

# STM32를 활용한 FreeRTOS입문

# Firmware

## Firmware

컴퓨팅과 공학 분야에서 특정 하드웨어 장치에 포함된 소프트웨어, 소프트웨어를 읽어 실행하거나, 수정되는 것도 가능한 장치를 뜻한다. 펌웨어는 **ROM**이나 **PROM**에 저장되며, 하드웨어보다는 **F** 교환하기가 쉽지만, 소프트웨어보다는 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-Time**은 원하는 시간안에 작업이 완료되서 결과를 반환을 의미
- 스케줄러를 이용한 **multi tasking**을 지원
- 선점형 스케줄링
- **Task** 우선순위를 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 Firmware**보다 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](https://en.wikipedia.org/wiki/Comparison_of_real-time_operating_systems)

- vxWorks
  - 미국 윈드리버에서 개발
  - 빠른 멀티태스킹
  - WindSh 셸 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

## 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 리눅스 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 라이선스 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

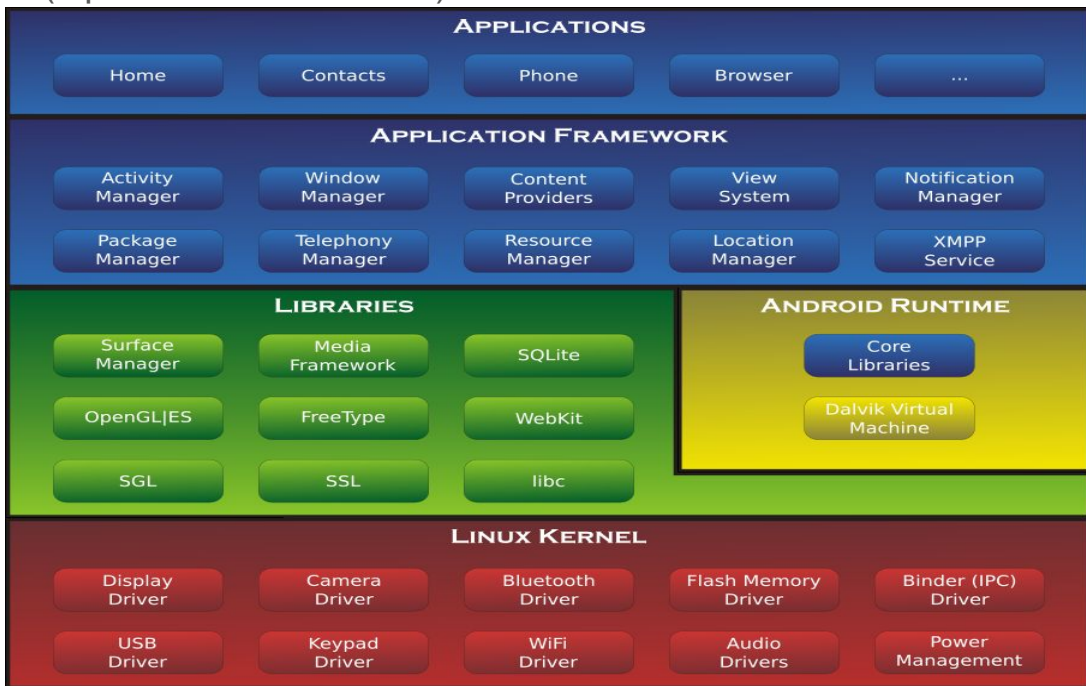
# Operating System 비교

GPOS	RTOS
다목적 용도	단일 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 장치용(자판기, 키호스크 등)

# Mobile Operating System

## Android

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

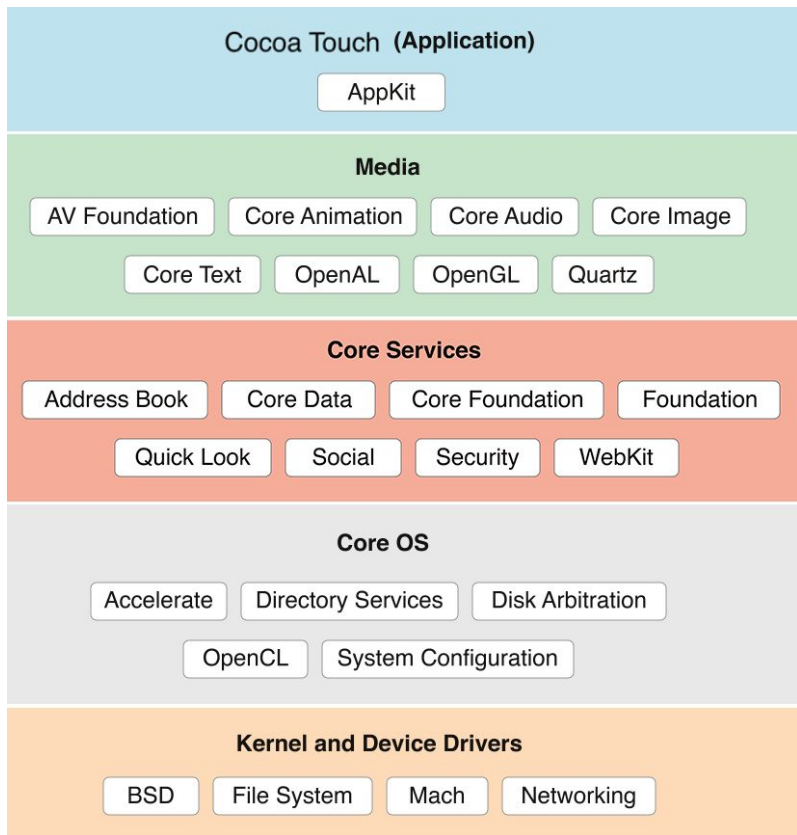




# Mobile Operating System

## IOS

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



# FreeRTOS 추가

The screenshot shows the STM32CubeIDE Pinout & Configuration window. On the left, a sidebar lists various categories: System Core, Analog, Timers, Connectivity, Multimedia, Security, Computing, and Middleware. Under the Middleware category, several options are listed: FATFS, **FreeRTOS** (which is checked and highlighted in blue), TOUCHSENSING, USB\_DEVICE, and USB\_HOST. The main area of the window is titled 'FREERTOS Mode and Configuration' and shows the 'Interface' set to 'CMSIS\_V1'. A warning dialog box is overlaid on the main window. The dialog has a title bar that says 'Warning: Code Generation' and a yellow warning icon. The text inside the dialog reads: 'WARNINGS: - When FreeRTOS is used, it is strongly recommended to use a HAL timebase source other than the SysTick. The HAL timebase source can be changed from the Pinout tab under SYS'. Below this text, it asks 'Do you still want to generate code ?' and provides two buttons: 'Yes' and 'No'.

Pinout & Configuration

Search

Categories A-Z

- System Core
- Analog
- Timers
- Connectivity
- Multimedia
- Security
- Computing
- Middleware
  - FATFS
  - FreeRTOS**
  - TOUCHSENSING
  - USB\_DEVICE
  - USB\_HOST

FREERTOS Mode and Configuration

Mode

Interface CMSIS\_V1

MX Warning: Code Generation

WARNINGS:

- When FreeRTOS is used, it is strongly recommended to use a HAL timebase source other than the SysTick. The HAL timebase source can be changed from the Pinout tab under SYS

Do you still want to generate code ?

Yes No

Reset Configuration

# FreeRTOS 추가

The image shows the STM32CubeIDE interface with three main panels:

- Left Panel (System Core):** A list of system components. 'SYS' is selected and highlighted in blue. Other components include DMA, GPIO, IWDG, NVIC, RCC, TSC, and WWDG.
- Center Panel (Configuration):** Shows various system settings. Under 'Debug', 'Serial Wire' is selected. 'System Wake-Up' options 1 through 5 are unchecked. 'Power Voltage Detector In' is set to 'Disable'. 'VREFBUF Mode' is set to 'Disable'. 'Timebase Source' is set to 'SysTick'. A dropdown menu for 'Timebase Source' is open, showing options: SysTick, TIM1, TIM2, TIM3, TIM4, TIM5, TIM6 (highlighted), and TIM7.
- Right Panel (Project Explorer):** Displays the project structure for 'LED\_test'. The 'Middlewares' folder is expanded, and the 'Third\_Party' sub-folder is highlighted with a red rectangle. Inside 'Third\_Party', the 'FreeRTOS' folder is visible. Other project files include 'LED\_test.ioc', 'LED\_test.launch', 'STM32L475VGTX\_FLASH.Id', and 'STM32L475VGTX\_RAM.Id'.

# Peripheral 파일 분리

	Pinout & Configuration	Clock Configuration	Project Manager
Project	<p>STM32Cube MCU packages and embedded software packs</p> <ul style="list-style-type: none"><li><input type="radio"/> Copy all used libraries into the project folder</li><li><input checked="" type="radio"/> Copy only the necessary library files</li><li><input type="radio"/> Add necessary library files as reference in the toolchain project configuration file</li></ul>		
Code Generator	<p>Generated files</p> <ul style="list-style-type: none"><li><input checked="" type="checkbox"/> Generate peripheral initialization as a pair of '.c/.h' files per peripheral</li><li><input type="checkbox"/> Backup previously generated files when re-generating</li><li><input checked="" type="checkbox"/> Keep User Code when re-generating</li><li><input checked="" type="checkbox"/> Delete previously generated files when not re-generated</li></ul>		
Advanced Settings	<p>HAL Settings</p> <ul style="list-style-type: none"><li><input type="checkbox"/> Set all free pins as analog (to optimize the power consumption)</li><li><input type="checkbox"/> Enable Full Assert</li></ul>		
	<p>Template Settings</p> <p>Select a template to generate customized code</p> <p>Settings...</p>		

# 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 stack 크기 (Word 길이)

예) 32bit system에서 sStackDepth이 100이면 400byte가 할당됨

configMINIMAL\_STACK\_SIZE("FreeRTOSConfig.h") Idle task에서 사용되는 stack의 크기를 정의

pvParameters

task 함수에 전달되는 파라미터

uxPriority

task의 실행 우선순위

가장 낮은 우선순위 0 ~ (configMAX\_PRIORITIES - 1)까지 할당

configMAX\_PRIORITIES("FreeRTOSConfig.h")

pxCreatedTask

task handle

task handle을 사용하지 않는다면 NULL 전달

return value

pdPASS: 성공

pdFAIL: 실패 (task를 생성할 heap memory가 부족)

# FreeRTOS Blinky

Example00 – Blinky

# FreeRTOS Blinky

Example01 – Blinky

# UART

## UART (Universal Asynchronous serial Receiver and Transmitter)

1 대 1통신

비동기 통신 - 동기를 위한 클럭신호를 사용하지 않음

- Baud Rate
- 데이터 전송 속도로 Bit-per-Second(bps)단위로 표시

### 데이터 구조

- **start bit**: 통신의 시작을 의미하며 한 비트 시간 길이 만큼 유지한다. 지금 부터 정해진 약속에 따라 통신을 시작한다.
- **data bit**: 5~8비트의 데이터 전송을 한다. 몇 비트를 사용할 것인지는 해당 레지스터 설정에 따라 결정된다.
- **Parity bit**: 오류 검증을 하기 위한 패리티 값을 생성하여 송신하고 수신쪽에 오류 판단한다. 사용안함, 짝수, 홀수 패리티 등의 세가지 옵션으로 해당 레지스터 설정에 따라 선택할 수 있다. '사용안함'을 선택하면 이 비트가 제거된다.
- **Stop bit**: 통신 종료를 알린다. 세가지의 정해진 비트 만큼 유지해야 한다. 1, 1.5, 2비트로 해당 레지스터 설정에 따라 결정된다.

비트 수	1	2	3	4	5	6	7	8	9	10	11
	시작 비트 (Start bit)	5-8 데이터 비트								패리티 비트 (parity bit)	종료 비트 (Stop bit(s))
	Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Parity	Stop



# UART

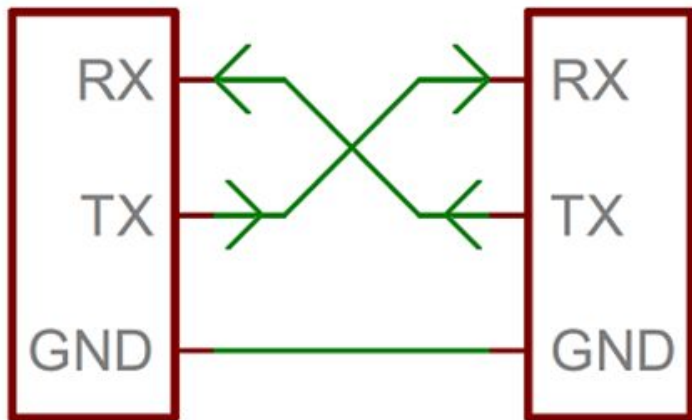
통신 속도 예)

9600 8N1 : 9600 baud rates, 8 data bits, no parity, 1 stop bit

9600 bps 속도로 보내므로 각 비트는  $1/(9600 \text{ bps}) = 104\mu\text{s}$ 이고

8bit 전송시 start bit, stop bit를 더해 10bit 패킷을 사용하므로 초당 960byte를 전송할 수 있다.

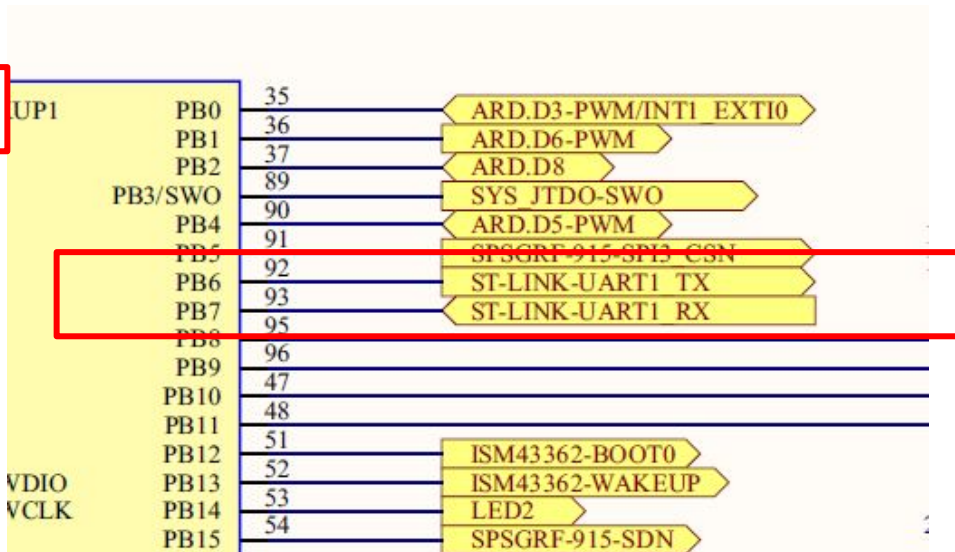
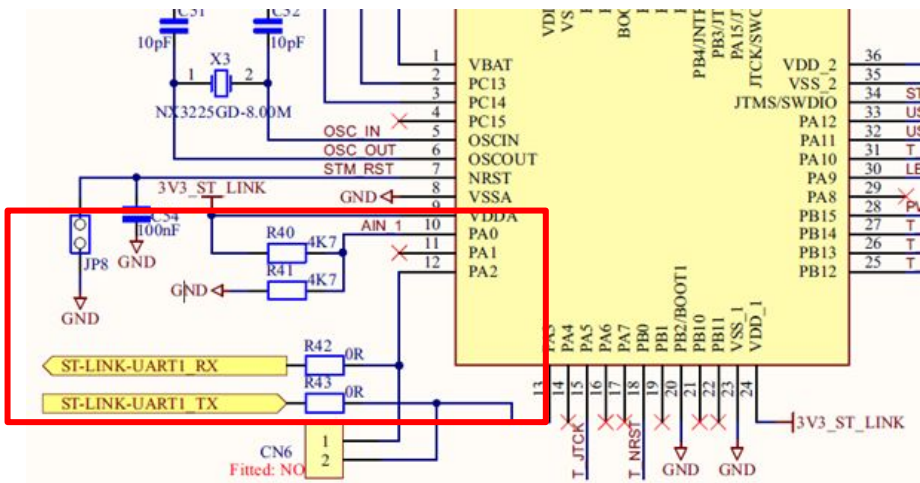
## UART 회선 연결 방법



# UART

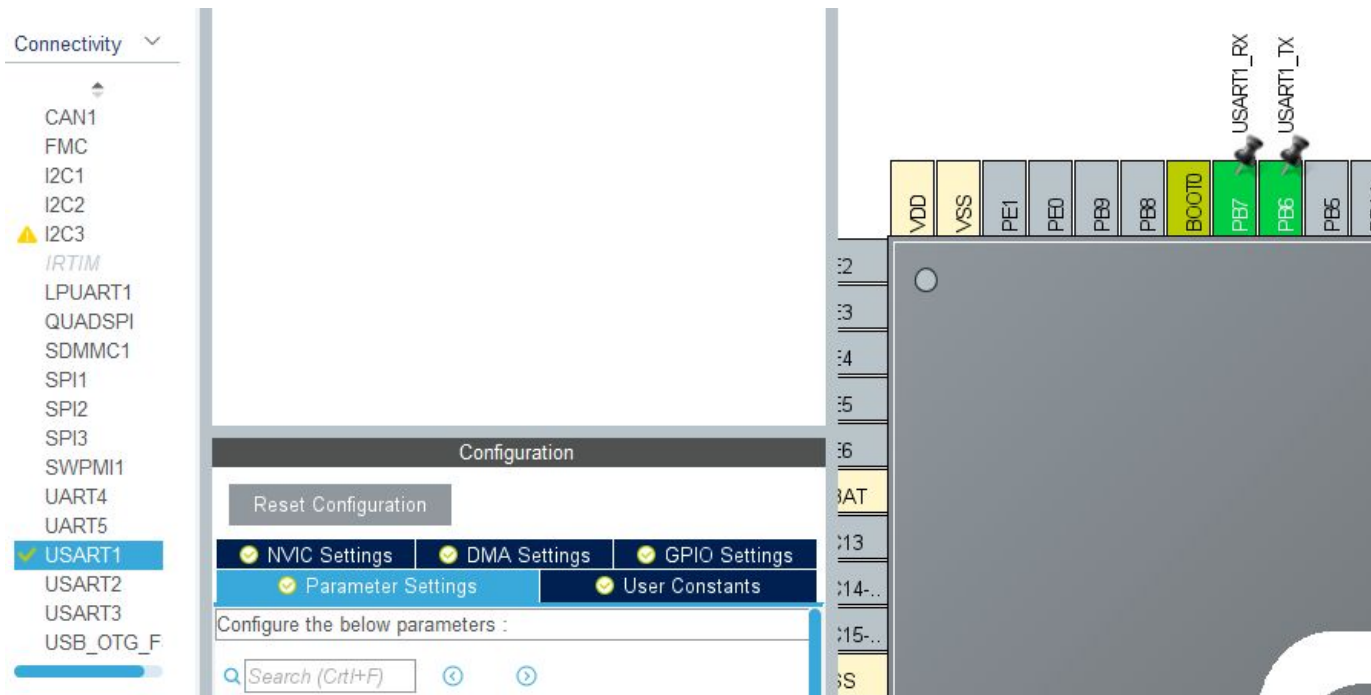
## Debug Uart 설정

90	PB4	TIM3_CH1	ARD.D5-PWM
91	PB5	GPIO_Output	SPSGRF-915-SPI3_CSN
92	PB6	USART1_TX	ST-LINK-UART1_TX
93	PB7	USART1_RX	ST-LINK-UART1_RX
94	BOOT0	Boot	BOOT0



# UART

## Debug Uart 설정



# UART

## UART - printf 코드추가

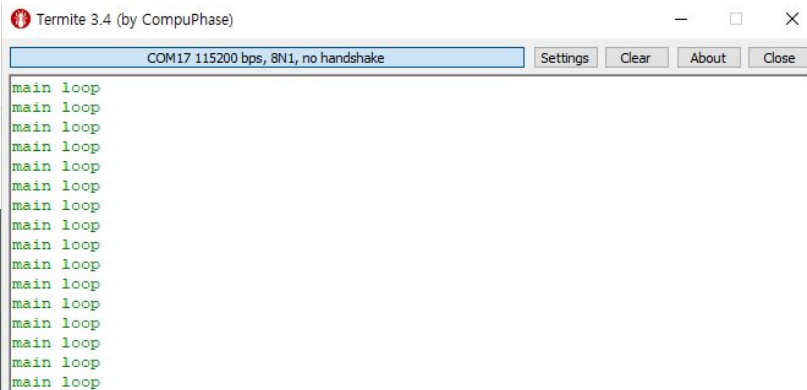
```
/* Private includes -----  
/* USER CODE BEGIN Includes */  
#include <stdio.h>  
/* USER CODE END Includes */
```

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART1_UART_Init();
/* USER CODE BEGIN 2 */
printf("START \r\n");
/* USER CODE END 2 */
```

```

/* USER CODE BEGIN 0 */
int __io_putchar(int ch)
{
    while (HAL_OK != HAL_UART_Transmit(&huart1, (uint8_t *) &ch, 1, 30000))
    {
    }
    return ch;
}
/* USER CODE END 0 */

```



# FreeRTOS

Example02 – Task parameter

# FreeRTOS

## 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`

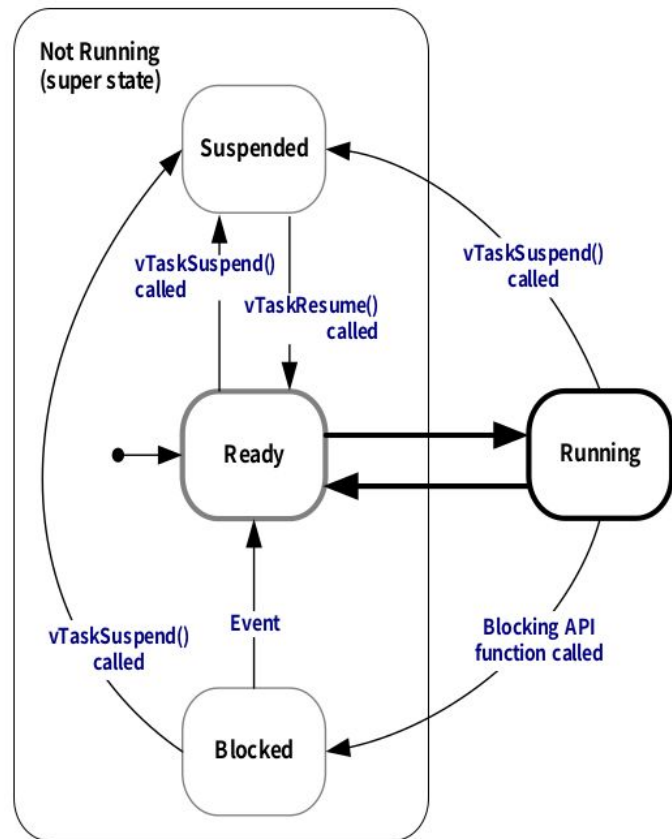
# FreeRTOS

Example03 – 우선순위

# FreeRTOS - Task Management

FreeRTOS의 어플리케이션은 Task로 이루어진다.

하나의 프로세서에는 하나의 task만 주어진 시간만큼 실행된다





# 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 우선순위 변경

### vTaskPrioritySet()

스케줄러가 시작한 후에 **task**의 우선 순위를 변경

- `INCLUDE_vTaskPrioritySet ("FreeRTOSConfig.h")` 설정 1

```
void vTaskPrioritySet( TaskHandle_t pxTask, UBaseType_t uxNewPriority );
```

pxTask	우선순위를 변경할 <b>task</b> 의 handle, NULL은 자신의 우선순위를 변경
uxNewPriority	변경할 우선순위 값

### uxTaskPriorityGet()

**task**의 우선순위를 쿼리

- `INCLUDE_uxTaskPriorityGet ("FreeRTOSConfig.h")` 설정 1

```
UBaseType_t uxTaskPriorityGet( TaskHandle_t pxTask );
```

pxTask	우선순위를 쿼리할 <b>task</b> 의 handle, NULL은 자신의 우선순위를 쿼리
return value	쿼리한 <b>task</b> 에대한 현재 우선순위 값

# FreeRTOS - Task Management

Example05 – 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

Example06 – Task 삭제

# FreeRTOS - Queue Management

## Queue

fixed size data item의 한정된 수만큼 보유

'length'는 queue item의 최대 수

각 data item의 length와 size는 queue가 생성될때 설정

데이터를 queue에 보낼때 queue에 데이터를 복사(Queue by copy)

## Queue의 Blocking

읽을때

- queue가 비어있다면 blocked state
- 다른 task나 interrupt에서 queue에 쓰거나, 정의한 block time이 다 지난 경우 자동으로 Ready state

쓸때

- queue가 full일때 block state
- 다른 task나 interrupt에서 queue를 읽거나, 정의한 block time이 다 지난 경우 자동으로 Ready state

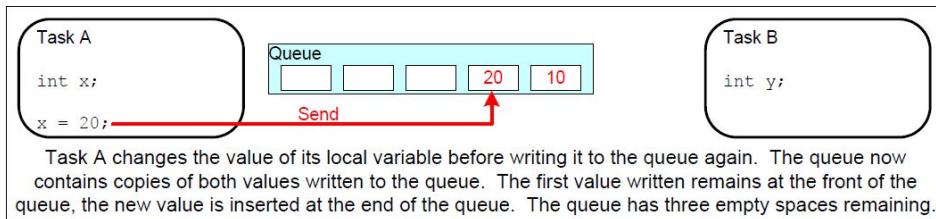
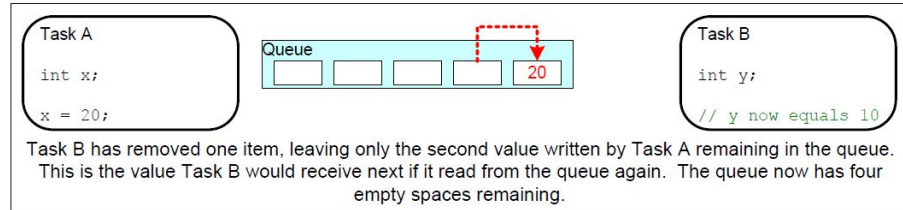
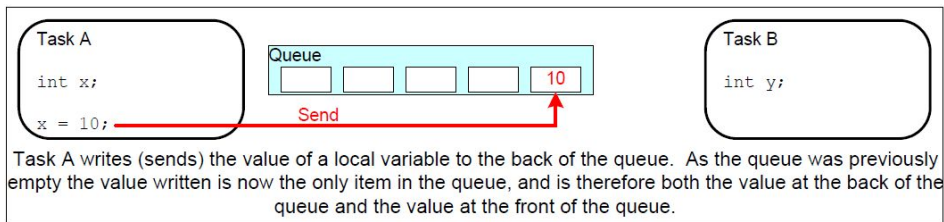
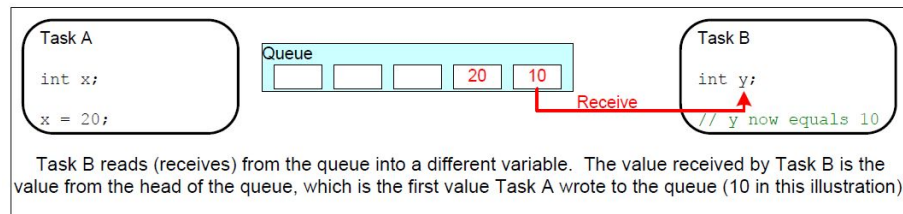
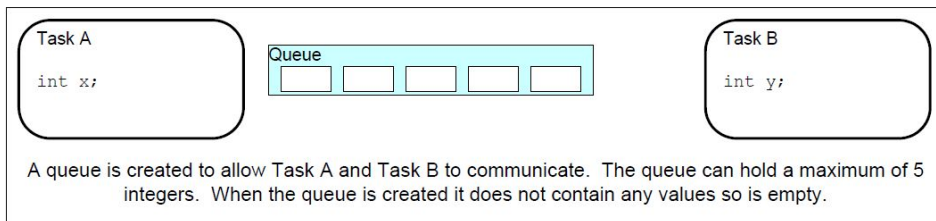
## Queue의 Unblocking 순서

특정 queue에 대해서 multiple reader/writer 있을때

- 우선순위가 높은 task 가 우선
- 우선순위가 같다면 기다린 시간이 가장 긴 task가 우선



# FreeRTOS - Queue Management



# FreeRTOS - Queue Management

## Queue 사용

xQueueCreate( ) : Queue 생성

QueueHandle_t xQueueCreate( UBaseType_t uxQueueLength, UBaseType_t uxItemSize );	
uxQueueLength	item의 최대수
uxItemSize	각 data item의 size(byte)
return value	NULL: 생성 실패 non-NULL: 생성 완료, 생성된 queue handle

xQueueSendToBack() : data를 tail에 write == xQueueSend() 함수

xQueueSendToFront() : data를 head에 write

BaseType_t xQueueSendToFront( QueueHandle_t xQueue, const void * pvItemToQueue, TickType_t xTicksToWait );	
BaseType_t xQueueSendToBack( QueueHandle_t xQueue, const void * pvItemToQueue, TickType_t xTicksToWait );	
xQueue	queue handle
pvItemToQueue	queue 복사할 data의 pointer
xTicksToWait	task가 Blocked state에 있을 시간의 최대값 0이면 바로 리턴 portMAX_DELAY은 무한대기 (INCLUDE_vTaskSuspend ("FreeRTOSConfig.h") 1로 설정)
return value	pdPASS : write 성공 errQUEUE_FULL : write 실패 (queue full or xTicksToWait expired)

# FreeRTOS - Queue Management

## Queue 사용

`xQueueReceive( )` : data를 queue에서 읽음

BaseType_t xQueueReceive( QueueHandle_t xQueue, void * const pvBuffer, TickType_t xTicksToWait );	
xQueue	queue handle
pvBuffer	queue에서 data를 복사할 memory의 pointer
xTicksToWait	task가 Blocked state에 있을 시간의 최대값 0이면 바로 리턴 portMAX_DELAY은 무한대기 (INCLUDE_vTaskSuspend ("FreeRTOSConfig.h") 1로 설정)
return value	pdPASS : write 성공 errQUEUE_EMPTY : read 실패 (queue empty or xTicksToWait expired)

`uxQueueMessagesWaiting( )` : queue에 대한 item의 수를 쿼리

UBaseType_t uxQueueMessagesWaiting( QueueHandle_t xQueue );	
xQueue	queue handle
return value	queue의 item의 수

# FreeRTOS - Queue Management

Example07 – Queue create, write, read

# Interrupt

## Interrupt?

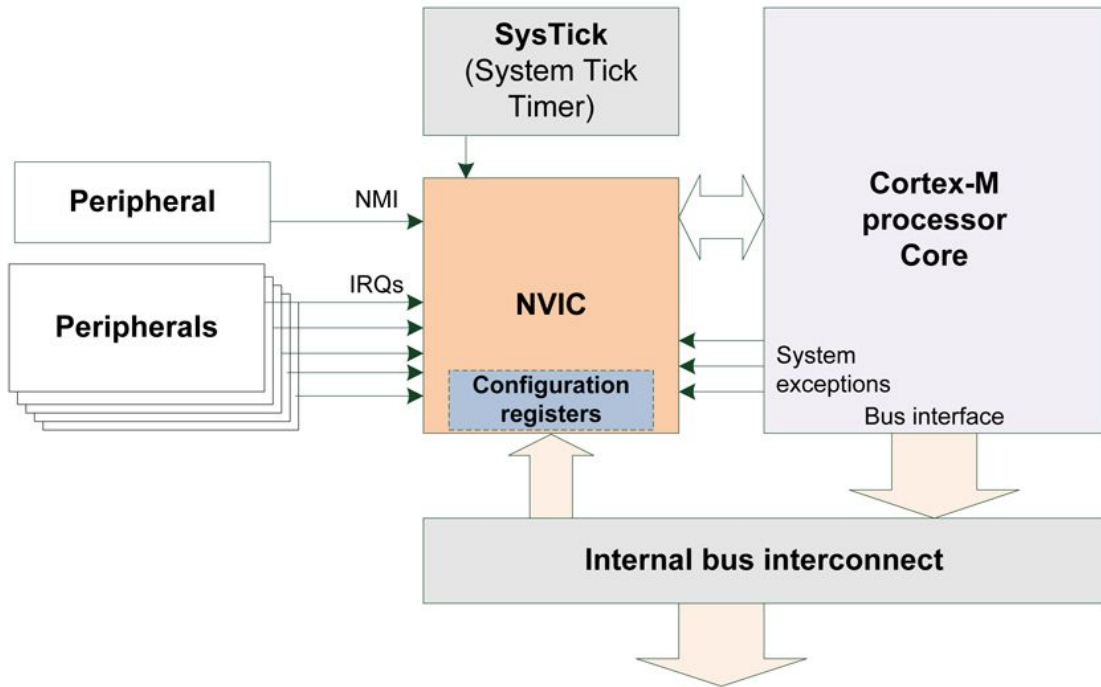
- 사전적의미 “끼어들다”, “중단시키다”
- 프로그램을 실행하는 도중에 예기치 않은 상황이 발생할 경우 현재 실행중인 작업을 중단하고 발생한 상황을 처리한 후 다시 실행중인 작업으로 복구



# Interrupt

## NVIC(Nested Vectored Interrupt Controller)

- interrupt에 대한 우선순위 설정
- interrupt에 대한 Vector Table 존재



# Interrupt

## Cortex-M Interrupt

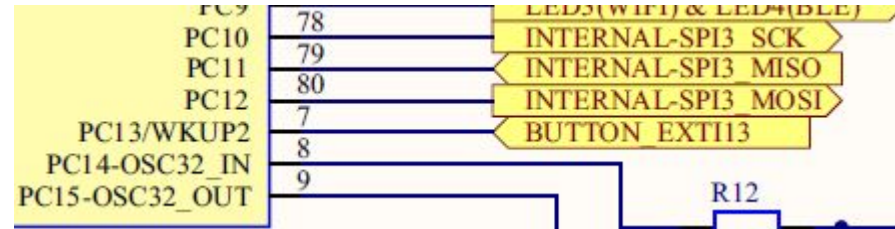
Exception Type	ARMv6-M (Cortex-M0/M0+/M1)	ARMv7-M (Cortex-M3/M4/M7)	ARMv8-M Baseline (Cortex-M23)	ARMv8-M Mainline (Cortex-M33)	Vector Table	Vector address offset (initial)
495		Not supported in Cortex-M3/M4/M7	Not supported in Cortex-M23		Interrupt#479 vector	10x000007BC
256						
255				Device Specific Interrupts	Interrupt#239 vector	10x000003FC
31	Device Specific Interrupts	Device Specific Interrupts	Device Specific Interrupts		Interrupt#31 vector	10x000000BC
17					Interrupt#1 vector	10x00000044
16					Interrupt#0 vector	10x00000040
15	SysTick	SysTick	SysTick	SysTick	SysTick vector	10x0000003C
14	PendSV	PendSV	PendSV	PendSV	PendSV vector	10x00000038
13	Not used	Not used	Not used	Not used	Not used	0x00000034
12		Debug Monitor	Not used	Debug Monitor	Debug Monitor vector	10x00000030
11	SVC	SVC	SVC	SVC	SVC vector	10x0000002C
10					Not used	0x00000028
9		Not used		Not used	Not used	0x00000024
8					Not used	0x00000020
7	Not used		Not used	SecureFault	SecureFault (ARMv8-M Mainline)	10x0000001C
6		Usage Fault		Usage Fault	Usage Fault vector	10x00000018
5		Bus Fault		Bus Fault	Bus Fault vector	10x00000014
4		MemManage (fault)		MemManage (fault)	MemManage vector	10x00000010
3	HardFault	HardFault	HardFault	HardFault	HardFault vector	10x0000000C
2	NMI	NMI	NMI	NMI	NMI vector	10x00000008
1					Reset vector	10x00000004
0					MSP initial value	0x00000000



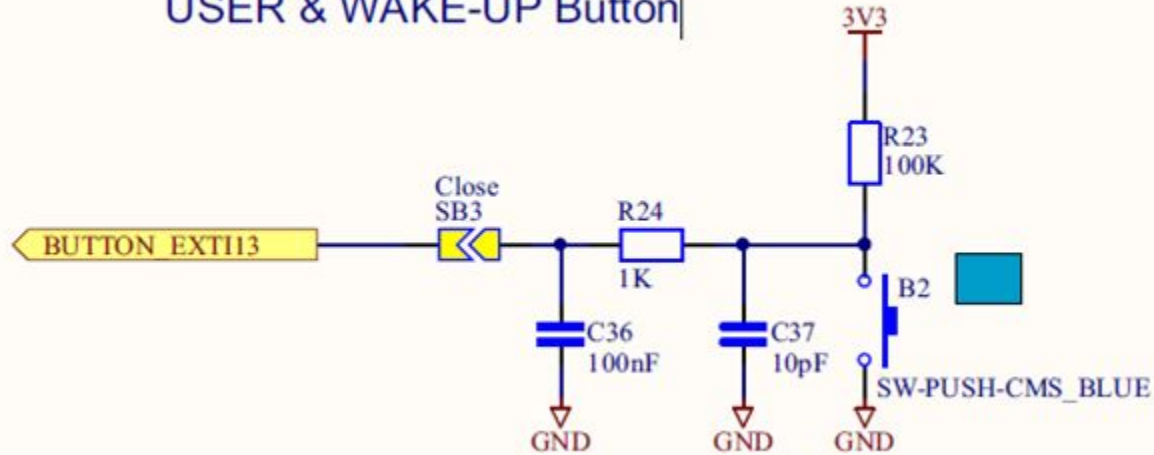


# Interrupt

## User Button Interrupt

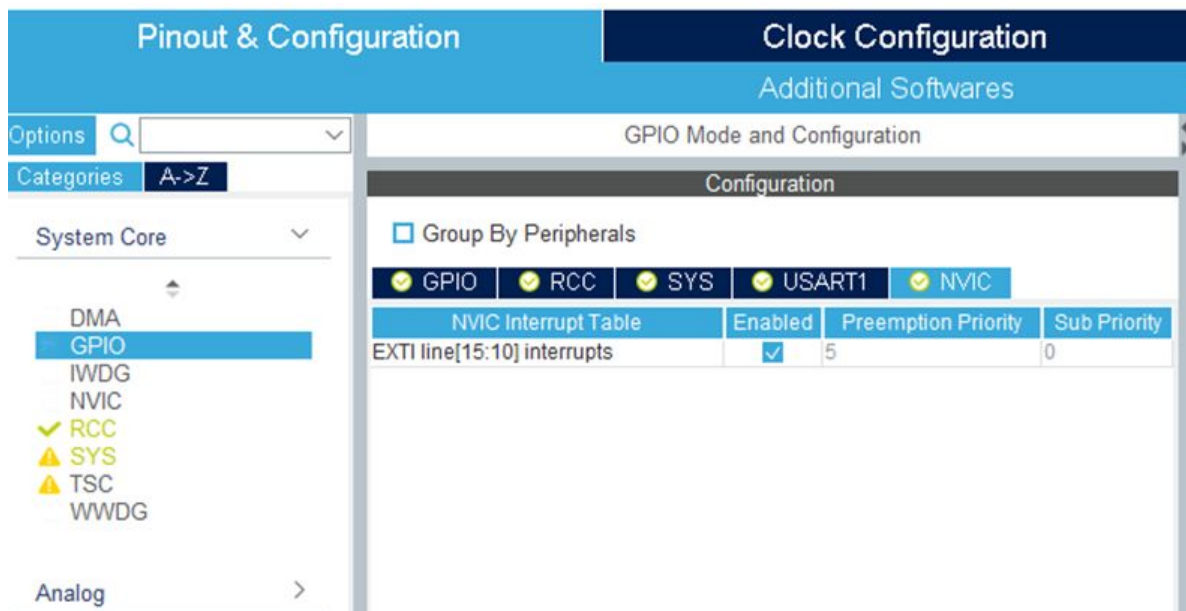
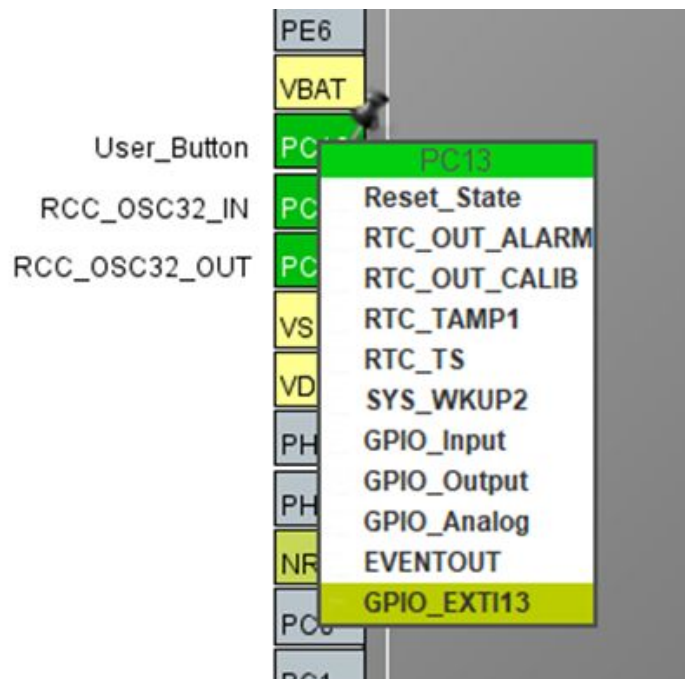


### USER & WAKE-UP Button



# Interrupt

## User Button Interrupt

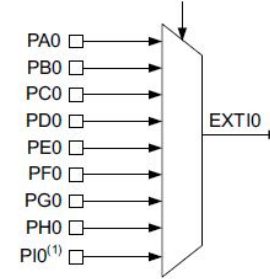


# Interrupt

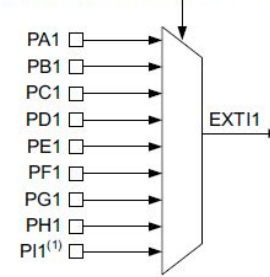
## EXTI

(Extended interrupts and events controller)

EXTI0[3:0] bits in the SYSCFG\_EXTICR1 register

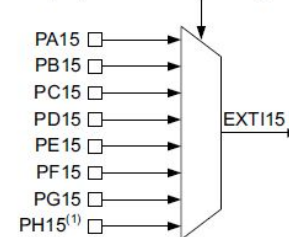


EXTI1[3:0] bits in the SYSCFG\_EXTICR1 register



⋮

EXTI15[3:0] bits in the SYSCFG\_EXTICR4 register



# Interrupt

## Example08 – Toggle LED


```
void EXTI15_10_IRQHandler(void)
{
    /* USER CODE BEGIN EXTI15_10_IRQn 0 */

    /* USER CODE END EXTI15_10_IRQn 0 */
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);
    /* USER CODE BEGIN EXTI15_10_IRQn 1 */

    /* USER CODE END EXTI15_10_IRQn 1 */
}

__weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    /* Prevent unused argument(s) compilation warning */
    UNUSED(GPIO_Pin);

    /* NOTE: This function should not be modified, when the callback is needed,
       the HAL_GPIO_EXTI_Callback could be implemented in the user file */
}
```



# FreeRTOS - Interrupt Management

## Interrupt Safe API

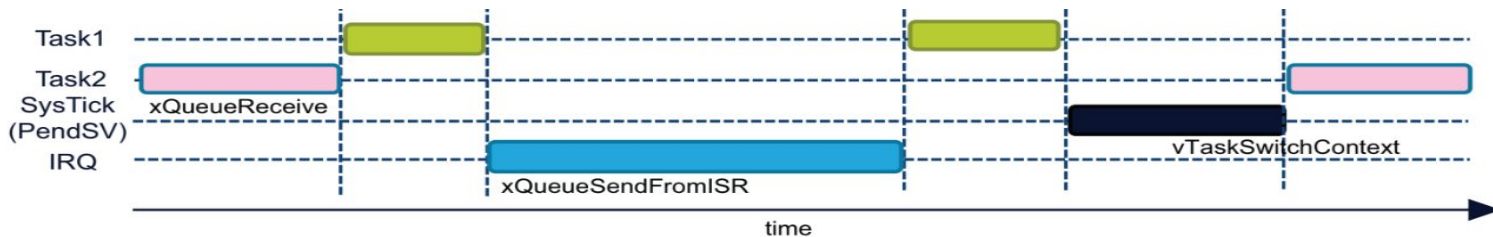
FreeRTOS에서는 task버전과 ISR버전의 두가지 버전의 함수를 제공  
ISR버전 함수는 "FromISR" 접미사가 붙는다

# FreeRTOS - Interrupt Management

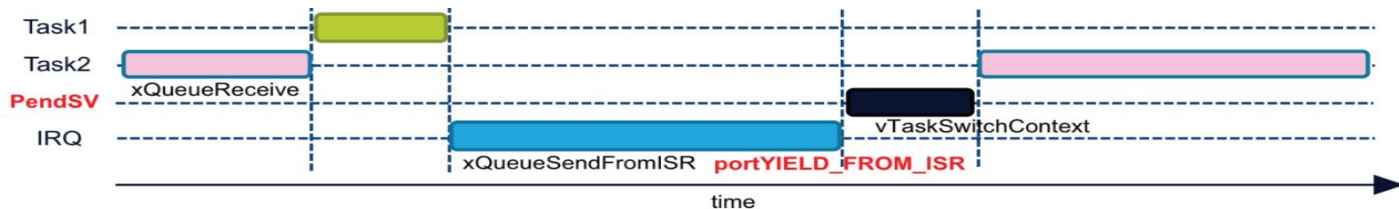
portYIELD\_FROM\_ISR() : ISR처리 이후 테스크 스케줄링을 바로 해야되는 경우에 호출

```
portYIELD_FROM_ISR( xHigherPriorityTaskWoken );
```

portYIELD\_FROM\_ISR 사용하지 않을 경우

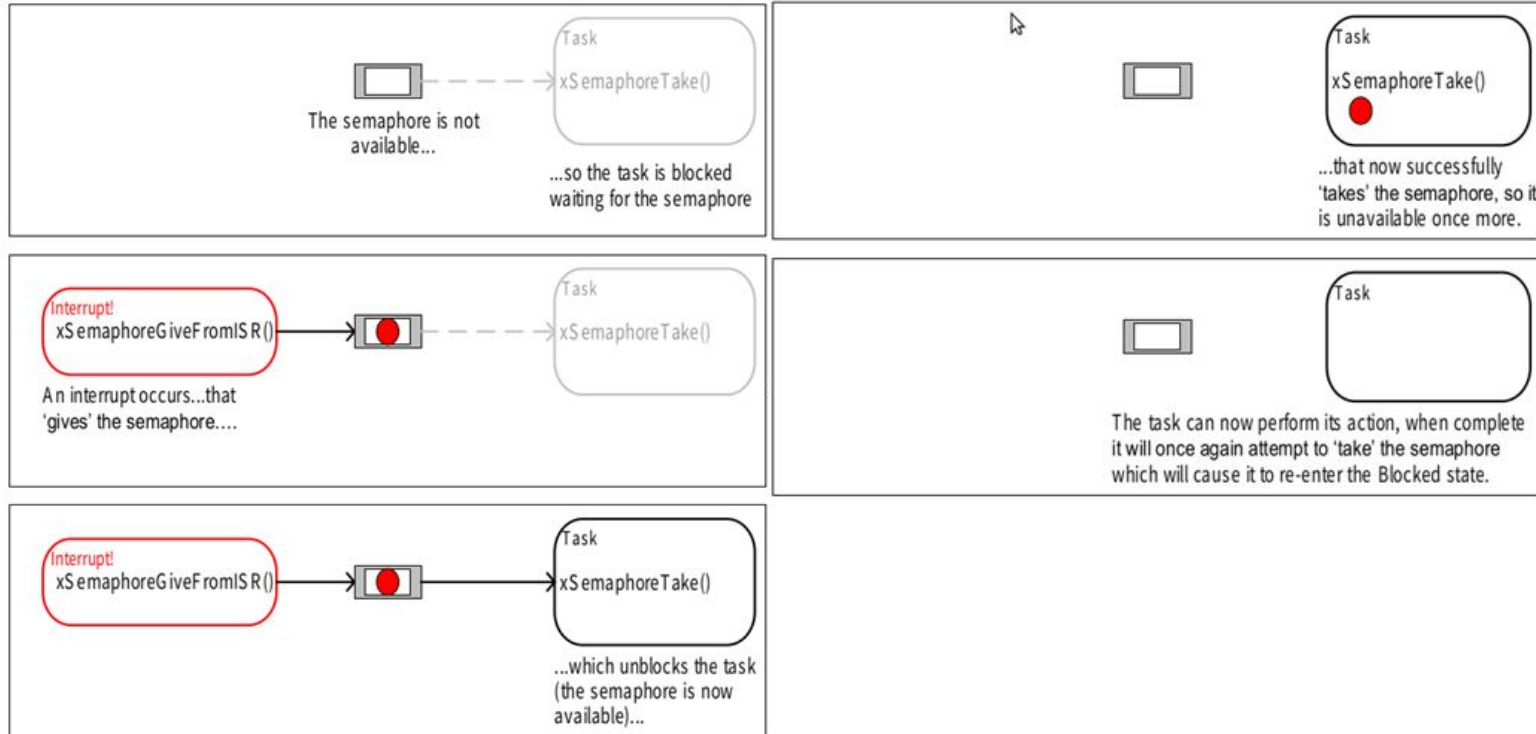


portYIELD\_FROM\_ISR 사용할 경우



# FreeRTOS - Semaphore

## Binary Semaphore



# FreeRTOS - Semaphore

## Binary Semaphore

### xSemaphoreCreateBinary() : binary semaphore 생성

SemaphoreHandle_t xSemaphoreCreateBinary( void );	
return value	NULL: 실패 non-NULL; 성공, SemaphoreHandle_t 타입 handle 반환

### xSemaphoreTake() : semaphore를 얻는다

BaseType_t xSemaphoreTake( SemaphoreHandle_t xSemaphore, TickType_t xTicksToWait );	
xSemaphore	Semaphore Handle
xTicksToWait	task가 Blocked state에 있을 시간의 최대값 0이면 바로 리턴 portMAX_DELAY은 무한대기 (INCLUDE_vTaskSuspend ("FreeRTOSConfig.h") 1로 설정
return value	pdPASS : semaphore taking 성공 pdFALSE: taking 가능한 semaphore가 없다(시간초과)



# FreeRTOS - Semaphore

## Binary Semaphore

`xSemaphoreGiveFromISR()` : semaphore를 준다.

BaseType_t xSemaphoreGiveFromISR( SemaphoreHandle_t xSemaphore, BaseType_t *pxHigherPriorityTaskWoken );	
xSemaphore	Semaphore Handle
pxHigherPriorityTaskWoken	단일 semaphore가 하나이상의 semaphore를 기다리는 block상태의 task를 가질수 있다. xSemaphoreGiveFromISR ()을 호출하면 task의 Blocked state를 벗어나고 unblock된 task의 우선 순위가 현재 실행중인 task 보다 높으면 내부적으로 xSemaphoreGiveFromISR ()이 pxHigherPriorityTaskWoken을 pdTRUE로 설정한다.
return value	pdPASS: 성공 pdFAIL: 실패

# FreeRTOS - Semaphore

Example09 – binary semaphore를 이용한 task와 interrupt 동기화

# FreeRTOS - Semaphore

## Counting Semaphore

binary semaphore는 길이가 하나인 queue, counting semaphore는 하나 이상의 길이를 가지는 queue로 볼수 있다.

- configUSE\_COUNTING\_SEMAPHORES ("FreeRTOSConfig.h") 설정 1

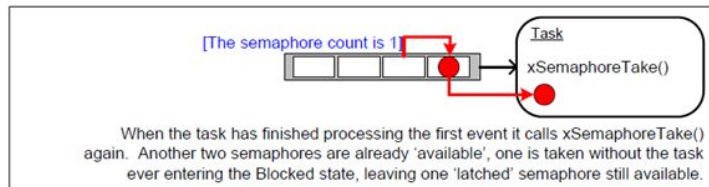
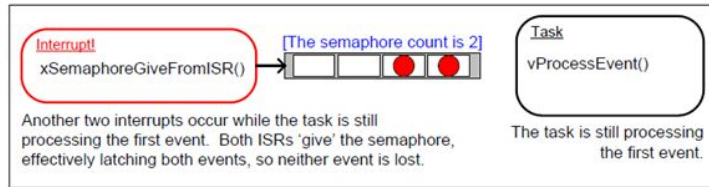
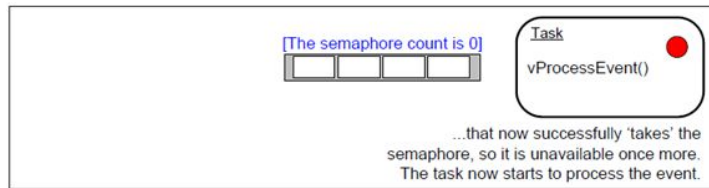
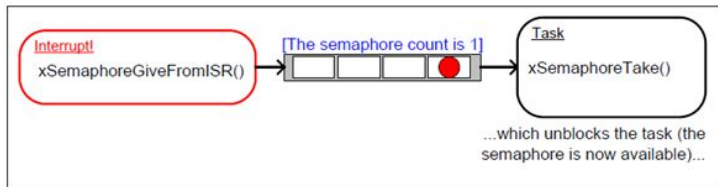
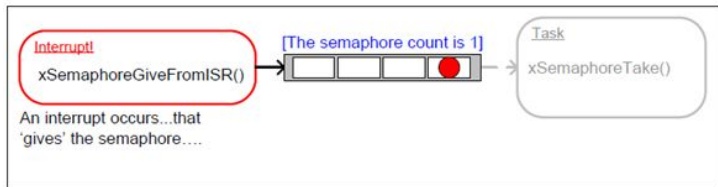
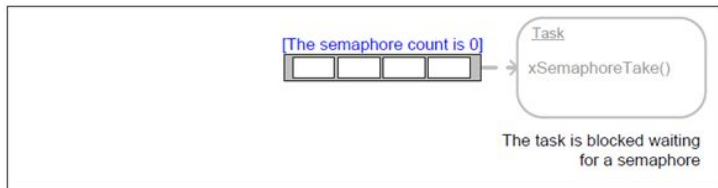
일반적으로 다음의 두경우에 사용

- 1.Counting event – event count에 사용(초기화 count value는 0)
- 2.Resource management - 사용가능한 resource의 수를 표시 (초기화 count value는 resource의 수)

# FreeRTOS - Semaphore

## Counting Semaphore

### Counting Event 예



# FreeRTOS - Semaphore

## Counting Semaphore

xSemaphoreCreateCounting() : counting semaphore 생성

SemaphoreHandle_t xSemaphoreCreateCounting( UBaseType_t uxMaxCount, UBaseType_t uxInitialCount );	
uxMaxCount	Semaphore의 최대 수
uxInitialCount	Semaphore 생성시 초기화 count 값
return value	NULL: 실패 non-NULL: 성공, SemaphoreHandle_t 타입 handle 반환

# FreeRTOS - Semaphore

Example10 – counting semaphore를 이용한 task와 interrupt 동기화