

AG32 下使用 freeRTOS 的参考

在 SDK 下已经有移植好的 FreeRTOS Kernel V10.4.6 版本，可供用户使用。

源码位于：\AgRV_pio\packages\framework-agrv_freertos\

用户使用时，不必关注该源码，只需关注 API 使用即可。

一、简单验证：

使用时，从 VSCODE 中直接打开工程：\AgRV_pio\platforms\AgRV\examples\freeRTOS
打开工程后，可以先验证简单的运行情况。

这个工程中，并没有 ve 文件（而是使用了默认的 ve 文件）。

便于后续开发，可以先建立一份 ve。

步骤：

1. 从 example 路径下 copy 一份 example_board.ve 过来；
2. 打开该 example_board.ve，删除掉里边除 clk 和 led 灯以外的其他引脚配置；
3. 在 platformio.ini 中添加对该 ve 的引用：board_logic.ve = example_board.ve

接下来，编译 ve 并烧录，然后编译 code 并烧录。

两项都烧录成功后，就可以看到 led 灯的闪烁了。

二、使用样例：

打开 main.c，可以看到在 main 函数中使用了几种元素：xQueue、xSemaphore、xTask、xTimer。

同时在 main.c 中对接的 hook 回调中，会配合使用这些元素。

这里注意，如果要精简样例，比如只跑一个 task 时，删除其他元素时，要同时把 hook 里对应的调用也删除。

main() 函数中，可精简到调用 3 个函数：

```
int main(void)
{
    prvSetupHardware();
```

```

xTaskCreate(led_task,
            "led_task",
            configMINIMAL_STACK_SIZE,
            NULL,
            mainEVENT_SEMAPHORE_TASK_PRIORITY,
            &xGPTimerTask);
vTaskStartScheduler();
for(;;);
}

```

其中，prvSetupHardware()函数中有 init 函数是必须的：

```

static void prvSetupHardware(void)
{
    /* Setup the clocks, etc. here, if they were not configured before
    | main() was called. */
    board_init();
    INT_DisableIntGlobal(); // Disable all interrupts. Let the OS handle the

    /*
    | // Initialize GPTIMER
    | PERIPHERAL_ENABLE(GPTIMER, 0);
    | INT_EnableIRQ(GPTIMER0_IRQn, PLIC_MAX_PRIORITY);
    | GPTIMER_EnableIntUpdate(GPTIMER0);
    | GPTIMER_InitTypeDef init;
    | GPTIMER_StructInit(&init);
    | init.Autoreload = SYS_GetSysClkFreq() / 1000 * mainGPTIMER_PERIOD_MS;
    | GPTIMER_Init(GPTIMER0, &init);
    | GPTIMER_EnableCounter(GPTIMER0);
    | */
}

```

在 led_task 中可写闪灯代码如下：

```

void led_task()
{
    while (1)
    {
        vTaskDelay(1000);
        GPIO_Toggle(LED_GPIO, GPIO_BIT1);
        GPIO_Toggle(LED_GPIO, GPIO_BIT2);
        GPIO_Toggle(LED_GPIO, GPIO_BIT3);
        GPIO_Toggle(LED_GPIO, GPIO_BIT4);
    }
}

```

另外，由于不再使用 xEventSemaphore，也需要在 vApplicationTickHook 中去除对 xSemaphore 的操作：

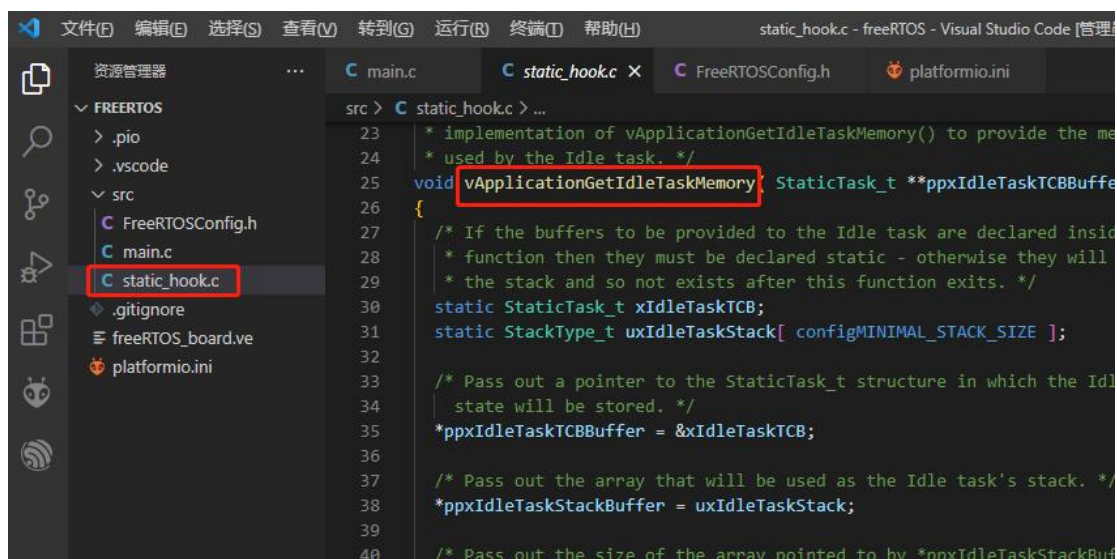
```
void vApplicationTickHook(void)
{
    return;
}
```

可尝试编译运行，查看效果。

除了以上简化过程，还有用户会使用到 **freeRTOS** 的静态方法。

如果使用静态方法，即打开了宏：configSUPPORT_STATIC_ALLOCATION，需要新增两个 hook 函数，可参考：https://blog.csdn.net/m0_46451722/article/details/113053257 中的描述。

使用如：



编译通过后，可在函数中使用 **static** 那组函数。

用法和动态那组函数相似，注意参数使用静态分配好的即可。

三、使用自建工程：

如果用户希望在自建工程中使用 **freeRTOS**，该如何把它添加进来？

比如，在 **example** 下要把 **freeRTOS** 加进来。步骤：

1. 在 **platformio.ini** 中增加对 **freertos** 的引用：

```
framework = agrv_sdk, agrv_freertos
program = agm_example
```

注意，多个库之间用“逗号+空格”来隔开。

2. 确认 **ve** 文件里的配置正常（时钟+led 引脚）；
3. 在 **example.c** 中引入 **freertos** 的头文件及用到的宏定义：

```

/* Kernel includes. */
#include "FreeRTOS.h" /* Must come first. */
#include "task.h"      /* RTOS task related API prototypes. */
#include "queue.h"     /* RTOS queue related API prototypes. */
#include "timers.h"    /* Software timer related API prototypes. */
#include "semphr.h"    /* Semaphore related API prototypes. */

#include "board.h"

/* Priorities at which the tasks are created. The event semaphore task is
   given the maximum priority of (configMAX_PRIORITIES - 1) to ensure it r
   soon as the semaphore is given. */
#define mainQUEUE_RECEIVE_TASK_PRIORITY    (tskIDLE_PRIORITY + 2)
#define mainQUEUE_SEND_TASK_PRIORITY      (tskIDLE_PRIORITY + 1)
#define mainEVENT_SEMAPHORE_TASK_PRIORITY (configMAX_PRIORITIES - 1)

```

4. 在 example.c 中新增几个函数:

```

static TaskHandle_t xLedTask;
void led_task()
{
    while (1)
    {
        vTaskDelay(1000);
        GPIO_Toggle(LED_GPIO, GPIO_BIT1 | GPIO_BIT2 | GPIO_BIT3 | GPIO_BIT4);
    }
}
void vApplicationTickHook(void)
{
    return;
}

```

5. 在 main.c 中 init 后直接启动 freeRTOS:

```

int main(void)
{
    // This will init clock and uart on the board
    board_init();

    xTaskCreate(led_task,
               "led_task",
               configMINIMAL_STACK_SIZE,
               NULL,
               mainEVENT_SEMAPHORE_TASK_PRIORITY,
               &xLedTask);
    vTaskStartScheduler();
    for(;;);
}

```

然后，编译 ve 并烧录，再编译 code 并烧录，就可以看到 led 的闪烁了。