FreeRTOS for STM32

Step 2

Firmware

Firmware

컴퓨팅과 공학 분야에서 특정 하드웨어 장치에 포함된 소프트웨어, 소프트웨어를 읽어 실행하거나, 수정되는 것도 가능한 장치를 뜻한다. 펌웨어는 ROM이나 PROM에 저장되며, 하드웨어보다는F 교환하기가 쉽지만, 소프웨어보다는 어렵다.
인용 : 위키백과 (https://ko.wikipedia.org/wiki/%ED%8E%8C%EC%9B%A8%EC%96%B4)

Firmware : 임베디드 하드웨어를 동작시키며 특정 기능의 수행을 목표로하는 프로그램

Real-Time Operating System

RTOS 란

- 주로 임베디드 시스템에서 사용
- Real-Time은 원하는 시간안에 작업이 완료되서 결과를 반환을 의미
- 스케줄러을 이용한 multi tasking을 지원
- 선점형 스케줄링
- Task 우선준위를 가짐

RTOS 구분

- Hard Real-Time
  - ✓ 특정 작업을 일정 시간안에 반드시 처리해야하는 시스템
  - ✓ 군사장비, 의료장비, 비행기 등
- Soft Real-Time
  - ✓ 특정 작업에 대한 시간 제약이 있지만 그렇지 못하더라도 큰 영향이 없는 시스템
  - ✓ TV, 세탁기, 라우터 등

RTOS Firmware 개발

- 특정 작업이 시간에 대한 일관성이 필요한 경우
- Non-OS Firmware 개발 보다 프로그램 코드의 복잡도 증가
- 같은 작업하에 Non-OS Firmware보다 많은 메모리가 필요
- 사용하는 프로세서 제품 및 생산회사에 대한 종속성은 존재

Real-Time Operating System

RTOS 종류

❖ https://en.wikipedia.org/wiki/Comparison_of_real-time_operating_systems

- vxWorks
  - ✓ 미국 윈드리버에서 개발
  - ✓ 빠른 멀티태스킹
  - ✓ WindSh 쉘 지원
  - ✓ 파일 시스템 입출력 지원
  - ✓ 다양한 프로세서 지원 (ARM, IA-32, x86-64, MIPS, PowerPC 등)
- FreeRTOS
  - ✓ 2003년 Richard Barry가 개발
  - ✓ Open source
  - ✓ 2017년 아마존에 인수
  - ✓ 35개 이상의 마이크로 컨트롤러 지원(ARM, AVR, ColdFire, IA-32, PIC, MSP430 등)
- mbed-OS
  - ✓ ARM에서 개발
  - ✓ Open Source
  - ✓ IoT Device를 위한 RTOS
  - ✓ ARM의 Cortex-M, Cortex-R 만 지원

General-Purpose Operating System

GPOS	RTOS
단일 용도, 다목적 용도	단일 용도, 다목적 용도
Not time critical	Time critical
높은 처리량 우선 스케줄링	우선순위 기반 스케줄링
RTOS에 비해 무겁고 크기가 크다	GPOS에 비해 가볍고 크기가 작다
범용 장치용(PC, 워크스테이션, 서버 등)	독립형 장차용(자판기, 키호스크 등)

General-Purpose Operating System

Linux

- 1991년 리누스 토르발스에 의해서 시작
- Open source
- UNIX기반 커널이 아님
- 1994년 GNU 유틸을 포함하는 커널 버전 1.0 발표
- IBM, Google, Microsoft 등의 기업에서 리국스 개발을 지원
- PC, 워크스테이션, 서버, 모바일 등 다양한 곳에서 사용
- 스팀이 지원됨
- 기술지원 비용은 공짜가 아님
- 다양한 프로세서에 이식(x86, ARM, MIPS 등)
- 다양한 배포판이 존재(fedora, Ubuntu, Gentoo, slackware 등)

임베디드 Linux

- Open source로 라이센스 비용이 없음
- 다양한 프로세서에 이식
- 용도와 크기에 맞게 변경 가능
- 안정된 커널
- Yocto, Linaro, OpenEmbedded, Buildroot, OpenWrt, LTIB 등의 빌드시스템 사용가능

General-Purpose Operating System

Windows

- Microsoft사 개발
- 미려한 GUI
- Windows NT계열 과 Windows Embedded계열의 제품군이 있음

Windows NT계열

- PC, 서버, 워크스테이션등에 사용
- XP, Vista, 7, 8, 10

Windows embedded계열

- Windows 10의 임베디드 버전 부터 "Windows Embedded"를 "Windows IoT"로 변경
- 리소스가 제한적인 장치에 대한 범용 운영체제
- Enterprise, Mobile Enterprise, IoT Mobile, Core, Core Pro

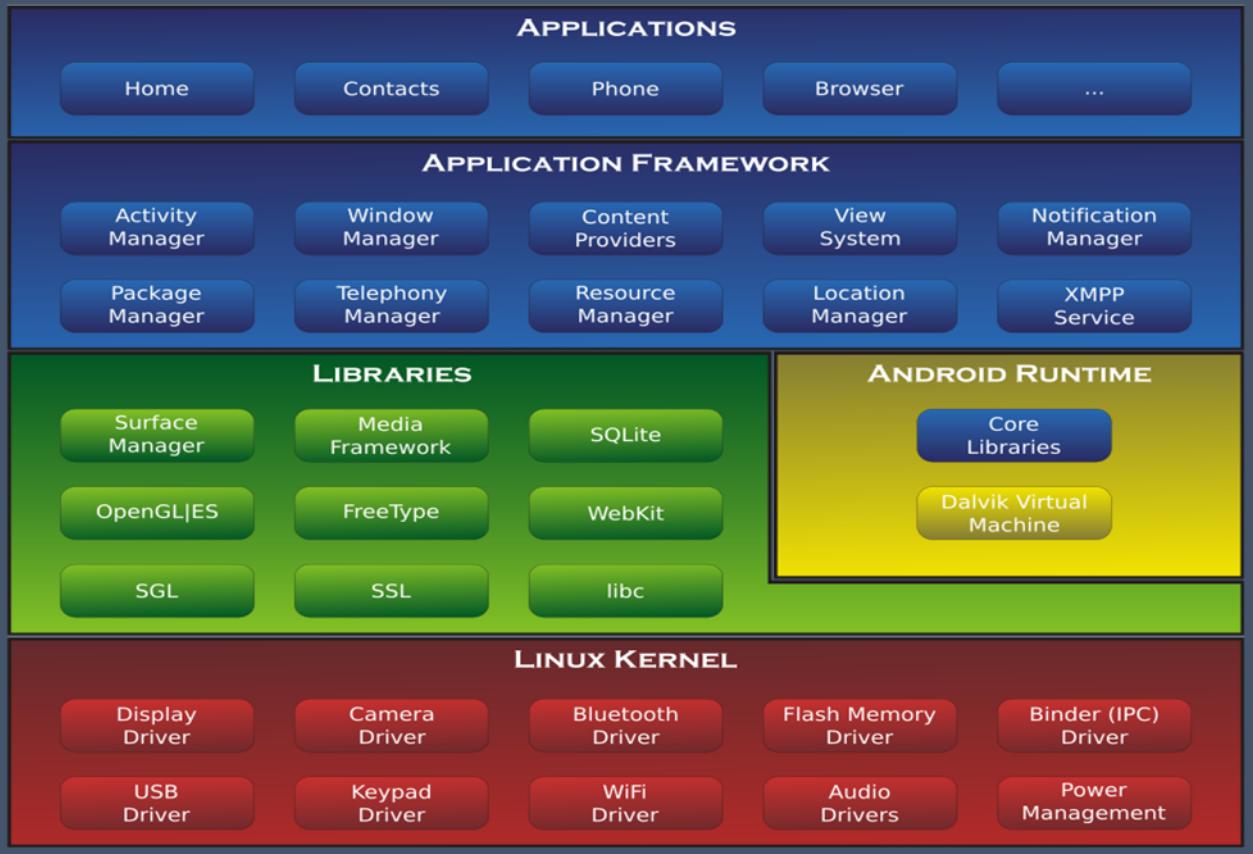
기타 계열

- Windows Phone
  - ✓ 이전 Windows Mobile에서 Windows Phone으로 이전
  - ✓ 모바일 운영체제
  - ✓ Windows Phone 7, Windows Phone 8, Windows Phone 8.1, Windows 10 Mobile
- Xbox OS
  - ✓ Xbox One 운영체제

Mobile Operating System

Android

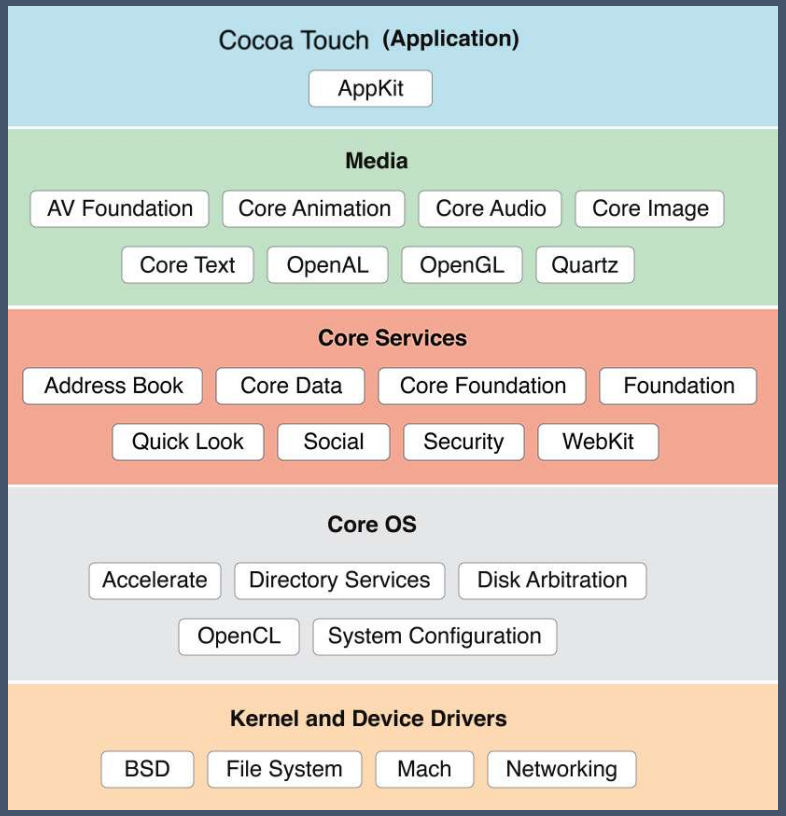
- 2007년 11월 안드로이드 알파로 시작
- 구글, OHA(Open Handset Alliance)가 개발한 모바일 운영체제



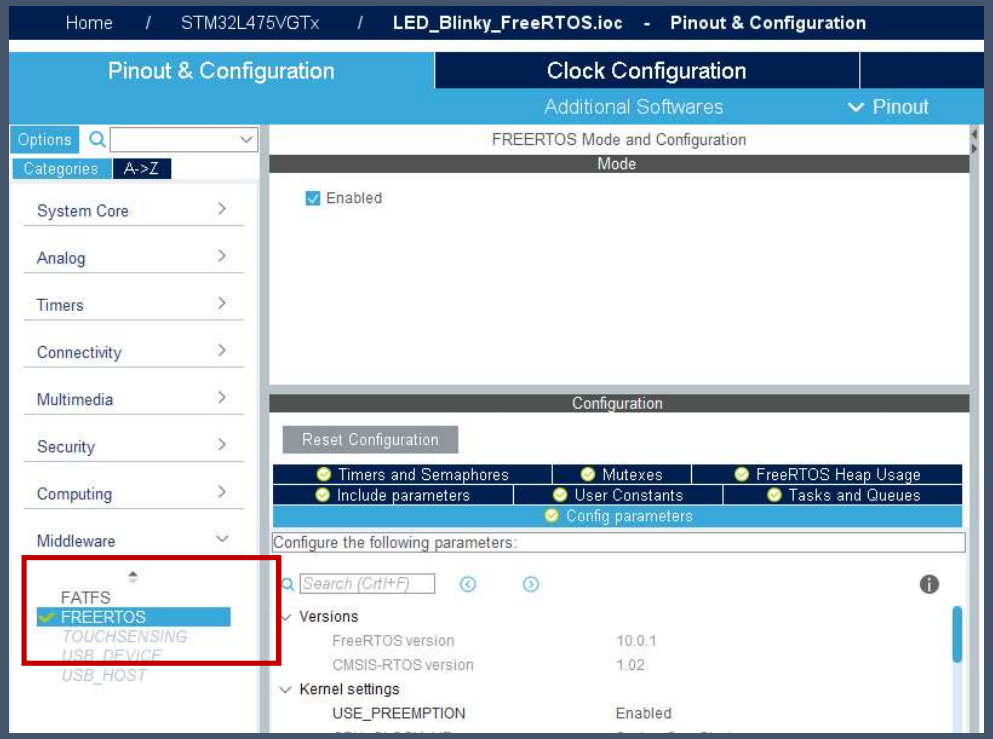
Mobile Operating System

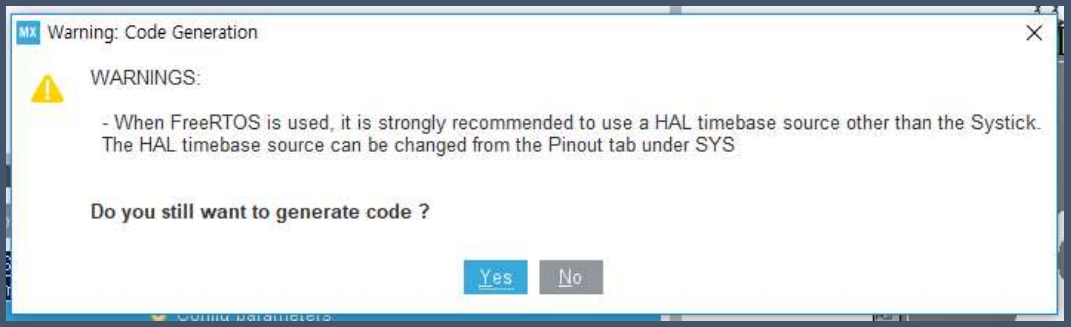
IOS

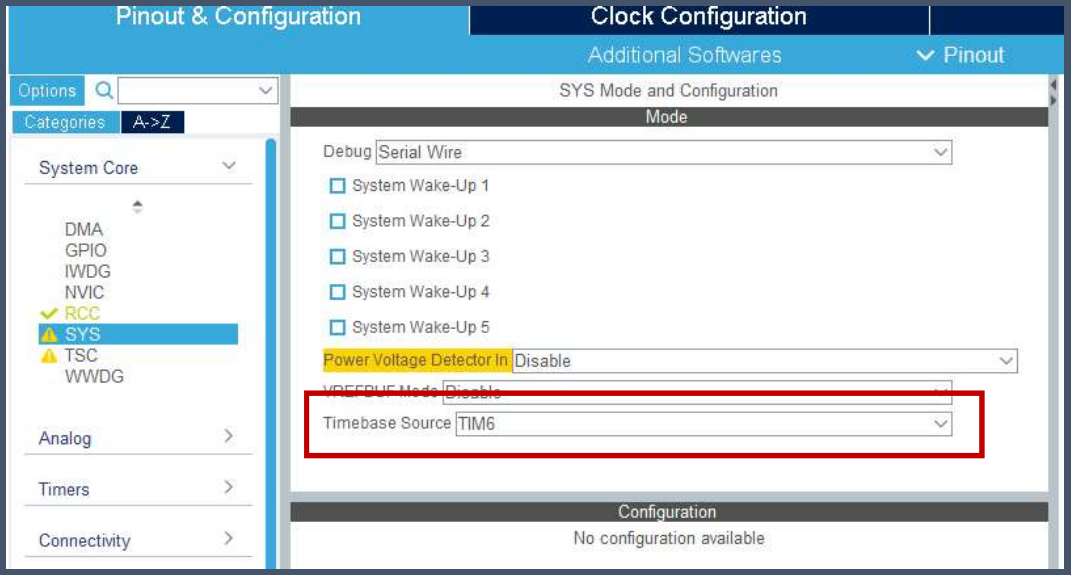
- 2007년 애플이 공개한 모바일 운녕체제
- 2008년 App용 SDK공개
- 애플의 OS X를 기반으로 만들어짐



FreeRTOS Blinky







FreeRTOS

Data Types and Coding Style

변수명 (prefix)
'c' : char
's' : int16_t (short)
'l' : int32_t (long)
'x' : 'BaseType_t'
'u' : unsigned
'p' : pointer

함수명 (prefix)
함수의 return type을 접두어로 사용
예) - vTaskPrioritySet() : void
- xQueueRecevie() : BaseType_t
- pvTimerGetTimerID() : void pointer
- 'prv' : private function

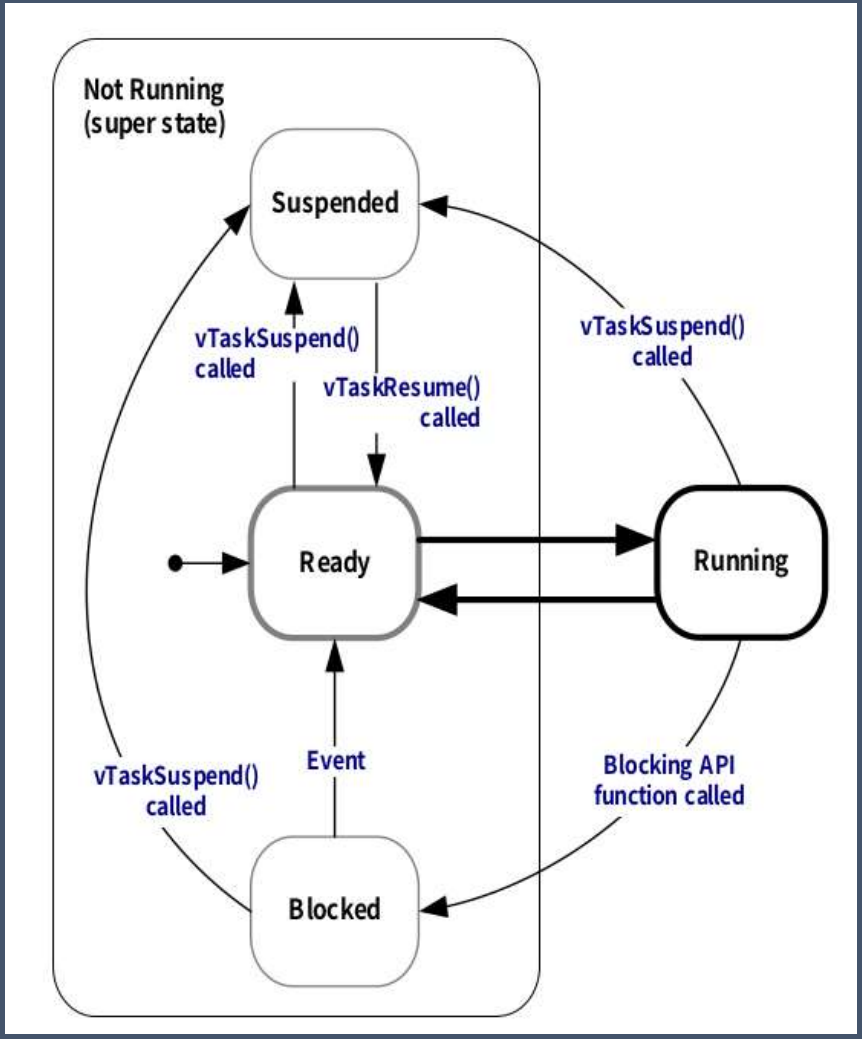
매크로 이름
매크로 이름은 대문자이고 접두어는 소문자이다

"TickType_t" : tick 인터럽트에서 호출하는 Tick 변수 타입
unsigned 16 bit 값을 쓸때는
configUSE_16_BIT_TICKS ("FreeRTOSConfig.h")
아니면 unsigned 32bit를 사용

"BaseType_t" : 일반적으로 매우 제한된 범위의 값만 취할 수 있는
return 타입과 pdTRUE/pdFALSE 타입의 Boolean에 사용
일반적으로 cpu architecture의 bit수를 따른다.
(예. 32bit cpu-> 32bit type)

FreeRTOS - Task Management

FreeRTOS의 어플리케이션은 Task로 이루어진다.
하나의 프로세서에느는 하나의 task만 주어진 시간만큼 실행된다.



FreeRTOS - Task Management

Task 생성

Task 함수

```
void ATaskFunction( void *pvParameters );
```

Task 생성 함수

BaseType_t xTaskCreate(TaskFunction_t pvTaskCode, const char * const pcName, uint16_t usStackDepth, void *pvParameters, UBaseType_t uxPriority, TaskHandle_t *pxCreatedTask);	
pvTaskCode	Task 함수
pcName	task 이름 configMAX_TASK_NAME_LEN ("FreeRTOSConfig.h") 마지막 NULL 포함 이름의 길이를 정의
sStackDepth	task의 stack크기(Word 길이) 예) 32bit system에서 sStackDepth이 100이면 400byte가 할당됨 configMINIMAL_STACK_SIZE("FreeRTOSConfig.h") Idle task에서 사용되는 stack의 크기를 정의
pvParameters	task함수에 전달되는 파라메터
uxPriority	task의 실행 우선순위 가장 낮은 우선순위 0 ~ (configMAX_PRIORITIES - 1)까지 할당 configMAX_PRIORITIES("FreeRTOSConfig.h")
pxCreatedTask	task handle task handle을 사용하지 않는다면 NULL전달
return value	pdPASS: 성공 pdFAIL: 실패 (task를 생성할 heap memory가 부족)

FreeRTOS - Task Management

Example01 – Task 생성

FreeRTOS - Task Management

Example02 – Task parameter

FreeRTOS - Task Management

Task Priority

- Task 생성시 : uxPriority
- 우선순위 최대값 : configMAX_PRIORITIES("FreeRTOSConfig.h") 설정

LOW ---------- HIGH
0 ~ (configMAX_PRIORITIES - 1)

tick interrupt

스케줄러의 시간분배는 'tick interrupt'를 사용
configTICK_RATE_HZ("FreeRTOSConfig.h") tick 주파수 설정

pdMS_TO_TICKS() macro로 설정한 ms만큼의 tick값으로 변환
예) TickType_t xTimeInTicks = pdMS_TO_TICKS(200); // 200ms

FreeRTOS - Task Management

Example03 – 우선순위

FreeRTOS - Task Management

‘Not Running’ State

Blocked state

task가 event를 기다리는 상태
Blocked state에서 기다리는 두가지 종류의 event

- 시간적 이벤트: 지연 기간 만료 또는 절대 시간 도달 중 하나, 예) 10ms가 지나가길 기다림
- 동기화 이벤트: 다른 task나 interrupt로 부터 발생하는 event (예) queue, semaphore 등)

Suspended State

Suspended State는 스케줄러를 사용할 수 없다.
vTaskSuspend() 호출로 suspended,
vTaskResume() 또는 xTaskResumeFromISR()로 Ready state

Ready State

실행(Running State) 준비가 된 상태

FreeRTOS - Task Management

Task Delay

vTaskDelay( )
INCLUDE_vTaskDelay("FreeRTOSConfig.h") 설정 1

void vTaskDelay(TickType_t xTicksToDelay);	
xTicksToDelay	tick interrupt의 수 예) vTaskDelay(pdMS_TO_TICKS(100)); // 100 ms

FreeRTOS - Task Management

Example04 – Task Delay

FreeRTOS - Task Management

Task Delay

vTaskDelay( )
Task가 Blocked state로 유지되는 시간은 vTaskDelay () 매개 변수에 의해 지정되지만 Task가 Blocked state를 벗어나는 시간은 vTaskDelay ()가 호출 된 시간을 기준으로한다.

vTaskDelayUntil( )
매개 변수는 호출하는 Task를 Blocked state에서 Ready state로 이동해야하는 정확한 tick 값을 대신 지정
정확인 주기의 실행이 필요한 경우 사용

void vTaskDelayUntil(TickType_t * pxPreviousWakeTime, TickType_t xTimeIncrement);	
pxPreviousWakeTime	pxPreviousWakeTime은 task가 마지막으로 Blocked State 를 벗어난 시간이고, 이 시간은 task가 다음에 Blocked State를 벗어나는 시간을 계산하기위한 참조 점으로 사용 pxPreviousWakeTime는 함수내에서 자동으로 업데이트 일반적으로 사용되기 전에 현재 tick값으로 초기화
xTimeIncrement	wait되는 tick값 예) pdMS_TO_TICKS(200) // 200 ms의 tick의 값

FreeRTOS - Task Management

Example05 – vTaskDelayUntil( )

FreeRTOS - Task Management

Example06 – blocking과 non-blocking

FreeRTOS - Task Management

Idle Task

Running state에는 항상하나의 task가 실행되어야한다.
FreeRTOS에서는 이러한 용도로 Idle task를 사용하고 vTaskStartScheduler()가 호출될때 자동으로 생성
configIDLE_SHOULD_YIELD("FreeRTOSConfig.h") Idle Priority 설정

Idle Task Hook 함수

application에서는 특정함수를 idle hook(idle callback)함수 설정이 가능
Hook함수는 idle task loop마다 idle task에서 한번 호출
configUSE_IDLE_HOOK("FreeRTOSConfig.h") 설정 1

Idle task hook 함수

```
void vApplicationIdleHook( void );
```

FreeRTOS - Task Management

Example07 – Idle Task Hook

FreeRTOS - Task Management

Task 우선순위 변경

vTaskPrioritySet()
스케줄러가 시작한 후에 task의 우선 순위를 변경
INCLUDE_vTaskPrioritySet ("FreeRTOSConfig.h") 설정 1

void vTaskPrioritySet(TaskHandle_t pxTask, UBaseType_t uxNewPriority);	
pxTask	우선순위를 변경할 task의 handle NULL은 자신의 우선순위를 변경
uxNewPriority	변경할 우선순위 값

uxTaskPriorityGet()
task의 우선순위를 쿼리
INCLUDE_uxTaskPriorityGet ("FreeRTOSConfig.h") 설정 1

UBaseType_t uxTaskPriorityGet(TaskHandle_t pxTask);	
pxTask	우선순위를 쿼리할 task의 handle NULL은 자신의 우선순위를 쿼리
return value	쿼리한 task에대한 현재 우선순위 값

FreeRTOS - Task Management

Example08 – Task 우선순위 변경

FreeRTOS - Task Management

Task 삭제

vTaskDelete()
INCLUDE_vTaskDelete ("FreeRTOSConfig.h") 설정 1

void vTaskDelete(TaskHandle_t pxTaskToDelete);	
pxTaskToDelete	삭제할 task의 handle NULL은 자신의 task를 삭제

FreeRTOS - Task Management

Example09 – Task 삭제

FreeRTOS - Task Management

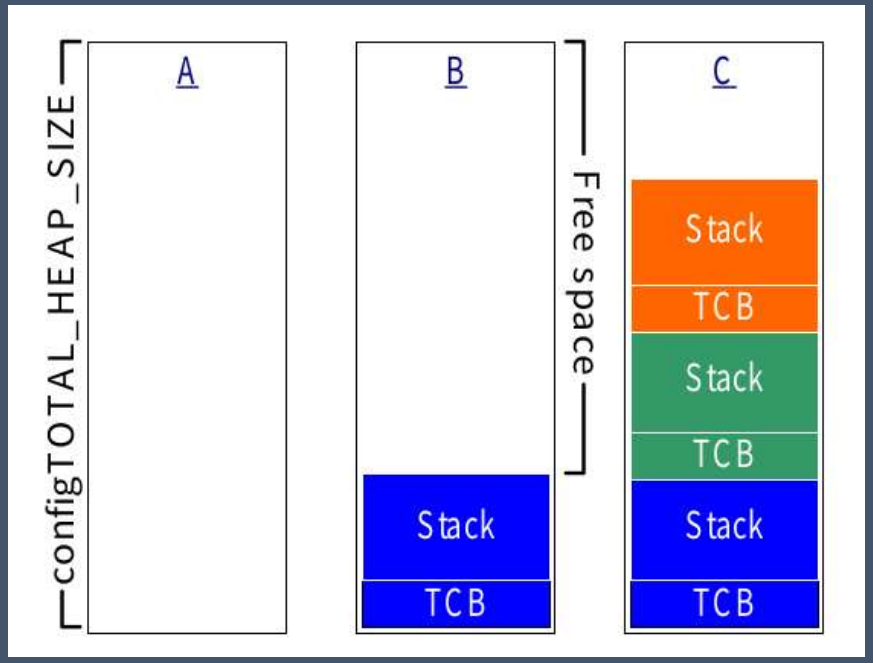
Heap Memory Management

FreeRTOS에서 RAM을 할당할때는 "malloc() -> pvPortMalloc()", "free() -> vPortFree()" 를 사용
FreeRTOS에서는 pvPortMalloc(), vPortFree()에 대한 다섯가지 방식을 예제로 제공
(heap_1.c , heap_2.c, heap_3.c, heap_4.c, heap_5.c)

FreeRTOS - Task Management

Heap Memory Management

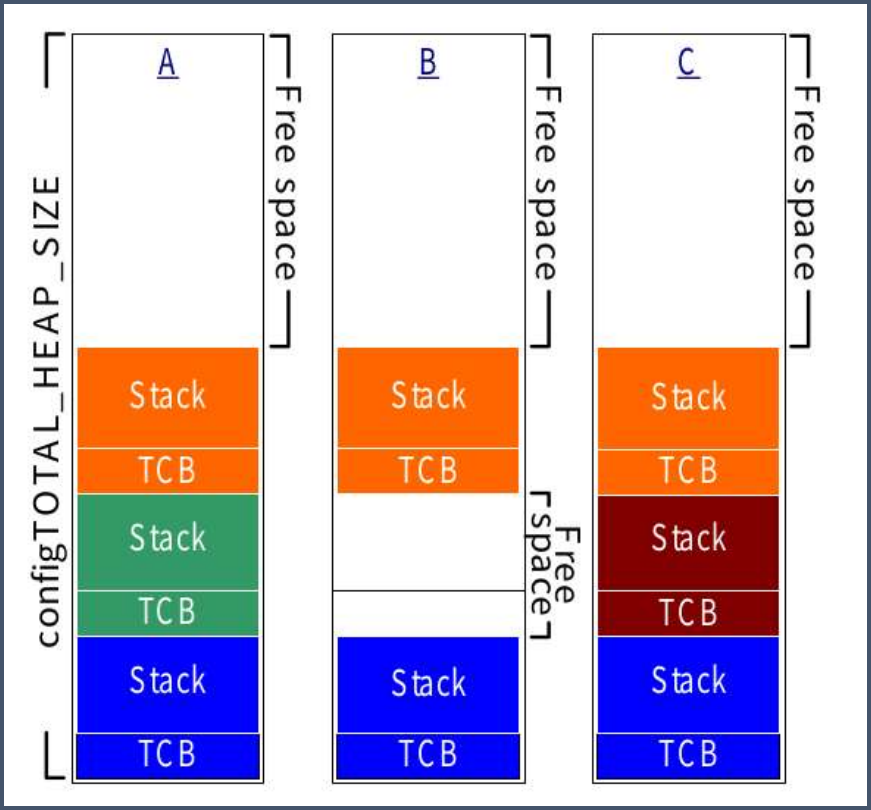
Heap_1
스케줄러가 시작되기 전에 태스크와 다른 커널 오브젝트만 생성
이 방법은 메모리 설정 및 파편화 같은 복잡성 문제보다는 코드크기와 단순성 같은 특성을 고려한 것
가장 기본적인 pvPortMalloc()를 제공, vPortFree()는 구현되어 있지 않다.
configTOTAL_HEAP_SIZE ("FreeRTOSConfig.h") 전체 힙사이즈 정의



FreeRTOS - Task Management

Heap Memory Management

Heap_2 (best fit algorithm)
이전 버전에 대한 호환성을 위해서 존재, 사용하지 않는 것을 권고
heap_2 대신 heap_4를 사용을 권장 함



FreeRTOS - Task Management

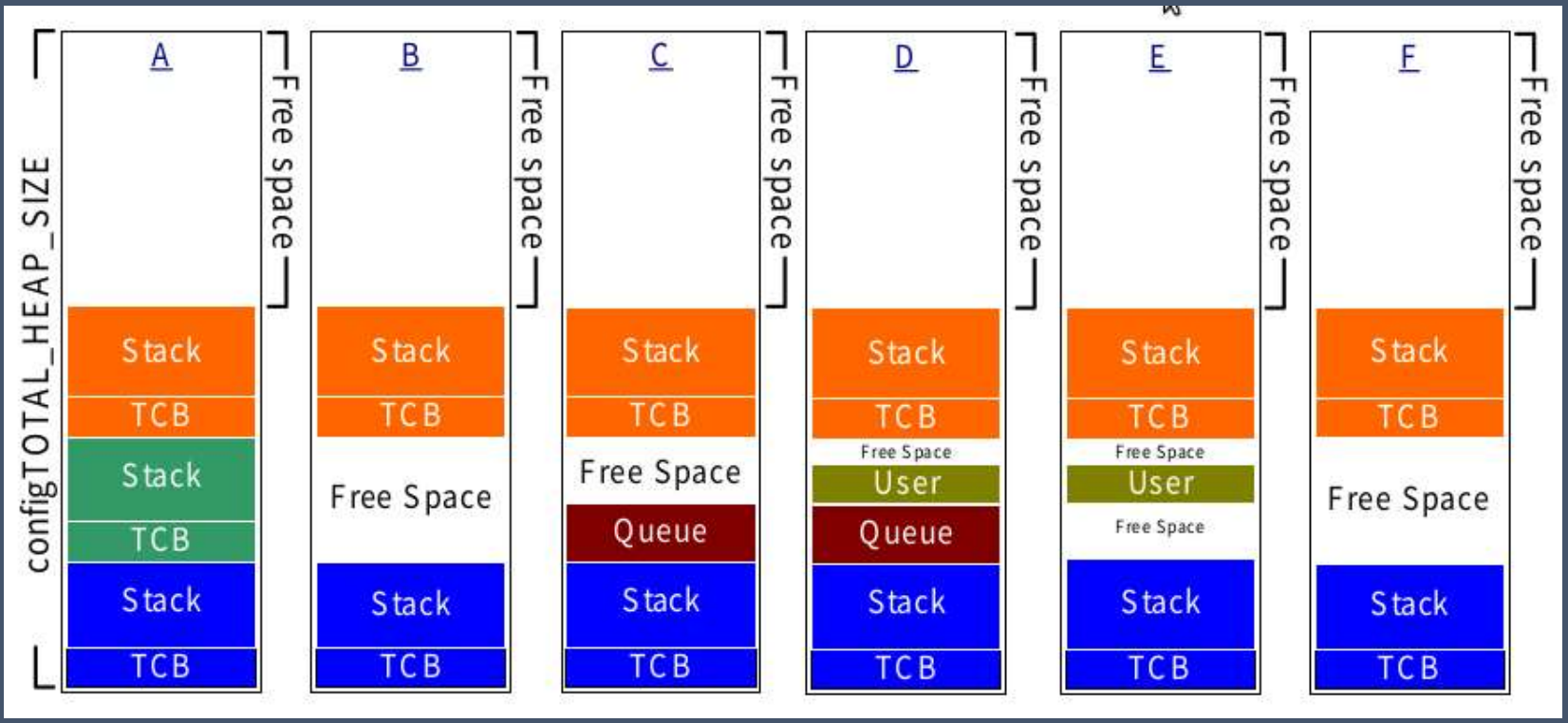
Heap Memory Management

Heap_3
C 표준 라이브러리인 malloc(), free()를 사용, 따라서 heap 크기는 링커에 의해서 결정
configTOTAL_HEAP_SIZE ("FreeRTOSConfig.h") 는 아무련 영향이 없다.

FreeRTOS - Task Management

Heap Memory Management

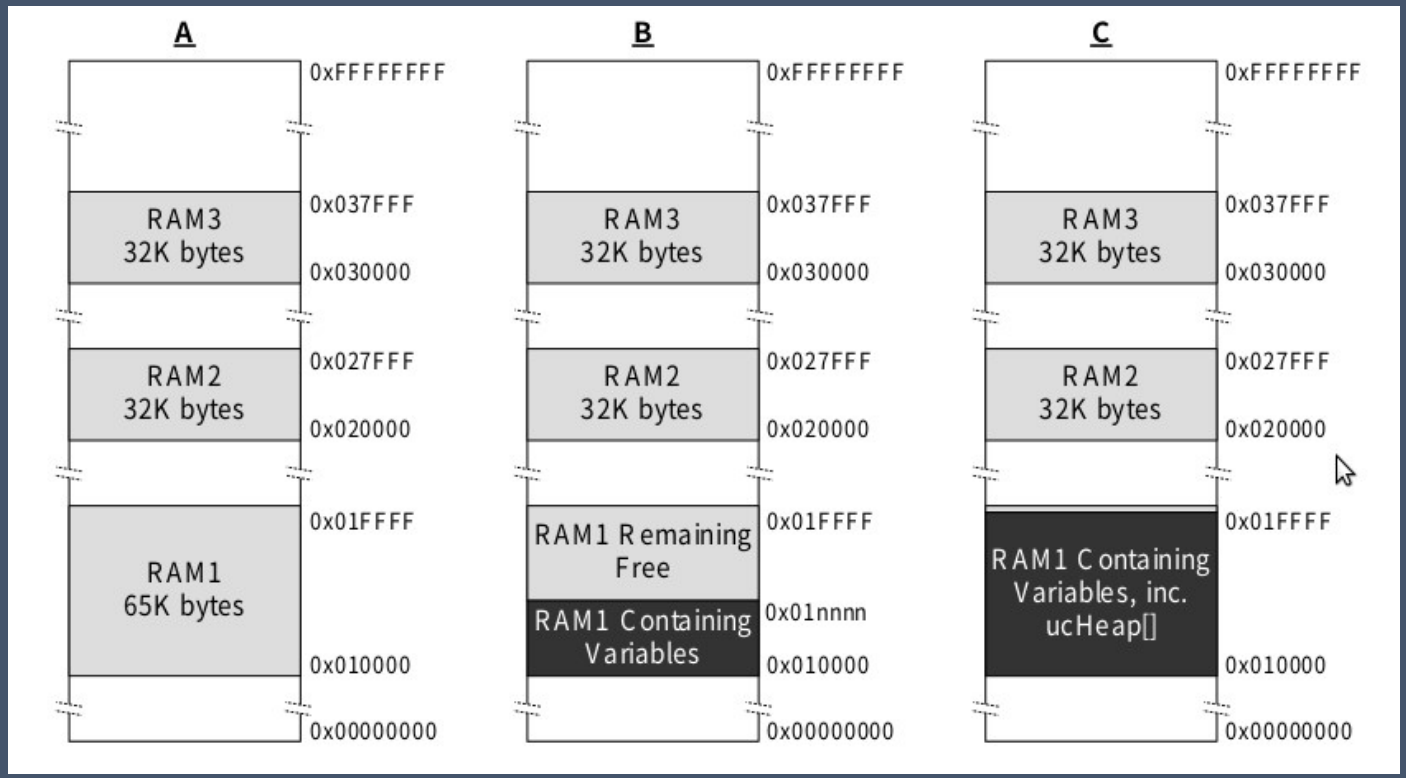
Heap_4 (first fit algorithm)
heap_1, heap_2와 같이 heap_4도 힙을 작은 블럭으로 나누어 사용
힙의 크기는 configTOTAL_HEAP_SIZE (" FreeRTOSConfig.h ")로 정의
heap_2와는 다르게 heap_4는 하나의 큰 블럭안 메모리의 인접한 free 블럭을 합친다.



FreeRTOS - Task Management

Heap Memory Management

Heap_5
다중의 분리된 메모리 영역으로 부터 메모리를 할당
시스템의 메모리 맵이 하나의 연속적인 블럭을 제공하는 시스템에서 유용하
메모리 할당 및 커널 오브젝트를 생성하기 전에 "vPortDefineHeapRegisions()"를 이용해서 초기화



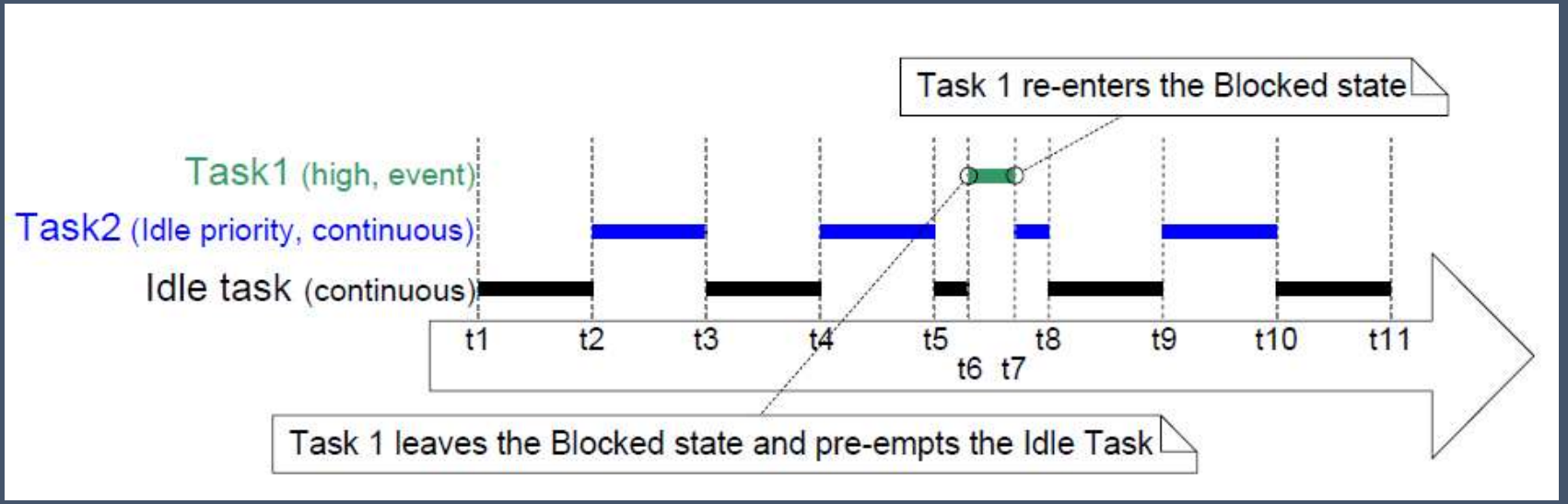
FreeRTOS - Task Management

Scheduling Algorithms

스케줄링 알고리즘
Running state로 전환할 Ready state task를 결정하는 소프트웨어 루틴

Fixed Priority Pre-emptive Scheduling with Time Slicing
대부분의 작은 RTOS에서 사용되는 방식

configUSE_PREEMPTION	1
configUSE_TIME_SLICING	1

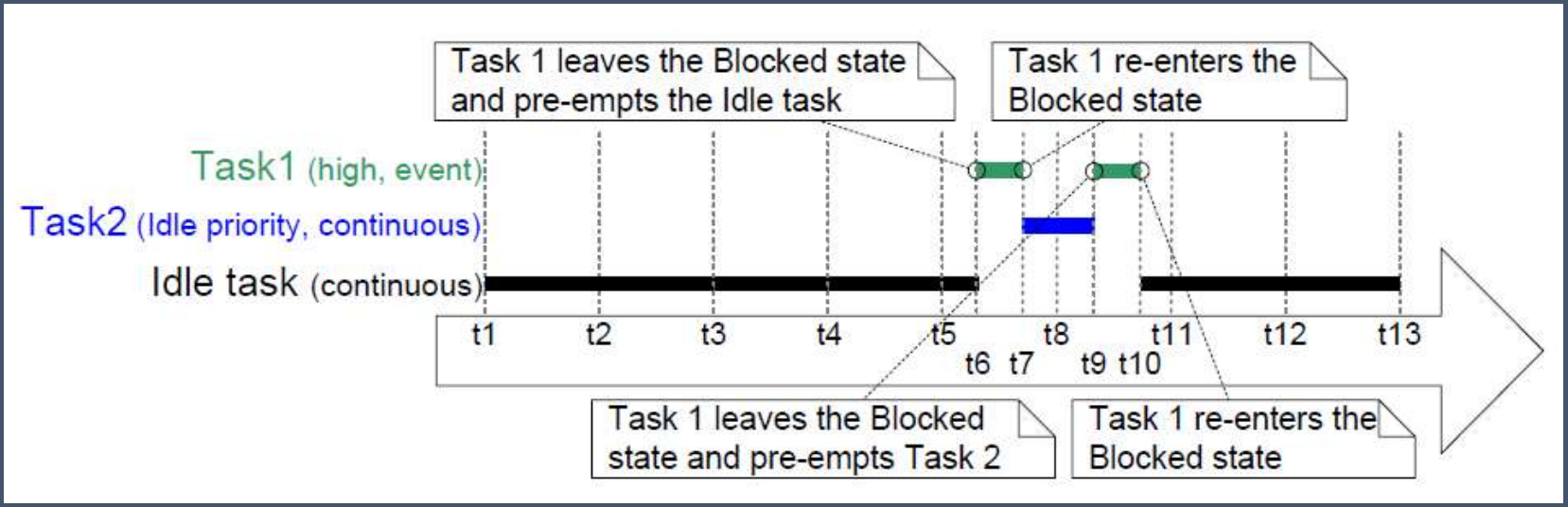


FreeRTOS - Task Management

Scheduling Algorithms

우선순위 선점형 스케줄링(time slicing 사용하지 않음)
task 선택과 선점 알고리즘을 time slicing없이 관리
time slicing 사용하지 않기 때문에 같은 우선순위의 task 사이에 processing time을 공유하지 않는다.
scheduler는 Running state에 들어갈 새로운 task선택
- Ready state에 높은 우선순위 task가 들어올때
- Running state안 task가 Blocked 또는 Suspended state로 들어갈때

configUSE_PREEMPTION	1
configUSE_TIME_SLICING	0

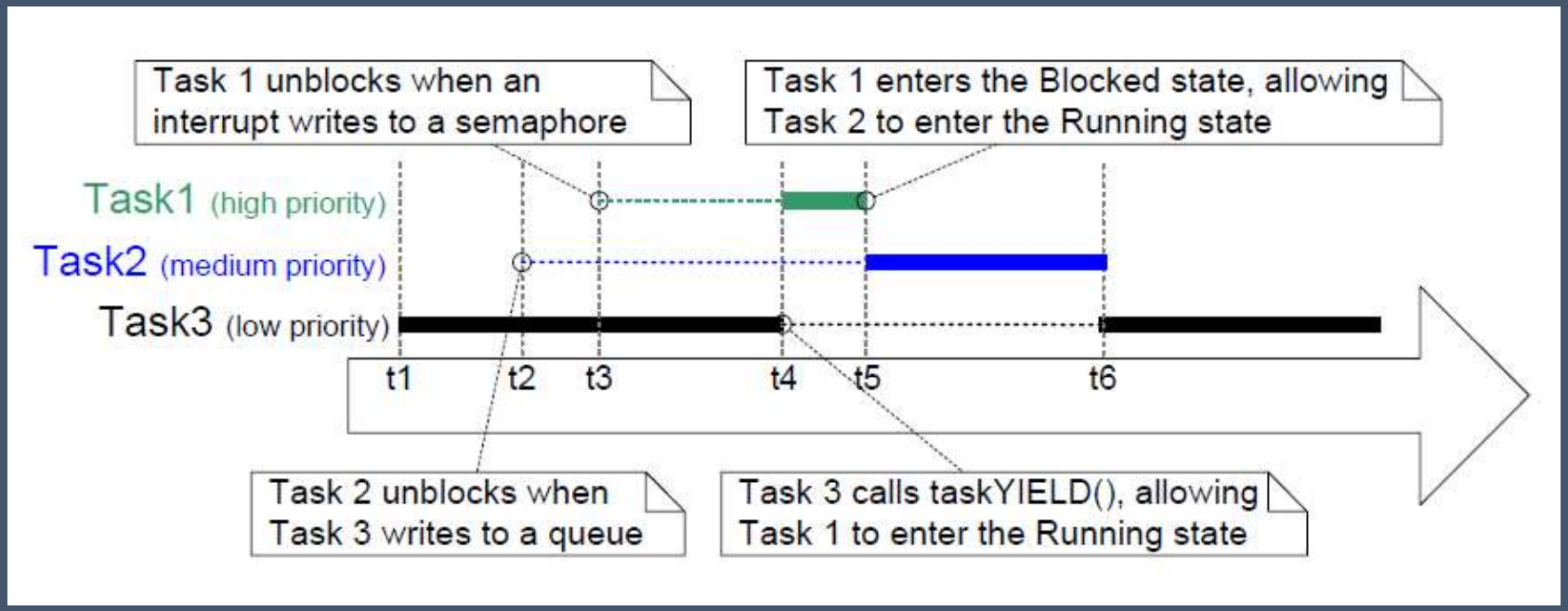


FreeRTOS - Task Management

Scheduling Algorithms

Co-operative Scheduling(협동적 스케줄링)
context switch는 Running state task가 Blocked state되거나 명시적으로 yield(taskYIELD()) 하는 경우 context switch가 된다.

configUSE_PREEMPTION	0
configUSE_TIME_SLICING	Any value



감사합니다.