ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Hệ nhúng (Embedded Systems)

Đỗ Công Thuần

Bộ môn Kỹ thuật Máy tính

Email: thuandc@soict.hust.edu.vn

Chương 9:
RTOS và FreeRTOS

Tại sao cần RTOS?

- CMSIS và HAL lib:
  - Giao tiếp với phần cứng/ngoại vi được thực hiện trên các API, driver đã chuẩn hóa.
  - Tương thích tốt giữa các dòng chip hoặc các thiết kế mạch khác nhau.
- Vấn đề:
  - Vẫn chỉ xây dựng được các ứng dụng đơn lẻ, giống như trên hệ 8 bit
  - Main loop + interrupt
  - Không tận dụng hết tài nguyên tính toán của CPU

➔ cần OS để hỗ trợ mô hình thực thi đa luồng

Key Requirements for RTOS

1. Predictable OS timing behavior
   - upper bound on the execution time of OS services
   - short times during which interrupts are disabled,
   - contiguous files to avoid unpredictable head movements
2. OS must manage the timing and scheduling
   - OS possibly has to be aware of task deadlines; (unless scheduling is done off-line).
   - OS must provide precise time services with high resolution.
   - all the requirements mentioned so far would be useless if it were very slow
3. OS must be fast
   - all the requirements mentioned so far would be useless if it were very slow

RTOS Classification

- Fast kernels
  - Example: QNX, PDOS, VCOS, VxWORKS, FreeRTOS (Open-source)
- Standard OS with real-time extensions
  - Example: RT-Linux

FreeRTOS

- FreeRTOS is a real-time kernel on top of which embedded applications can be built to meet their hard real-time requirements.
- Applications can be organized as a collection of independent threads of execution.

FreeRTOS

- Real-time OS cho hệ nhúng kích thước nhỏ - vừa
- Mã nguồn mở, phát triển bởi Richard Barry (2003)
- Maintain bởi Real Time Engineers Ltd
- Mua lại bởi Amazon (2017)
- Hỗ trợ nhiều dòng chip:
    - ARM (ARM7, ARM9, Cortex-M3, Cortex-M4, Cortex-A), Atmel AVR, AVR32, HCS12, MicroBlaze, Cortus (APS1, APS3, APS3R, APS5, FPF3, FPS6, FPS8), MSP430, PIC, Renesas H8/S, SuperH, RX, x86, 8052, Coldfire, V850, 78K0R, Fujitsu MB91460 series, Fujitsu MB96340 series, Nios II, Cortex-R4, TMS570, RM4x, Espressif ESP32, RISC-V

Why Used a Real-time Kernel?

- Ensuring an application meets its processing deadlines.
- Abstracting away timing information
- Maintainability/Extensibility
- Modularity
- Team development
- Easier testing
- Code reuse
- Imported efficiency
- Idle time
- Power management
- Flexible interrupt handling
- Mixed processing requirements

PreeRTOS Features

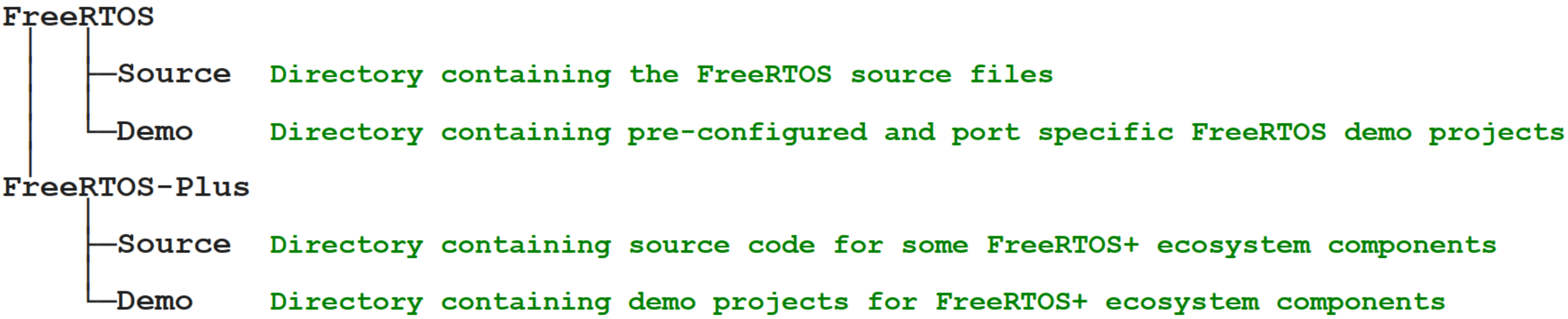
- Pre-exemptive or co-operative operation
- Very flexible task priority assignement
- Flexible, fast and light weight task notification mechanism
- Queues
- Binary semaphores
- Counting semaphores
- Mutexes
- Recursive Mutexes
- Software timers
- Event groups
- Tick hook functions
- Idle hook functions
- Stack overflow checking
- Trace recording
- Task run-time statistics gathering
- Optional commercial licensing and support
- Full interrupt nesting model (for some architectures)
- A tick-less capability for extreme low power applications
- Software managed interrupt stack when appropriate (this can help save RAM)

FreeRTOSConfig.h

- FreeRTOSConfig.h is used to tailor FreeRTOS for use in a specific application.
- Every demo application contains a FreeRTOSConfig.h file.
  - Never necessary to create a FreeRTOSConfig.h file from scratch.
  - It is recommended to start with, then adapt, the FreeRTOSConfig.h used by the demo application provided for the FreeRTOS port in use.

FreeRTOS Distribution

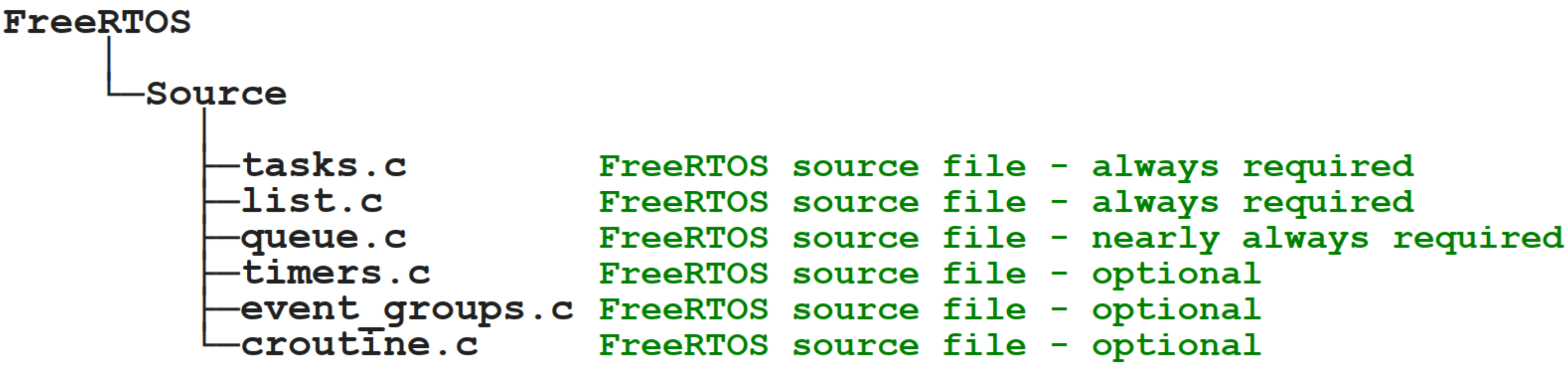
- Source code for all the FreeRTOS ports
    - Each supported combination of compiler and processor is considered to be a separate FreeRTOS port.
- Project files for all the FreeRTOS demo applications



```
FreeRTOS
 │   │
 │   ├─Source   Directory containing the FreeRTOS source files
 │   │
 │   └─Demo     Directory containing pre-configured and port specific FreeRTOS demo projects
 │
FreeRTOS-Plus
     │
     ├─Source   Directory containing source code for some FreeRTOS+ ecosystem components
     │
     └─Demo     Directory containing demo projects for FreeRTOS+ ecosystem components
```

Core FreeRTOS Source Files

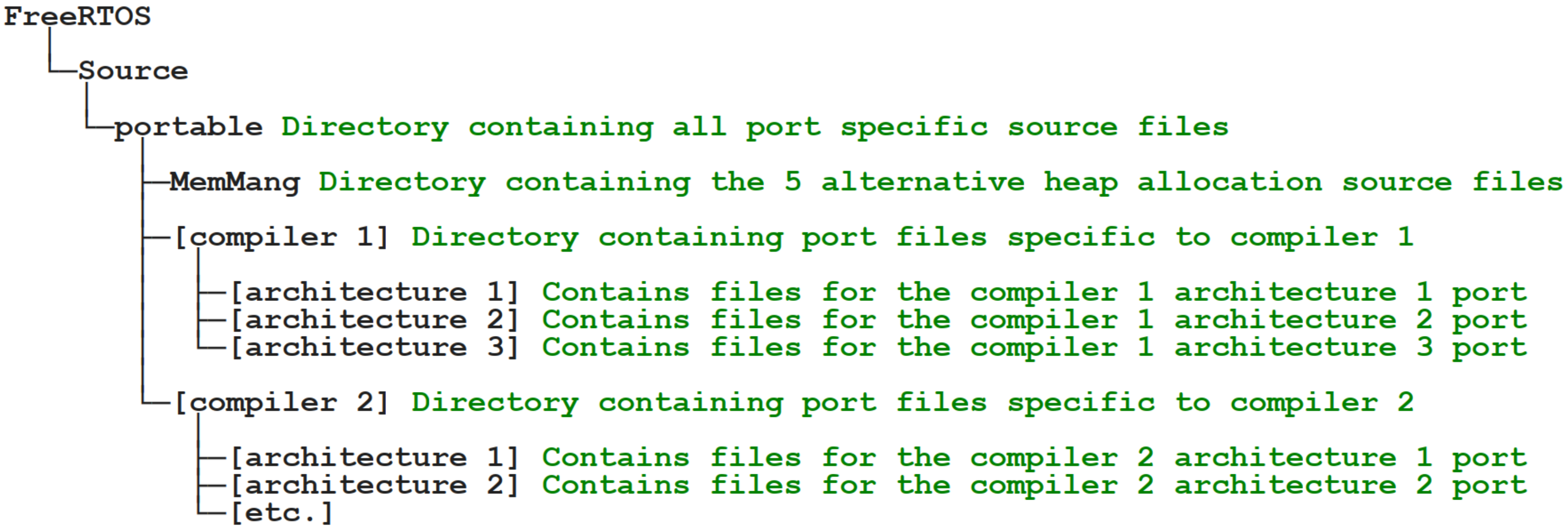
- **task.c** and **list.c** are common to all the FreeRTOS ports
- **queue.c** provides both queue and semaphore services
- **timers.c** provides software timer functionality
- **event_groups.c** provides event group functionality
- **croutine.c** implements the FreeRTOS co-routine functionality
    - Co-routines were intended for use on very small microcontrollers, are rarely used now



```
FreeRTOS
    │
    └─Source
        │
        ├─tasks.c         FreeRTOS source file - always required
        ├─list.c          FreeRTOS source file - always required
        ├─queue.c         FreeRTOS source file - nearly always required
        ├─timers.c        FreeRTOS source file - optional
        ├─event_groups.c  FreeRTOS source file - optional
        └─croutine.c      FreeRTOS source file - optional
```

FreeRTOS Source Files Specific to a Port

- Source files specific to a FreeRTOS port are contained within the FreeRTOS/Source/portable directory.
- The portable directory is arranged as a hierarchy, first by compiler, then by processor architecture.



```
FreeRTOS
 │
 └─Source
    │
    └─portable Directory containing all port specific source files
       │
       ├─MemMang Directory containing the 5 alternative heap allocation source files
       │
       ├─[compiler 1] Directory containing port files specific to compiler 1
       │   │
       │   ├─[architecture 1] Contains files for the compiler 1 architecture 1 port
       │   ├─[architecture 2] Contains files for the compiler 1 architecture 2 port
       │   └─[architecture 3] Contains files for the compiler 1 architecture 3 port
       │
       └─[compiler 2] Directory containing port files specific to compiler 2
           │
           ├─[architecture 1] Contains files for the compiler 2 architecture 1 port
           ├─[architecture 2] Contains files for the compiler 2 architecture 2 port
           └─[etc.]
```

Include Paths

- FreeRTOS requires three directories to be included in the compiler's include path:
  - The path to the core FreeRTOS header files.

    FreeRTOS/Source/include
  - The path to the source files that are specific to the FreeRTOS port in use.

    FreeRTOS/Source/portable/[compiler]/[architecture]
  - A path to the FreeRTOSConfig.h header file

Header Files

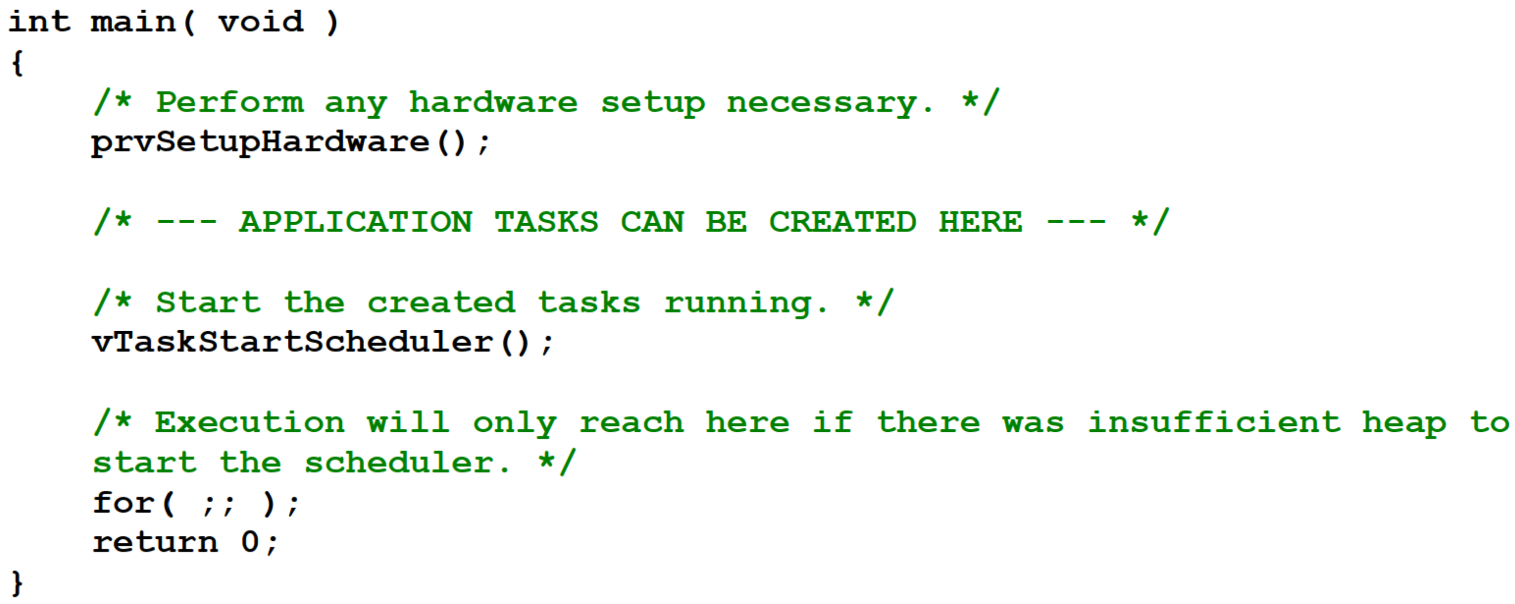
- A source file that uses the FreeRTOS API must include
  - ‘**FreeRTOS.h**’.
  - The header file that contains the prototype for the API function being used – either ‘**task.h**’, ‘**queue.h**’, ‘**semphr.h**’, ‘**timers.h**’ or ‘**event_groups.h**’

Demo Applications

- Each demo project is located in a unique sub-directory under the FreeRTOS/Demo directory.
- Every demo project includes a file called **main.c**, containing the main() function, from where all the demo application tasks are created.

Creating a FreeRTOS Project

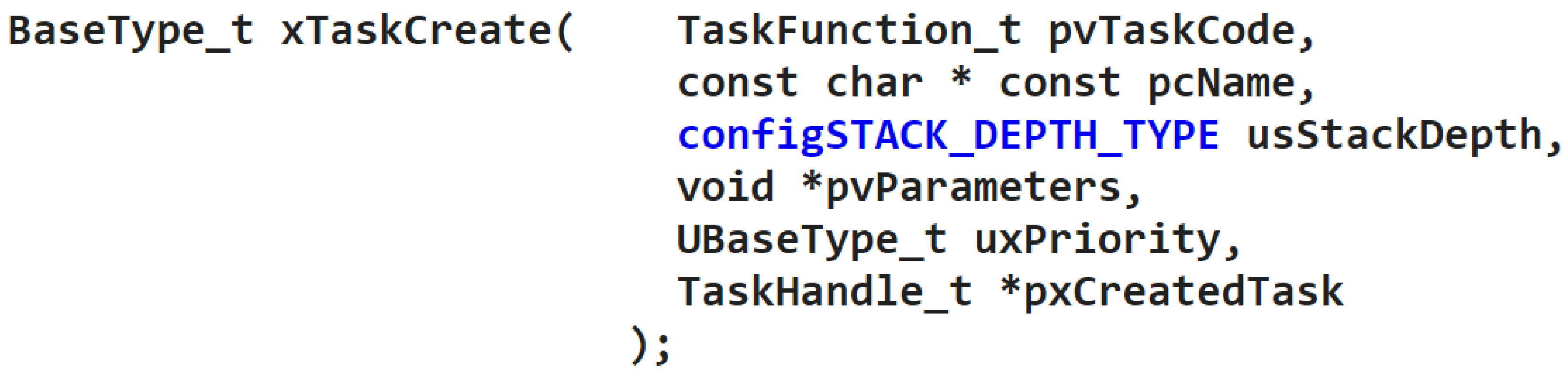
- **Adapting One of the Supplied Demo Projects**



```
int main( void )
{
    /* Perform any hardware setup necessary. */
    prvSetupHardware();

    /* --- APPLICATION TASKS CAN BE CREATED HERE --- */

    /* Start the created tasks running. */
    vTaskStartScheduler();

    /* Execution will only reach here if there was insufficient heap to
    start the scheduler. */
    for( ;; );
    return 0;
}
```

- **Creating a New Project from Scratch**

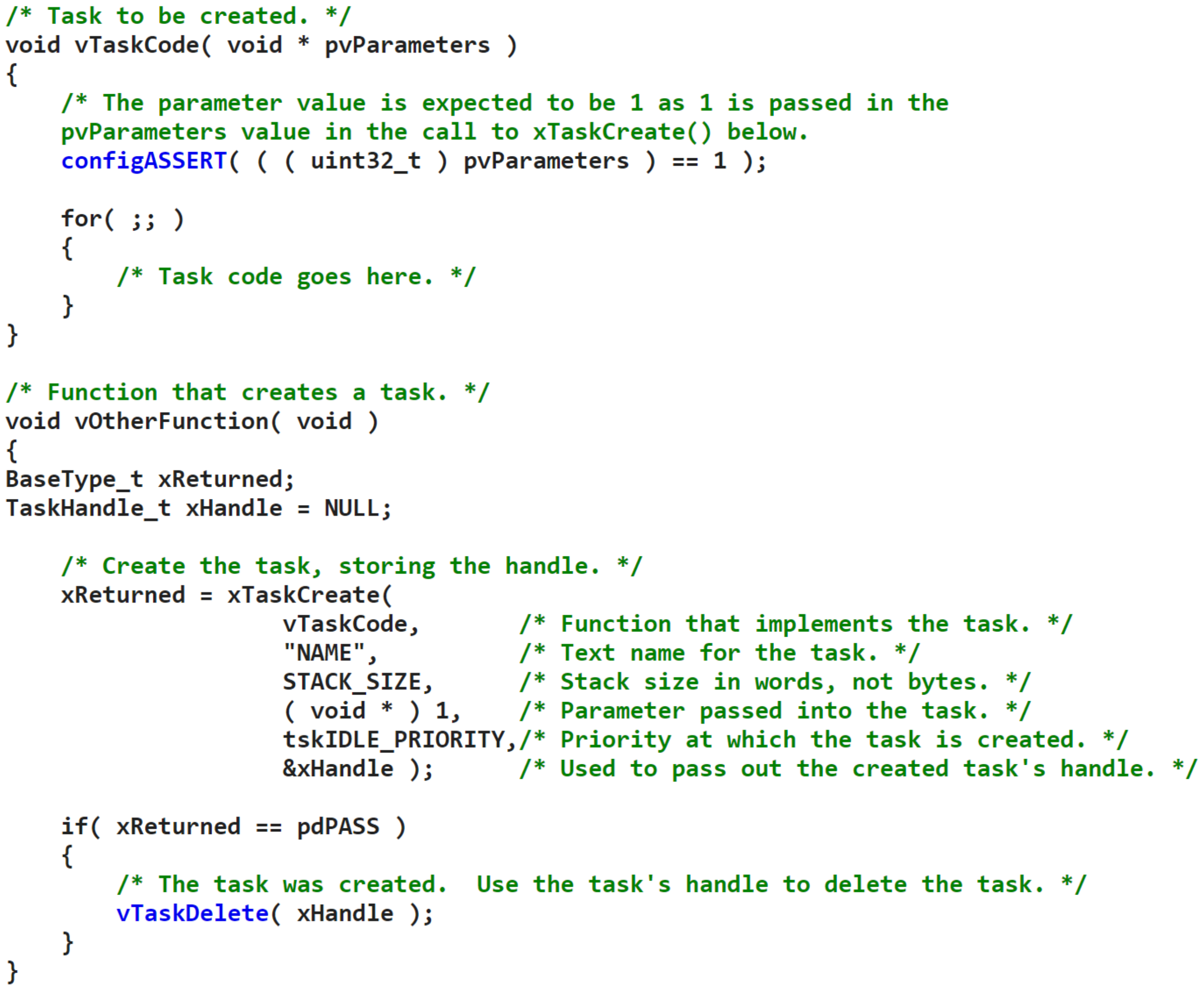
Các API chính của FreeRTOS



```
BaseType_t xTaskCreate(    TaskFunction_t pvTaskCode,
                           const char * const pcName,
                           configSTACK_DEPTH_TYPE usStackDepth,
                           void *pvParameters,
                           UBaseType_t uxPriority,
                           TaskHandle_t *pxCreatedTask
                         );
```

- *pvTaskCode* : task routine
- *pcName* : task name
- *usStackDepth* : stack size (in words)
- *pvParameters* : task param
- *uxPriority* : handle

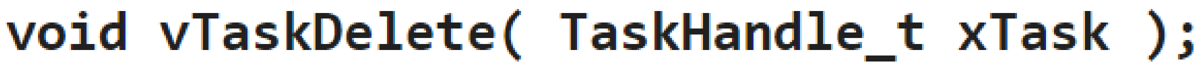
Ví dụ



```
/* Task to be created. */
void vTaskCode( void * pvParameters )
{
    /* The parameter value is expected to be 1 as 1 is passed in the
    pvParameters value in the call to xTaskCreate() below.
    configASSERT( ( ( uint32_t ) pvParameters ) == 1 );

    for( ;; )
    {
        /* Task code goes here. */
    }
}

/* Function that creates a task. */
void vOtherFunction( void )
{
BaseType_t xReturned;
TaskHandle_t xHandle = NULL;

    /* Create the task, storing the handle. */
    xReturned = xTaskCreate(
                    vTaskCode,       /* Function that implements the task. */
                    "NAME",          /* Text name for the task. */
                    STACK_SIZE,      /* Stack size in words, not bytes. */
                    ( void * ) 1,    /* Parameter passed into the task. */
                    tskIDLE_PRIORITY,/* Priority at which the task is created. */
                    &xHandle );      /* Used to pass out the created task's handle. */

    if( xReturned == pdPASS )
    {
        /* The task was created.  Use the task's handle to delete the task. */
        vTaskDelete( xHandle );
    }
}
```

Các API chính của FreeRTOS

- Xóa task



```
void vTaskDelete( TaskHandle_t xTask );
```

- Xóa task khỏi hệ thống, kể cả khi nó đang chạy
- Tài nguyên được thu hồi trong system idle task

Các API chính của FreeRTOS

- **Các hàm điều khiển task**
  - **vTaskDelay**
  - **vTaskDelayUntil**
  - **uxTaskPriorityGet**
  - **vTaskPrioritySet**
  - **vTaskSuspend**
  - **vTaskResume**
  - **xTaskResumeFromISR**
  - **xTaskAbortDelay**

Các API chính của FreeRTOS

- Các hàm hệ thống
  - taskYIELD
  - taskENTER_CRITICAL
  - taskEXIT_CRITICAL
  - taskENTER_CRITICAL_FROM_ISR
  - taskEXIT_CRITICAL_FROM_ISR
  - taskDISABLE_INTERRUPTS
  - taskENABLE_INTERRUPTS
  - vTaskStartScheduler
  - vTaskEndScheduler
  - vTaskSuspendAll
  - xTaskResumeAll
  - vTaskStepTick

FreeRTOS Kernel Services

- Heap Memory Management
- Task Management
- Queue Management
- Software Timer Management
- Interrupt Management
- Resource Management
- Event Groups
- Task Notifications

➔Xem thêm ***https://www.freertos.org***

➔Xem thêm ***FreeRTOS Kernel Quick Start Guide***

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Cảm ơn
đã lắng nghe!

soict.hust.edu.vn/ fb.com/groups/soict