 72MHz. A declaração SYSTICK_CLOCK_HZ configura um contador de tempo do sistema que incrementa a cada 8 ciclos da CPU. A declaração TICK_RATE_HZ configura uma interrupção no sistema a cada 250 vezes por segundo, portanto, a cada 4ms. Se esses valores não estiverem corretos, o FreeRTOS não operará com os clocks e atrasos corretos.

O valor de configMAX_PRIORITIES define o número máximo de prioridades que serão suportadas. No RTOS cada nível de prioridade requer RAM, portanto, os níveis não devem ser definidos acima do necessário. O tamanho mínimo da pilha, em words, especifica quanto espaço uma tarefa inativa

aloca com o FreeRTOS. Isso normalmente não deve ser modificado. A declaração TOTAL_HEAP_SIZE define, em bytes, quanta RAM pode ser alocada dinamicamente. Nesse exemplo, os 17K de SRAM do total de 20K estão disponíveis como HEAP.

A macro configIDLE_SHOULD_YIELD deve ser ativada se você desejar que a tarefa ociosa chame outra tarefa pronta para execução. Por último, a macro configCHECK_FOR_STACK_OVERFLOW foi ativada nesse exemplo, para podermos demonstrar a função vApplicationStackOverflowHook(). Se você não precisar dessa funcionalidade, desative-a, definindo-a como zero.

As macros a seguir são exemplos de outras personalizações. Não usaremos a função vTaskDelete(), portanto, INCLUDE_vTaskDelete é definido como zero. Isso reduz a sobrecarga do código FreeRTOS compilado. Precisamos da função vTaskDelay(), então, a macro INCLUDE_vTaskDelay deve ser configurada como 1.

```
#define   INCLUDE_vTaskPrioritySet            0
#define   INCLUDE_uxTaskPriorityGet           0
#define   INCLUDE_vTaskDelete                 0
#define   INCLUDE_vTaskCleanUpResources       0
#define   INCLUDE_vTaskSuspend                0
#define   INCLUDE_vTaskDelayUntil             0
#define   INCLUDE_vTaskDelay                  1
```

Convenção de nomenclatura do FreeRTOS

A convenção de nomenclatura do FreeRTOS difere da usada pela biblioteca libopencm3. O grupo FreeRTOS usa uma convenção de nomenclatura exclusiva para variáveis, macros e funções. Não é recomendada a prática de incluir informações de tipo em entidades nomeadas. O problema é que os tipos podem mudar à medida que o projeto amadurece, ou é portado para uma nova plataforma. Quando isso acontece, você se depara com duas opções:

- ✔ Deixe os nomes das entidades como estão e conviva com o fato de que as informações do tipo não estão corretas.
- ✔ Edite todas as referências de nome para refletir o novo tipo.

Apesar dessa convenção de nomenclatura ser estranha, não perca tempo reescrevendo o FreeRTOS ou colocando camadas em torno dele. Aqui, simplesmente identificaremos as convenções que eles usaram, para que seja mais fácil usar a API deles.

Prefixo	Descrição
v	void (função que retorna valor)
c	tipo char
s	tipo short
l	tipo long
x	BaseType_t e para outro tipo não coberto
u	tipo unsigned
p	ponteiro

Tabela 1 – Prefixo de tipos FreeRTOS

Portanto, se a variável tiver um tipo ‘unsigned char’, será usado o prefixo ‘uc’. Se a variável for um ponteiro para um ‘unsigned char’, será usado ‘puc’. Você já deve ter visto a função denominada vTaskDelay(), que indica que não há valor de retorno (void). A função FreeRTOS denominada xQueueReceive() retorna um tipo BaseType_t, e é por isso que o prefixo do nome da função é "x".

Macros FreeRTOS

O FreeRTOS escreve nomes de macro com um prefixo para indicar onde eles estão definidos. A tabela a seguir detalha esses prefixos.

Prefixo	Exemplo	Arquivo Fonte
port	portMAX_DELAY	portable.h
task	taskENTER_CRITICAL()	task.h
pd	pdTRUE	projdefs.h
config	configUSE_PREEMPTION	FreeRTOSConfig.h
err	errQUEUE_FULL	projdefs.h

Tabela 2 – Prefixo de macros FreeRTOS

Conforme a proposta inicial estamos evoluindo no aprendizado, portanto, aqui concluo o décimo artigo da série.