FreeRTOS for ESP32

ESP32 Feature

Wi-Fi Key Features

- 802.11 b/g/n
- 802.11 n (2.4 GHz), up to 150 Mbps

BT Key Features

- Compliant with Bluetooth v4.2 BR/EDR and BLE specifications
- Class-1, class-2 and class-3 transmitter without external power amplifier
- Enhanced Power Control
- +12 dBm transmitting power
- NZIF receiver with –94 dBm BLE sensitivity
- Adaptive Frequency Hopping (AFH)
- Standard HCI based on SDIO/SPI/UART
- High-speed UART HCI, up to 4 Mbps
- Bluetooth 4.2 BR/EDR BLE dual mode controller
- Synchronous Connection-Oriented/Extended (SCO/eSCO)
- CVSD and SBC for audio codec
- Bluetooth Piconet and Scatternet
- Multi-connections in Classic BT and BLE
- Simultaneous advertising and scanning

ESP32 Feature

MCU and Advanced Features

1.CPU and Memory
• Xtensa® single-/dual-core 32-bit LX6 microprocessor(s), up to 600 MIPS (200 MIPS for ESP32-S0WD/ESP32-U4WDH, 400 MIPS for ESP32-D2WD) (80, 160, 240 MHz)
• 448 KB ROM
• 520 KB SRAM
• 16 KB SRAM in RTC
• QSPI supports multiple flash/SRAM chips

2.Clocks and Timers
• Internal 8 MHz oscillator with calibration
• Internal RC oscillator with calibration
• External 2 MHz ~ 60 MHz crystal oscillator (40 MHz only for Wi-Fi/BT functionality)
• External 32 kHz crystal oscillator for RTC with calibration
• Two timer groups, including 2 × 64-bit timers and 1 × main watchdog in each group
• One RTC timer
• RTC watchdog

3.Advanced Peripheral Interfaces
• 34 × programmable GPIOs
• 12-bit SAR ADC up to 18 channels
• 2 × 8-bit DAC
• 10 × touch sensors
• 4 × SPI
• 2 × I²S
• 2 × I²C
• 3 × UART
• 1 host (SD/eMMC/SDIO)
• 1 slave (SDIO/SPI)
• Ethernet MAC interface with dedicated DMA and IEEE 1588 support
• CAN 2.0
• IR (TX/RX)
• Motor PWM
• LED PWM up to 16 channels
• Hall sensor

Security
• Secure boot
• Flash encryption
• 1024-bit OTP, up to 768-bit for customers
• Cryptographic hardware acceleration:
– AES
– Hash (SHA-2)
– RSA
– ECC
– Random Number Generator (RNG)

ESP32 개발환경 설정

References

- esp32_datasheet_en.pdf : CPU DataSheet
- esp32_technical_reference_manual_en.pdf : CPU Programming Guide
- esp32-wroom-32_datasheet_en.pdf : CPU Module DataSheet
- esp32_devkitc_v4-sch.pdf : ESP32 Devkit Schematics
- SDK Guide : https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html

ESP-IDF 개발도구 설치

64비트 윈도우버전만 지원, 32비트 윈도우는 레거시 GNU Make Build System 사용
64비트 윈도우버전에서 작업 진행할 것

Python, Git, Cross Compiler

크로스 컴파일러
크로스 컴파일러(cross compiler)는 컴파일러가 실행되는 플랫폼이 아닌 다른 플랫폼에서 실행 가능한 코드를 생성할 수 있는 컴파일러이다. 크로스 컴파일러 툴은 임베디드 시스템 혹은 여러 플랫폼에서 실행파일을 생성하는데 사용된다.

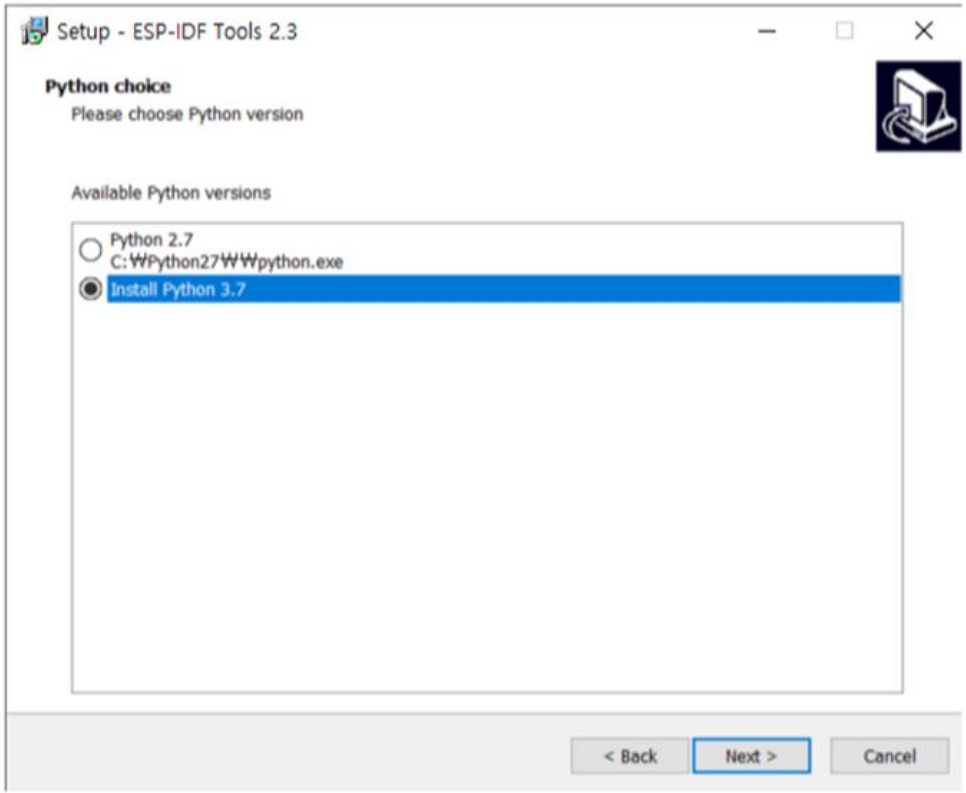
ESP-IDF 도구 설치 프로그램

ESP-IDF 필수 구성 요소를 설치하는 가장 쉬운 방법은 다음 URL에서 ESP-IDF 도구 설치 관리자를 다운로드하는 것입니다.

https://dl.espressif.com/dl/esp-idf-tools-setup-2.3.exe

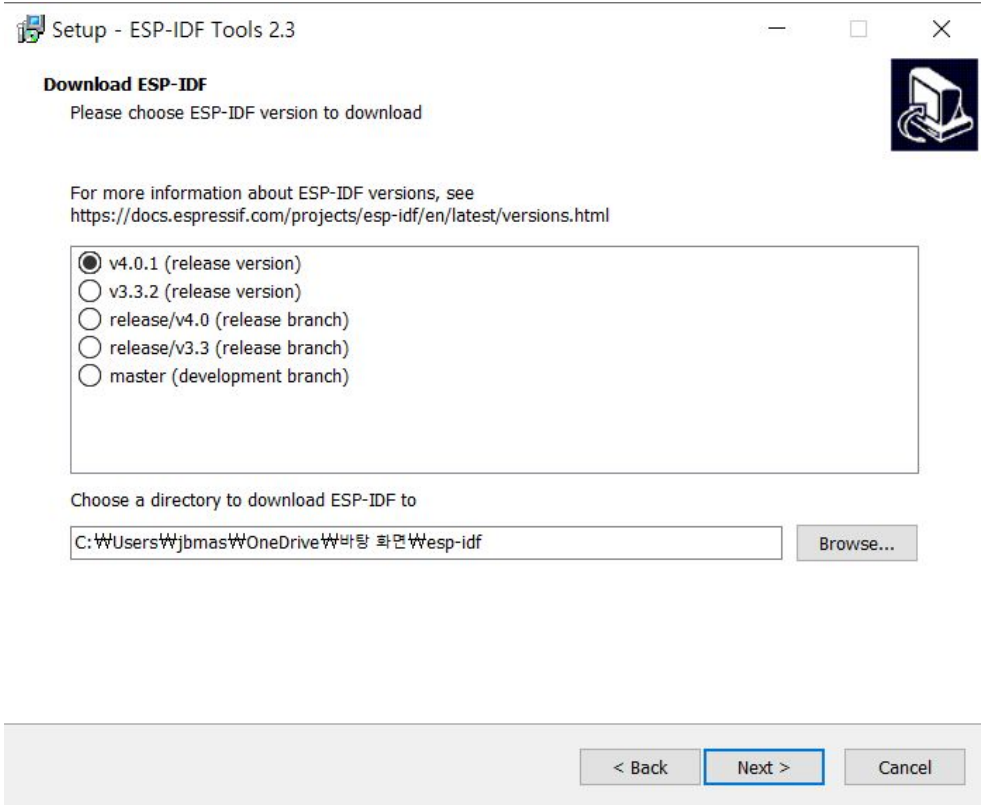
크로스 컴파일러, OpenOCD, cmake 및 Ninja 빌드 도구가 포함됩니다. 설치 프로그램은 Python 3.7 및 Git For Windows 용 설치 프로그램이 컴퓨터에 아직 설치되어 있지 않은 경우 다운로드하여 실행할 수 있습니다 .

ESP-IDF 개발도구 설치



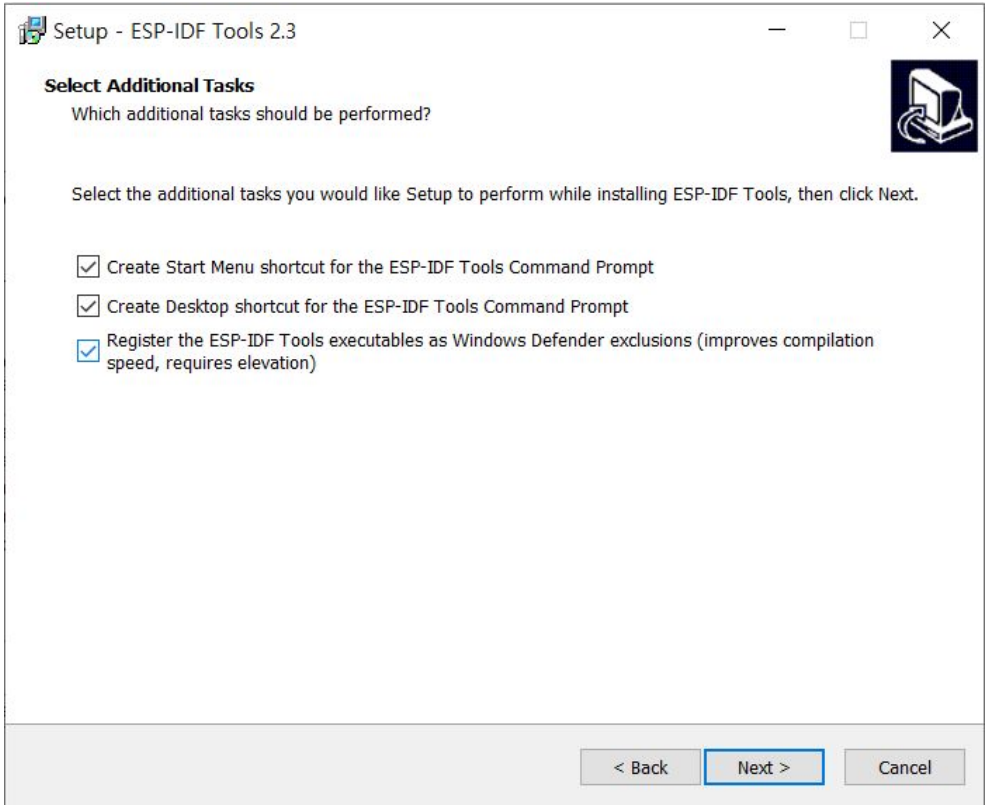
기존에 구축한 시스템과 별도로 설치하는 것이유리합니다.
특히 파이썬 버전을 맞춰두고 빌드하는 다른 시스템이 있으면 필히 Intall 지정해서 설치하시기 바랍니다.

ESP-IDF 개발도구 설치

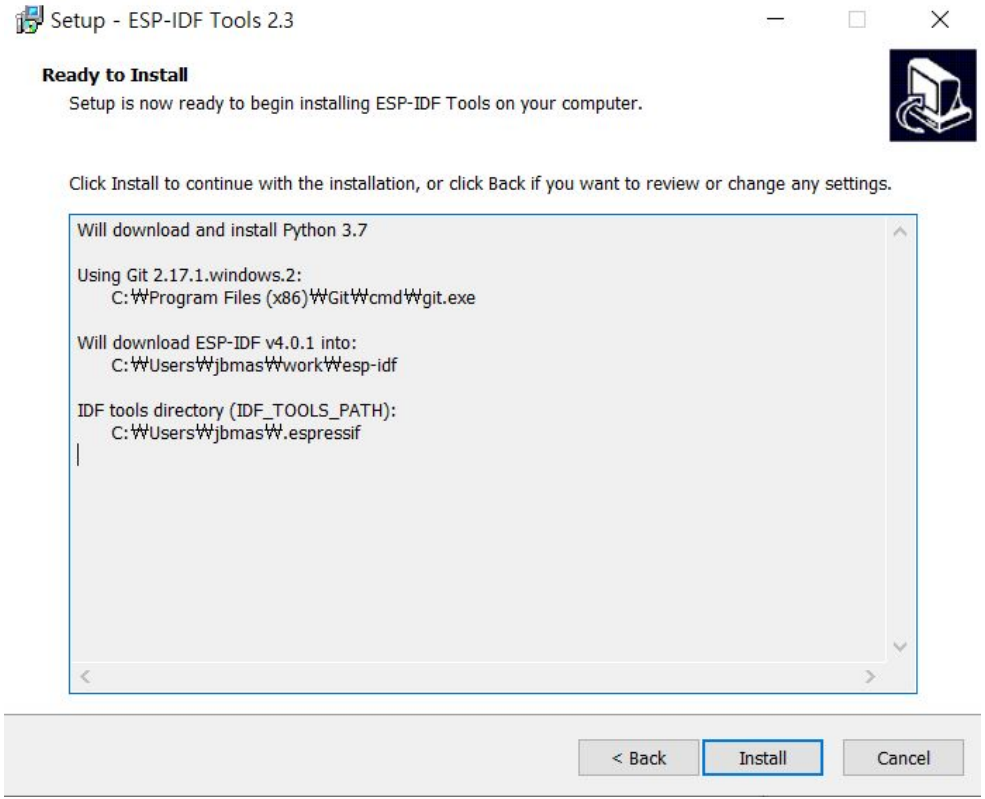


V4.0.1 (릴리즈버전 사용)
다운로드 받을 디렉토리 주소 체크
(한글 디렉토리X)

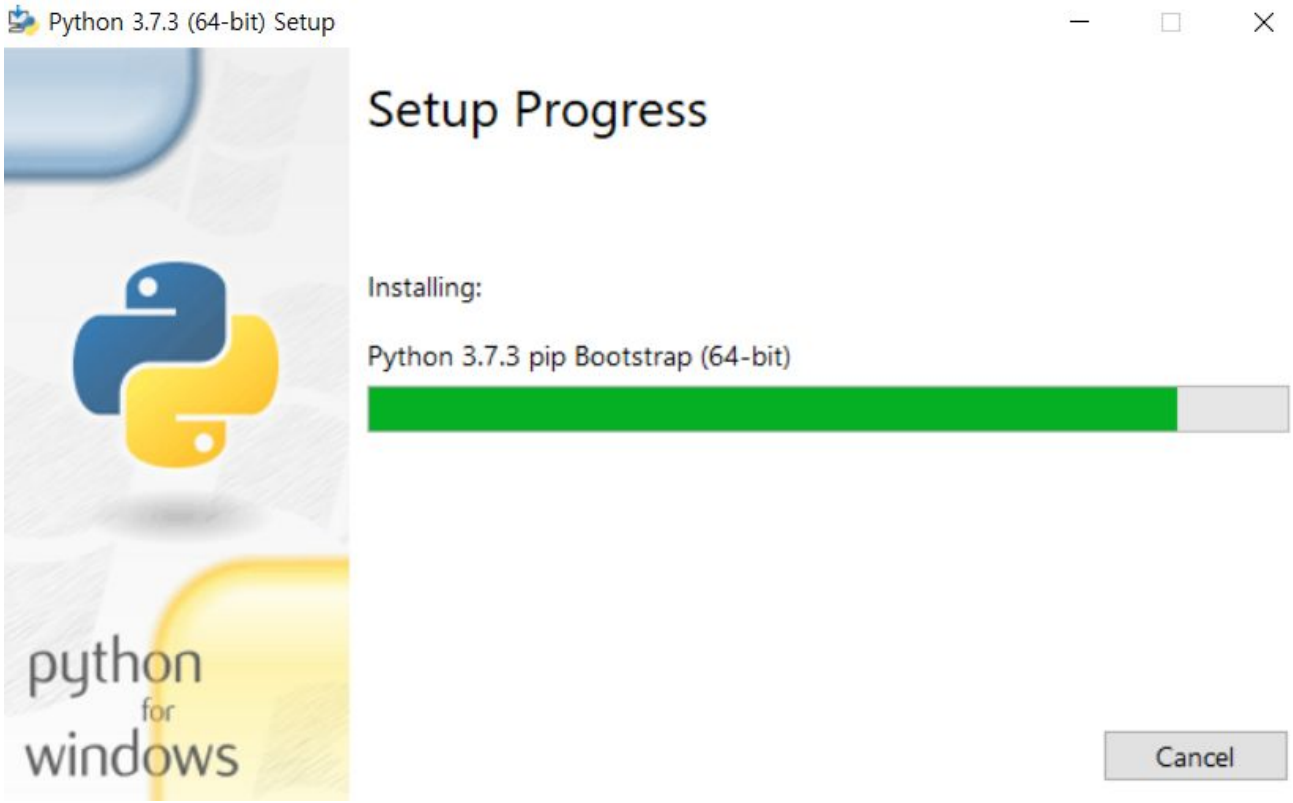
ESP-IDF 개발도구 설치



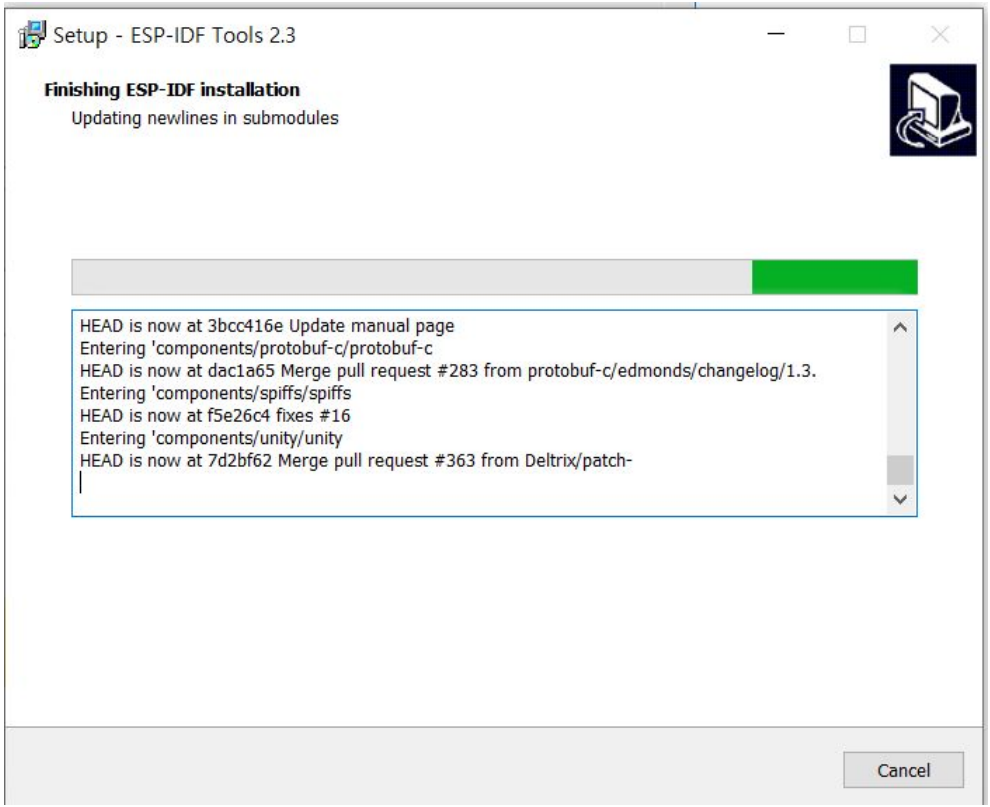
ESP-IDF 개발도구 설치



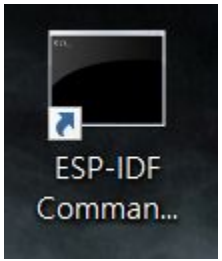
ESP-IDF 개발도구 설치

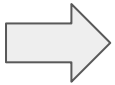


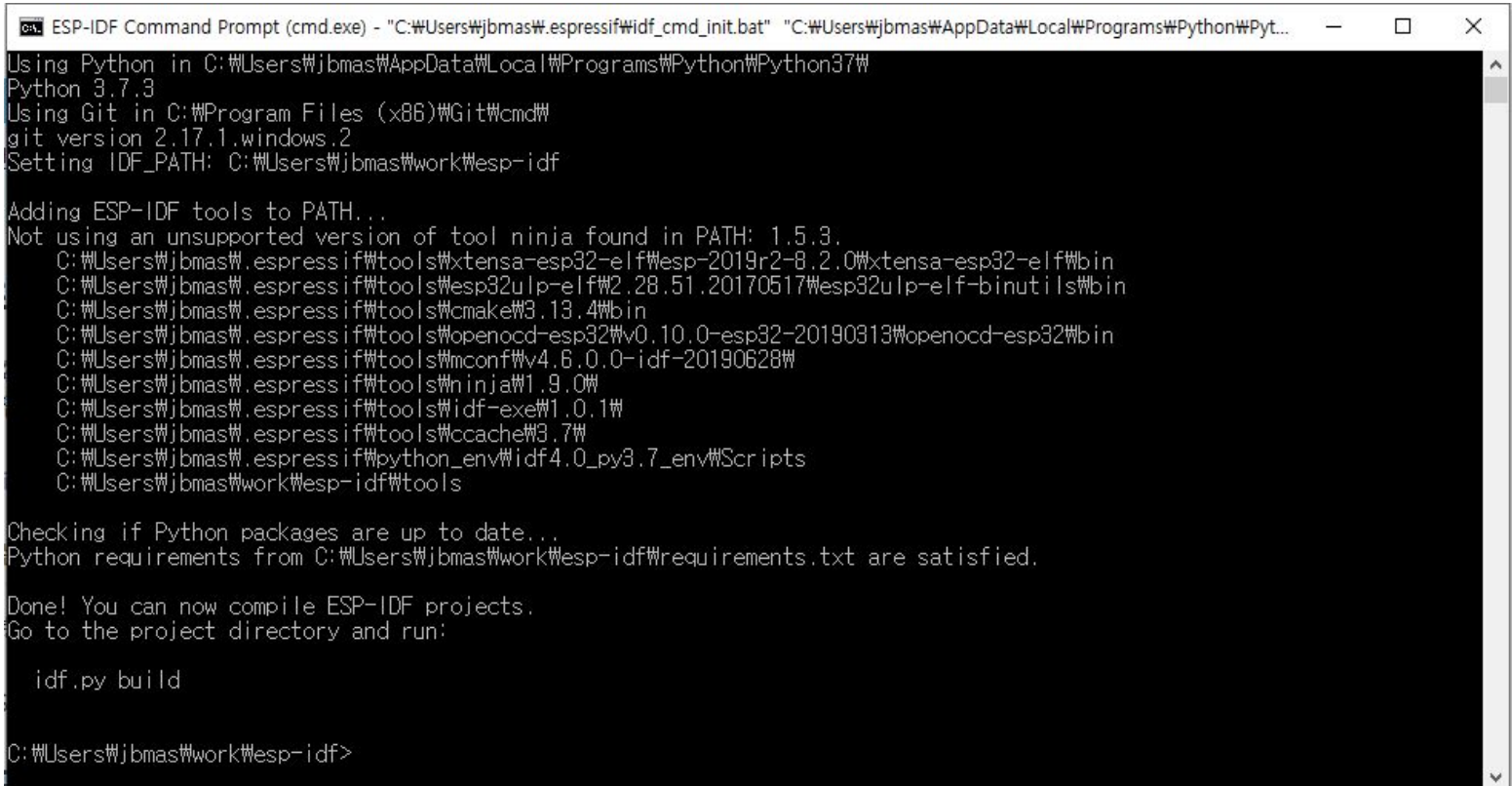
ESP-IDF 개발도구 설치



ESP-IDF 개발도구 설치







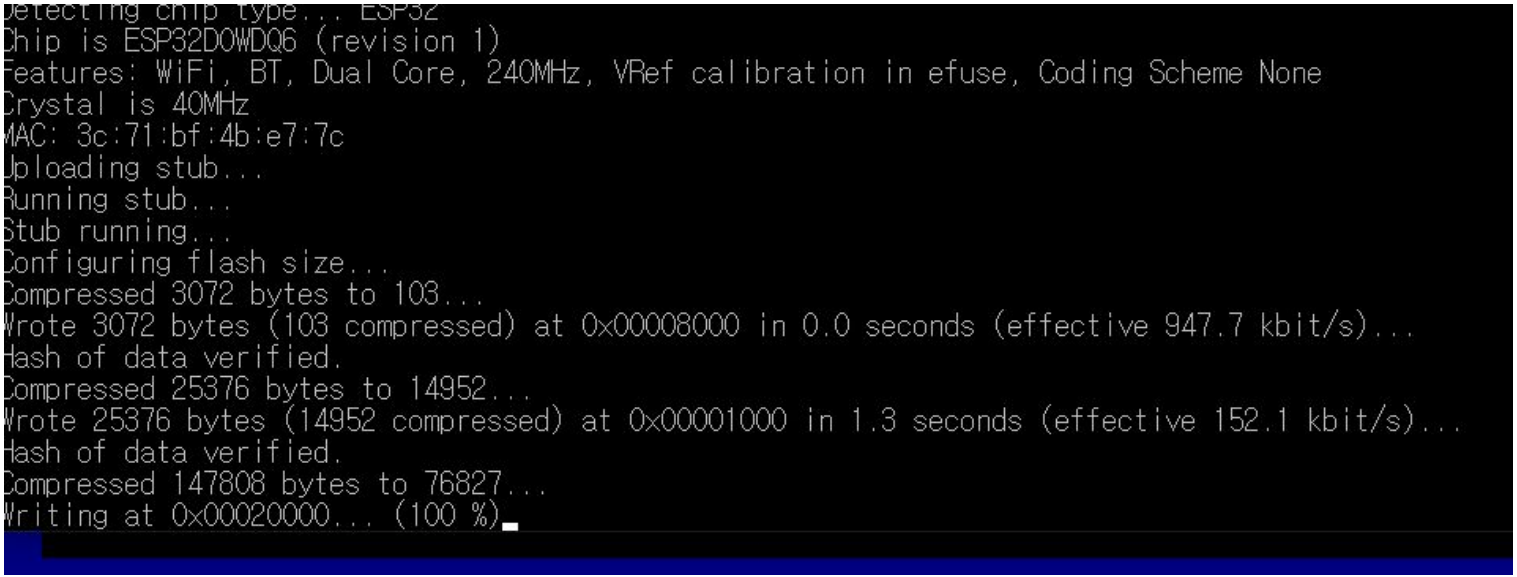
ESP-IDF 실습환경 설정

<ESP-IDF directory>/esp-idf/examples/get-started/hello_world
원하는 작업 디렉터리에 복사

보드 연결후 시리얼 포트확인

ESP-IDF 실습환경 설정

cmd창에서 <복사한 작업 디렉터리>로 이동후
> idf.py build
>idf.py -p com9 flash



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

FreeRTOS 주요 특징

- Preemptive or cooperative real-time kernel
- Tiny footprint (less than 10KB ROM)
- low power를 위한 tickless
- task간 통신을 위한 동기화기능 (queue, event 등)
- task 스케줄링을 위한 S/W timer

FreeRTOS - 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

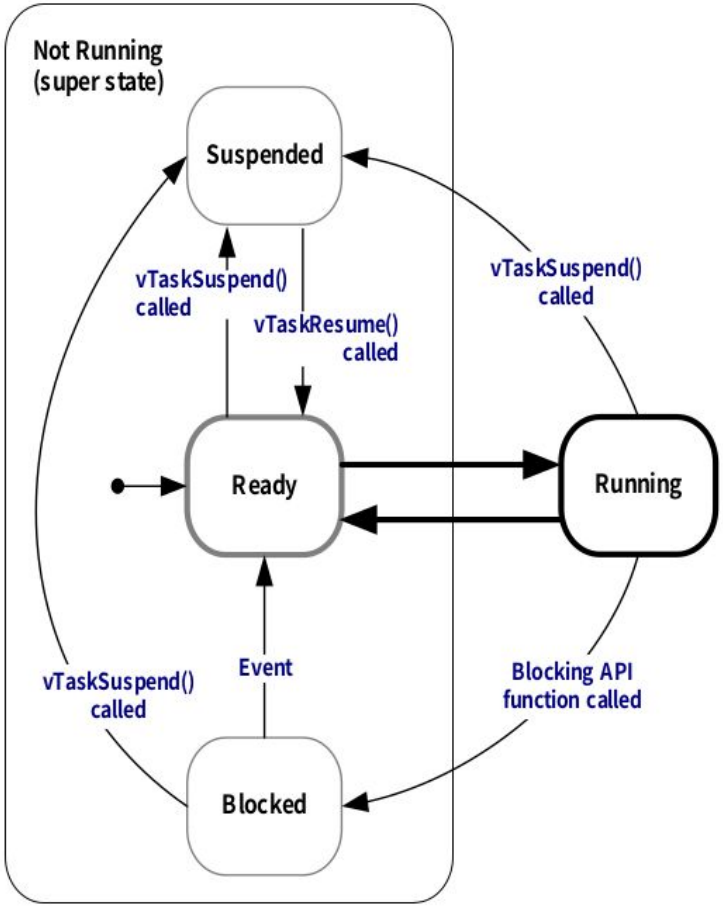
매크로 이름은 대문자이고 접두어는 소문자이다

"TickType_t" : tick 인터럽트에서 호출하는 Tick 변수 타입

"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

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: 이름의 최대 길이(기본값 16)
sStackDepth	task의 stack크기(byte)
pvParameters	task함수에 전달되는 파라메터
uxPriority	task의 실행 우선순위
pxCreatedTask	task handle
return value	pdPASS: 성공 다른 값: 실패(projdefs.h 에 정의)

Task함수
void ATaskFunction(void *pvParameters);

FreeRTOS - Task

Task Delay

void vTaskDelay(TickType_t xTicksToDelay);	
xTicksToDelay	tick interrupt의 수

Example01 – Task 생성

```
#include <stdio.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

void vTask1(void *pvParameters)
{
    for(;;)
    {
        printf("VTask1\n");
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}
```

```
void app_main(void)
{
    int i = 0;

    printf("Hello world!\n");

    xTaskCreate(vTask1, "Task 1", 1024, NULL, 3, NULL);

    while(1)
    {
        printf("Restarting in %d seconds...\n", i);
        i++;
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

ESP-IDF의 void app_main()

esp-idf/components/esp32/cpu_start.c

```
void start_cpu0_default(void)
{
    xTaskCreatePinnedToCore(&main_task, "main", ...);
}

static void main_task(void* args)
{
    app_main();
}
```

FreeRTOS - Task

‘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

Example02 – Task parameter

```
#include <stdio.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

const char *pcTextForTask1 = "vTask1";
const char *pcTextForTask2 = "vTask2";

void vTask1(void *pvParameters)
{
    char *pcTaskName;
    pcTaskName = (char*)pvParameters;

    for(;;)
    {
        printf("%s\n",pcTaskName);
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}
```

```
void app_main(void)
{
    int i = 0;

    printf("Hello world!\n");

    xTaskCreate(vTask1, "Task 1", 1024, (void*)pcTextForTask1, 3, NULL);
    xTaskCreate(vTask1, "Task 2", 1024, (void*)pcTextForTask2, 3, NULL);

    while(1)
    {
        printf("Restarting in %d seconds...\n", i);
        i++;
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

Example03 – 우선순위

```
#include <stdio.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

const char *pcTextForTask1 =
"1111111111111111111111111111111111111111111";
const char *pcTextForTask2 =
"2222222222222222222222222222222222222222222";

void vTask1(void *pvParameters)
{
    char *pcTaskName;
    pcTaskName = (char*)pvParameters;

    for(;;)
    {
        printf("%s\n",pcTaskName);
        //vTaskDelay(1 / portTICK_PERIOD_MS);
    }
}
```

```
void app_main(void)
{
    int i = 0;

    printf("Hello world!\n");

    xTaskCreate(vTask1, "Task 1", 1024, (void*)pcTextForTask1, 3, NULL);
    xTaskCreate(vTask1, "Task 2", 1024, (void*)pcTextForTask2, 4, NULL);

    while(1)
    {
        printf("Restarting in %d seconds...\n", i);
        i++;
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

FreeRTOS - Task

Task 우선순위 변경

vTaskPrioritySet()

스케줄러가 시작한 후에 task의 우선 순위를 변경

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

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

Example04 – Task 우선순위 변경

```
#include <stdio.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

/* handle of Task. */
TaskHandle_t xTask2Handle;

void vTask1(void *pvParameters)
{
    UBaseType_t uxPriority;
    uxPriority = uxTaskPriorityGet( NULL );

    for(;;)
    {
        printf("Task1 is running\r\n");

        /* Setting the Task2 priority */
        printf("About to raise the Task2 priority\r\n");
        vTaskPrioritySet( xTask2Handle, ( uxPriority + 1 ) );
    }
}
```

```
void vTask2(void *pvParameters)
{
    UBaseType_t uxPriority;
    uxPriority = uxTaskPriorityGet( NULL );

    for(;;)
    {
        printf("Task2 is running\r\n");

        /* Setting the Task2 priority */
        printf("About to lower the Task2 priority\r\n");
        vTaskPrioritySet( NULL, ( uxPriority - 2 ) );
    }
}
```

Example04 – Task 우선순위 변경

```
void app_main(void)
{
    printf("Hello world!\n");

    xTaskCreate(vTask1, "Task 1", 1024, NULL, 4, NULL);
    xTaskCreate(vTask2, "Task 2", 1024, NULL, 3, &xTask2Handle);

    while(1)
    {
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

FreeRTOS - Task

Task 삭제

vTaskDelete()

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

Example05 – Task 삭제

```
#include <stdio.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

/* handle of Task. */
TaskHandle_t xTask1Handle;

void vTask1(void *pvParameters)
{
    printf( "Task2 is running and delete\r\n" );
    vTaskDelete( xTask1Handle );
}
```

```
void app_main(void)
{
    printf("Hello world!\n");

    while(1)
    {
        printf("Task1 is running\r\n");
        xTaskCreate(vTask1, "Task 1", 1024, NULL, 3, &xTask1Handle);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

FreeRTOS - Queue

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

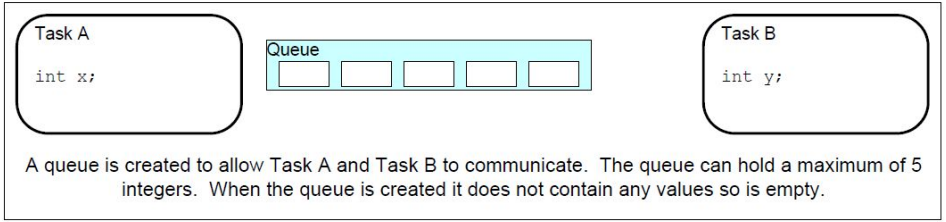
읽을때
- queue가 비어있다면 blocked state
- 다른 task나 interrupt에서 queue에 쓰거나, 정의한 block time이 다 지난 경우 자동으로 Ready state

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

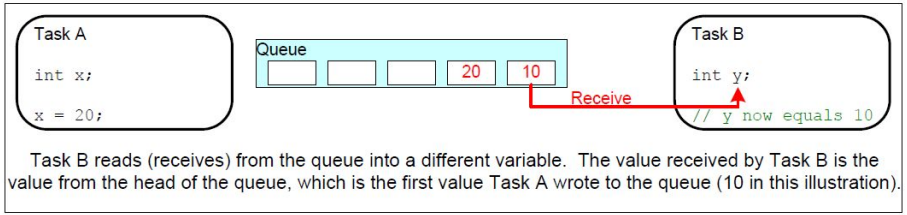
Queue의 Unblocking 순서

특정 queue에 대해서 multiple reader/writer 있을때
- 우선순위가 높은 task 가 우선
- 우선순위가 같다면 기다린 시간이 가장 긴 task가 우선

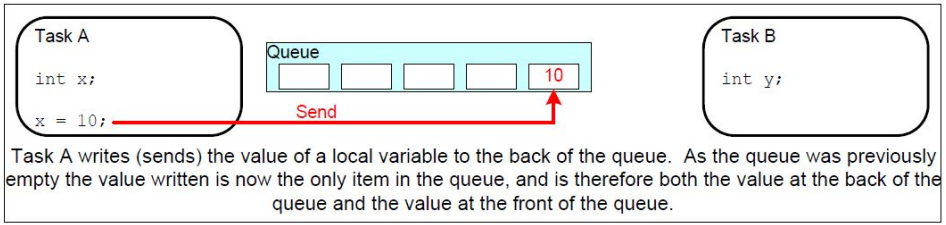
FreeRTOS - Queue



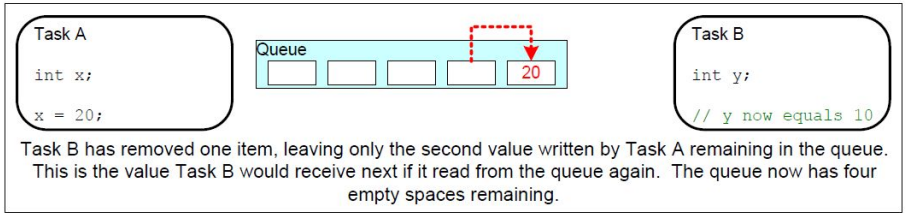
A queue is created to allow Task A and Task B to communicate. The queue can hold a maximum of 5 integers. When the queue is created it does not contain any values so is empty.



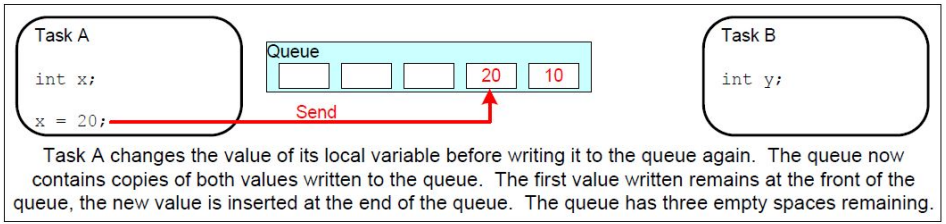
Task B reads (receives) from the queue into a different variable. The value received by Task B is the value from the head of the queue, which is the first value Task A wrote to the queue (10 in this illustration).



Task A writes (sends) the value of a local variable to the back of the queue. As the queue was previously empty the value written is now the only item in the queue, and is therefore both the value at the back of the queue and the value at the front of the queue.



Task B has removed one item, leaving only the second value written by Task A remaining in the queue. This is the value Task B would receive next if it read from the queue again. The queue now has four empty spaces remaining.



Task A changes the value of its local variable before writing it to the queue again. The queue now contains copies of both values written to the queue. The first value written remains at the front of the queue, the new value is inserted at the end of the queue. The queue has three empty spaces remaining.

FreeRTOS - Queue

Queue사용

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
return value	pdPASS : write 성공 errQUEUE_FULL : write 실패 (queue full or xTicksToWait expired)

FreeRTOS - Queue

Queue사용

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
return value	pdPASS : write 성공 errQUEUE_EMPTY : read 실패 (queue empty or xTicksToWait expired)

uxQueueMessagesWaiting( ) : queue에대한 item의 수를 쿼리

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

Example06 – Queue create, write, read

```
#include <stdio.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"

static void vSenderTask( void *pvParameters );
/* handle of Task. */
QueueHandle_t xQueue;

void app_main(void)
{
    printf("Hello world!\n");

    /* Create Queue */
    xQueue = xQueueCreate( 5, sizeof( int32_t ) );

    if(xQueue != NULL)
    {
        /* Create the thread(s) */
        xTaskCreate( vSenderTask, "Sender1", 1024, ( void * ) 100, 1, NULL );
    }
    else
    {
        printf("Create Queue failed\r\n");
    }
}

static void vSenderTask( void *pvParameters )
{
    int32_t lReceivedValue;
    BaseType_t xStatus;
    const TickType_t xTicksToWait = pdMS_TO_TICKS( 500UL );

    for(;;)
    {
        if( uxQueueMessagesWaiting( xQueue ) != 0 )
        {
            printf( "Queue should have been empty!\r\n" );
        }

        xStatus = xQueueReceive( xQueue, &lReceivedValue, xTicksToWait );
        if( xStatus == pdPASS )
        {
            /* Data was successfully received  */
            printf( "Received = %d\r\n", lReceivedValue );
        }
        else
        {
            printf( "Could not receive from the queue.\r\n" );
        }
    }
}
```

Example06 – Queue create, write, read

```
static void vSenderTask( void *pvParameters )
{
	int32_t lValueToSend;
	BaseType_t xStatus;

	lValueToSend = ( int32_t ) pvParameters;

	for(;;)
	{
	   xStatus = xQueueSendToBack( xQueue, &lValueToSend, 0 );
	   if(xStatus != pdPASS )
	   {
		printf("Could not send to the queue.\r\n");
	   }
	   else
	   {
		printf("Success send to the queue. \n");
	   }

	   vTaskDelay(300 / portTICK_PERIOD_MS);
	}
}
```

FreeRTOS - Queue

Queue사용
FreeRTOS는 일반적으로 하나이상의 소스로 부터 데이터를 수신하도록 설계된다.
따라서 수신 task는 전달된 데이터가 어디서 온것인지 어떻게 처리해야 하는지 알아야한다.
queue를 하나 만들어서 전달되는 데이터에 ID와 data를 전달

Example07 – Queue 구조체

```
#include <stdio.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"

static void vSenderTask( void *pvParameters );

typedef enum
{
    eSender1,
    eSender2
} DataSource_t;

/* Define the structure type that will be passed on the queue. */
typedef struct
{
    uint8_t ucValue;
    DataSource_t eDataSource;
} Data_t;
```

```
/* Declare two variables of type Data_t that will be passed on the queue. */
static const Data_t xStructsToSend[ 2 ] =
{
    { 100, eSender1 }, /* Used by Sender1. */
    { 200, eSender2 }  /* Used by Sender2. */
};

/* handle of Task. */
QueueHandle_t xQueue;

static void vSenderTask( void *pvParameters )
{
        BaseType_t xStatus;
        const TickType_t xTicksToWait = pdMS_TO_TICKS( 300UL );

        for(;;)
        {
            xStatus = xQueueSendToBack( xQueue, pvParameters, xTicksToWait );

            if(xStatus != pdPASS )
            {
                printf("Could not send to the queue.\r\n");
            }
        }
}
```

Example07 – Queue 구조체

```
void app_main(void)
{
    printf("Hello world!\n");

    /* Create Queue */
    xQueue = xQueueCreate( 3, sizeof( Data_t ) );

    if(xQueue != NULL)
    {
        /* Create the thread(s) */
        xTaskCreate( vSenderTask, "Sender1", 1024, ( void * )
&xStructsToSend[0], 2, NULL );
        xTaskCreate( vSenderTask, "Sender2", 1024, ( void * )
&xStructsToSend[1], 2, NULL );
    }
    else
    {
        printf("Create Queue failed\r\n");
    }


    BaseType_t xStatus;
    Data_t xReceivedStructure;
```

```
for(;;)
  {
        if( uxQueueMessagesWaiting( xQueue ) != 3 )
        {
            printf( "Queue should have been full!\r\n" );
        }

        xStatus = xQueueReceive( xQueue, &xReceivedStructure, 0 );

        if( xStatus == pdPASS )
        {
            /* Data was successfully received  */
            if( xReceivedStructure.eDataSource == eSender1 )
            {
                printf( "From Sender 1 = %d\r\n", xReceivedStructure.ucValue );
            }
            else
            {
                printf( "From Sender 2 = %d\r\n", xReceivedStructure.ucValue );
            }
        }
        else
        {
            printf( "Could not receive from the queue.\r\n" );
        }
  }
}
```

FreeRTOS - Queue

Queue Set

하나이상의 queue로 부터 task가 데이터를 수신
효율이 좋은 편은 아니기 때문에 설계제약 조건상 절대적으로 필요한 경우에만 사용

xQueueCreateSet( ) : Queue set 생성

QueueSetHandle_t xQueueCreateSet(const UBaseType_t uxEventQueueLength);	
uxEventQueueLength	queue set이 보유할 수 있는 queue handle의 최대수 queue set에는 semaphore도 포함할수 있다 .
return value	NULL : 실패 non-NULL: 성공, queue set handle 반환

FreeRTOS - Queue

Queue Set

xQueueAddToSet( ) : queue set에 queue나 semaphore를 추가

BaseType_t xQueueAddToSet(QueueSetMemberHandle_t xQueueOrSemaphore, QueueSetHandle_t xQueueSet);	
xQueueOrSemaphore	‘queue set’에 더할 queue 또는 semaphore의 handle
xQueueSet	‘queue set’의 handle
return value	pdPASS : 성공 pdFAIL : 실패

xQueueSelectFromSet( ) : queue set으로 부터 queue handle을 읽음

QueueSetMemberHandle_t xQueueSelectFromSet(QueueSetHandle_t xQueueSet, const TickType_t xTicksToWait);	
xQueueSet	‘queue set’의 handle
xTicksToWait	task가 Blocked state에 있을 시간의 최대값 0이면 바로 리턴 portMAX_DELAY은 무한대기
return value	NULL : 실패 non-NULL: 성공, 데이터를 포함하는 queue 또는 semaphore의 handle (casting)

Example08 – Queue Set

```
#include <stdio.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"

static void vSenderTask1( void *pvParameters );
static void vSenderTask2( void *pvParameters );

/* handle of queue. */
static QueueHandle_t xQueue1 = NULL, xQueue2 = NULL;
/* handle of queue set */
static QueueSetHandle_t xQueueSet = NULL;
```

```
static void vSenderTask1( void *pvParameters )
{
    const TickType_t xBlockTime = pdMS_TO_TICKS( 200 );
    const char * const pcMessage = "Message from vSenderTask1\r\n";

    for( ;; )
    {
        /* Block for 100ms. */
        vTaskDelay( xBlockTime );

        /* Send this task's string to xQueue1. */
        xQueueSend( xQueue1, &pcMessage, 0 );
    }
}
```

Example08 – Queue Set

```
static void vSenderTask2( void *pvParameters )
{
   const TickType_t xBlockTime = pdMS_TO_TICKS( 400 );
   const char * const pcMessage = "Message from
vSenderTask2\r\n";

   for( ;; )
   {
         /* Block for 200ms. */
         vTaskDelay( xBlockTime );

         /* Send this task's string to xQueue2. */
         xQueueSend( xQueue2, &pcMessage, 0 );
   }
}
```

```
void app_main(void)
{
   printf("Hello world!\n");

   /* Create Queue */
   xQueue1 = xQueueCreate( 1, sizeof( char * ) );
   xQueue2 = xQueueCreate( 1, sizeof( char * ) );

   /* create Queue-set */
   xQueueSet = xQueueCreateSet( 2 );

   /* Add the two queues to the set. */
   xQueueAddToSet( xQueue1, xQueueSet );
   xQueueAddToSet( xQueue2, xQueueSet );

   /* Create the tasks that send to the queues. */
   xTaskCreate( vSenderTask1, "Sender1", 1024, NULL, 1, NULL );
   xTaskCreate( vSenderTask2, "Sender2", 1024, NULL, 1, NULL );

   QueueHandle_t xQueueThatContainsData;
   char *pcReceivedString;
```

Example08 – Queue Set

```
 for( ;; )
    {
            /* Block on the queue set to wait for one of the queues in the set to contain data. */
            xQueueThatContainsData = ( QueueHandle_t ) xQueueSelectFromSet( xQueueSet, portMAX_DELAY );

            xQueueReceive( xQueueThatContainsData, &pcReceivedString, 0 );
            printf( pcReceivedString );
    }
}
```

FreeRTOS - Queue

xQueueOverwrite( ) : queue안의 저장된 데이터를 변경

BaseType_t xQueueOverwrite(QueueHandle_t xQueue, const void * pvItemToQueue);	
xQueue	queue handle
pvItemToQueue	변경할 데이터
return value	pdPASS만 리턴

xQueuePeek( ) : queue의 head에서 데이터를 읽지만 읽은 데이터를 지우지 않는다.

BaseType_t xQueuePeek(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
return value	pdPASS : write 성공 errQUEUE_EMPTY : read 실패 (queue empty or xTicksToWait expired)

Example09

```
#include <stdio.h>
#include <string.h>

#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"

static void vUpdateData( void *pvParameters );

typedef struct xExampleStructure
{
    TickType_t xTimeStamp;
    uint32_t ulValue;
} Example_t;

/* handle of queue. */
static QueueHandle_t xMailbox;
```

```
static void vUpdateData( void *pvParameters )
{
    Example_t xData;
    uint32_t value = 0;
    const TickType_t xBlockTime = pdMS_TO_TICKS( 500 );

    for(;;)
    {
        xData.ulValue = value++;
        xData.xTimeStamp = xTaskGetTickCount();

        xQueueOverwrite(xMailbox, &xData);

        vTaskDelay( xBlockTime );
    }
}
```

Example09

```
void app_main(void)
{
    printf("Hello world!\n");

    /* Create Queue */
    xMailbox = xQueueCreate( 1, sizeof( Example_t ) );

    /* Create the tasks that send to the queues. */
    xTaskCreate( vUpdateData, "Update", 1024, NULL, 1, NULL );

    Example_t xData;
    TickType_t xPrevTimeStamp;
    const TickType_t xBlockTime = pdMS_TO_TICKS( 100 );

    memset(&xData, 0, sizeof(Example_t));

    for(;;)
    {
        xPrevTimeStamp = xData.xTimeStamp;
        xQueuePeek(xMailbox, &xData, portMAX_DELAY);

        if(xData.xTimeStamp > xPrevTimeStamp)
        {
            printf("value: %d\r\n", xData.ulValue);
        }
        else
        {
            printf("Not receive Mailbox\r\n");
        }

        vTaskDelay( xBlockTime );
    }
}
```

FreeRTOS - Interrupt

이벤트 처리 구현 전략

1.event 검출을 어떻게 할것인가? (Interrupt or polling ?)
2.interrupt가 사용될때, ISR내의 처리와 ISR외부의 처리량을 어떻게 할것인가?
3.event가 main code와 어떻게 통신 하는지, 비동기 처리 할 수 있도록이 코드를 어떻게 구성 할 것인가?

Interrupt 처리

Interrupt의 처리시간은 짧게 해야 한다.
Interrupt처리에 의해서 수행이 지연된 task의 우선순위가 다른 task의 우선순위보다 높으면 ISR처리후 바로 수행된다.

Interrupt Safe API

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

Example10 - Interrupt

```
#include <stdio.h>
#include <string.h>

#include "sdkconfig.h"
#include "driver/gpio.h"

#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"

#define GPIO_INPUT_USER_BUTTON          0
#define GPIO_INPUT_BUTTON_SEL
(1ULL<<GPIO_INPUT_USER_BUTTON)
#define ESP_INTR_FLAG_DEFAULT           0

static xQueueHandle gpio_evt_queue = NULL;

static void IRAM_ATTR gpio_isr_handler(void* arg)
{
    uint32_t gpio_num = (uint32_t) arg;
    xQueueSendFromISR(gpio_evt_queue, &gpio_num, NULL);
}
```

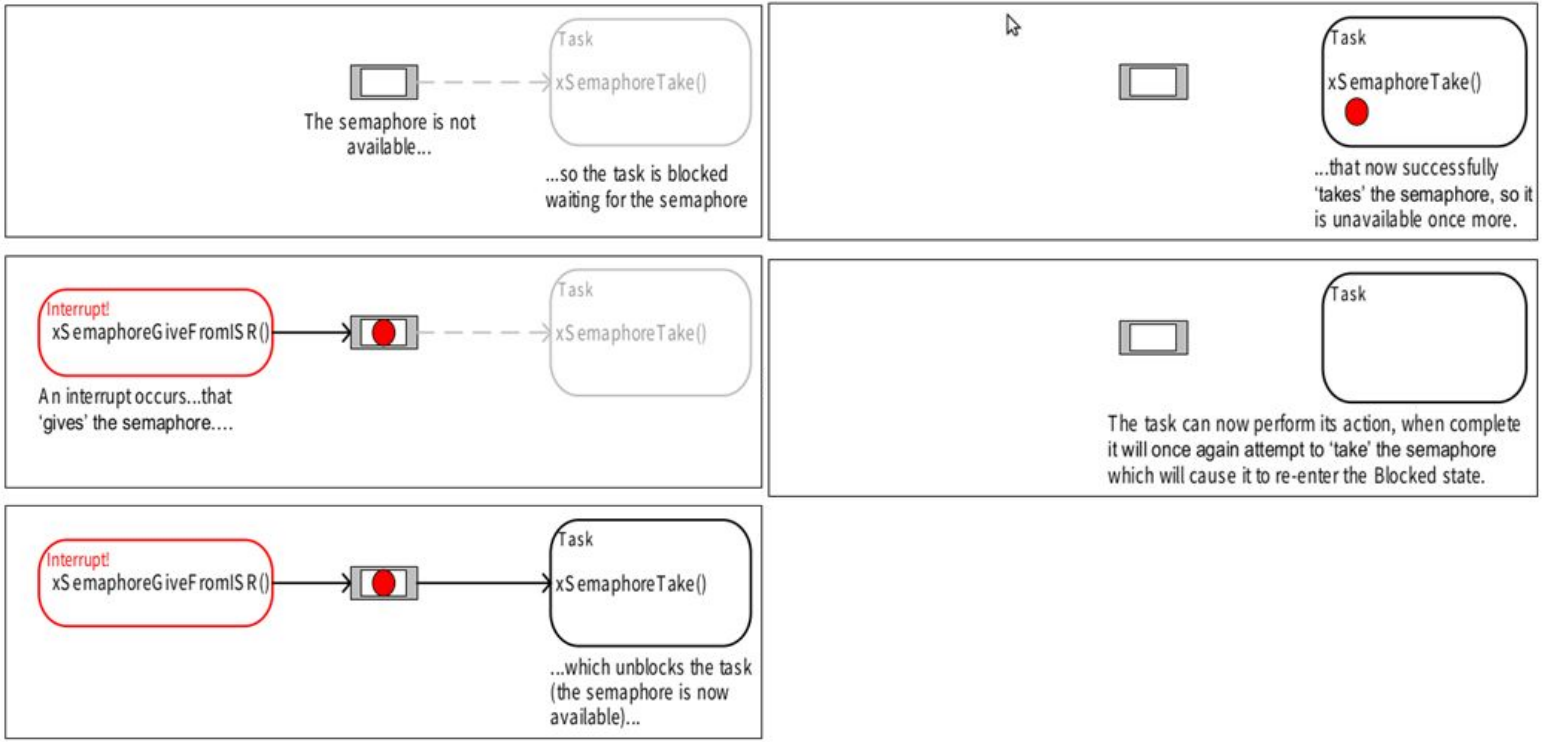
```
 void app_main(void)
{
    gpio_config_t io_conf;

    //interrupt of rising edge
    io_conf.intr_type = GPIO_PIN_INTR_POSEDGE;
    //bit mask of the pins
    io_conf.pin_bit_mask = GPIO_INPUT_BUTTON_SEL;
    //set as input mode
    io_conf.mode = GPIO_MODE_INPUT;
    //enable pull-up mode
    io_conf.pull_up_en = 1;
    gpio_config(&io_conf);
```

Example10 - Interrupt

```
    //install gpio isr service
    gpio_install_isr_service(ESP_INTR_FLAG_DEFAULT);
    //hook isr handler for specific gpio pin
    gpio_isr_handler_add(GPIO_INPUT_USER_BUTTON, gpio_isr_handler, (void*) GPIO_INPUT_USER_BUTTON);

    //create a queue to handle gpio event from isr
    gpio_evt_queue = xQueueCreate(10, sizeof(uint32_t));

    uint32_t io_num;
    while(1)
    {
            if(xQueueReceive(gpio_evt_queue, &io_num, portMAX_DELAY))
            {
                printf("GPIO[%d] intr, val: %d\n", io_num, gpio_get_level(io_num));
            }
    }
}
```

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
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

Example11 – task와 interrupt 동기화

```
#include <stdio.h>
#include <string.h>

#include "sdkconfig.h"
#include "driver/gpio.h"

#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/semphr.h"

#define GPIO_INPUT_USER_BUTTON          0
#define GPIO_INPUT_BUTTON_SEL
(1ULL<<GPIO_INPUT_USER_BUTTON)
#define ESP_INTR_FLAG_DEFAULT           0

/* Declare a variable of type SemaphoreHandle_t. */
SemaphoreHandle_t xBinarySemaphore;

static void IRAM_ATTR gpio_isr_handler(void* arg)
{
    /* 'Give' the semaphore to unblock the task. */
    xSemaphoreGiveFromISR( xBinarySemaphore, NULL );
```

```
 void app_main(void)
{
    gpio_config_t io_conf;

    //interrupt of rising edge
    io_conf.intr_type = GPIO_PIN_INTR_POSEDGE;
    //bit mask of the pins, use GPIO4/5 here
    io_conf.pin_bit_mask = GPIO_INPUT_BUTTON_SEL;
    //set as input mode
    io_conf.mode = GPIO_MODE_INPUT;
    //enable pull-up mode
    io_conf.pull_up_en = 1;
    gpio_config(&io_conf);
```

Example11 – task와 interrupt 동기화

```
    //install gpio isr service
    gpio_install_isr_service(ESP_INTR_FLAG_DEFAULT);
    //hook isr handler for specific gpio pin
    gpio_isr_handler_add(GPIO_INPUT_USER_BUTTON, gpio_isr_handler, (void*) GPIO_INPUT_USER_BUTTON);

    /* create binary semaphore */
    xBinarySemaphore = xSemaphoreCreateBinary();

    while(1)
    {
            xSemaphoreTake( xBinarySemaphore, portMAX_DELAY );
            printf( "Handler task - Processing event.\r\n" );
    }
}
```

FreeRTOS - Semaphore

Counting Semaphore

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

일반적으로 다음의 두경우에 사용

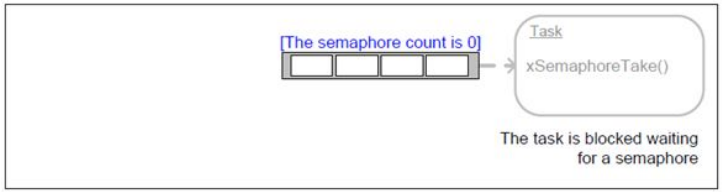
1.Counting event – event count에 사용(초기화 count value는 0)

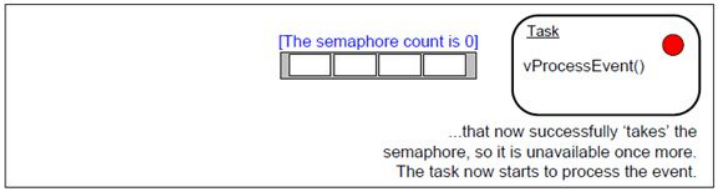
2.Resource management - 사용가능한 resource의 수를 표시 (초기화 count value는 resource의 수)

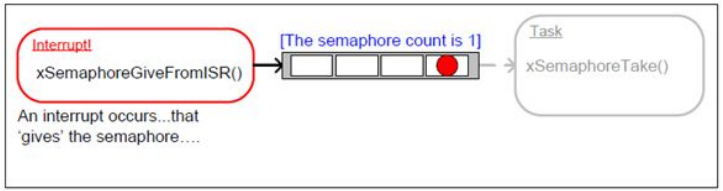
FreeRTOS - Semaphore

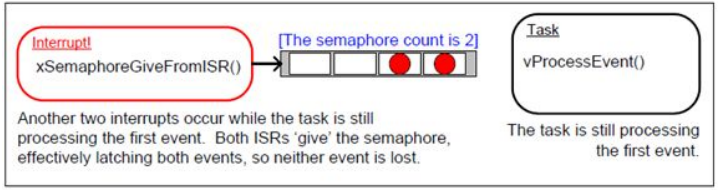
Counting Semaphore

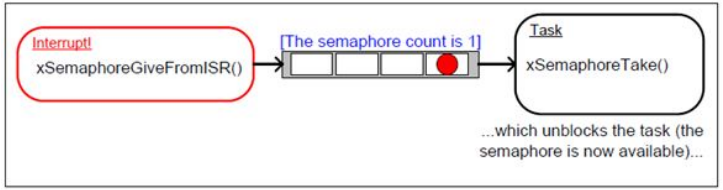
Counting Event 예

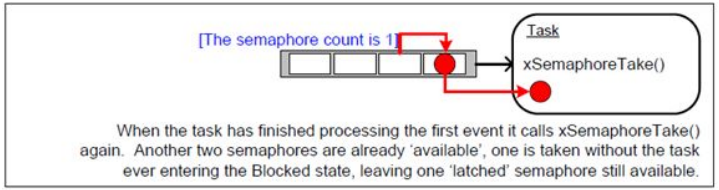












FreeRTOS - Semaphore

Counting Semaphore

xSemaphoreCreateCounting() : counting semaphore생성

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

Example12 – counting semaphore를 이용한 task와 interrupt 동기화

```
#include <stdio.h>
#include <string.h>

#include "sdkconfig.h"
#include "driver/gpio.h"

#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/semphr.h"

#define GPIO_INPUT_USER_BUTTON          0
#define GPIO_INPUT_BUTTON_SEL
(1ULL<<GPIO_INPUT_USER_BUTTON)
#define ESP_INTR_FLAG_DEFAULT           0

/* Declare a variable of type SemaphoreHandle_t. */
SemaphoreHandle_t xCountingSemaphore;

static void IRAM_ATTR gpio_isr_handler(void* arg)
{
    /* 'Give' the semaphore to unblock the task. */
    xSemaphoreGiveFromISR( xCountingSemaphore, NULL );
    xSemaphoreGiveFromISR( xCountingSemaphore, NULL );
    xSemaphoreGiveFromISR( xCountingSemaphore, NULL );
}
```

```
void app_main(void)
{
    gpio_config_t io_conf;

    //interrupt of rising edge
    io_conf.intr_type = GPIO_PIN_INTR_POSEDGE;
    //bit mask of the pins, use GPIO4/5 here
    io_conf.pin_bit_mask = GPIO_INPUT_BUTTON_SEL;
    //set as input mode
    io_conf.mode = GPIO_MODE_INPUT;
    //enable pull-up mode
    io_conf.pull_up_en = 1;
    gpio_config(&io_conf);
```

Example12 – counting semaphore를 이용한 task와 interrupt 동기화

```
    //install gpio isr service
    gpio_install_isr_service(ESP_INTR_FLAG_DEFAULT);
    //hook isr handler for specific gpio pin
    gpio_isr_handler_add(GPIO_INPUT_USER_BUTTON, gpio_isr_handler, (void*) GPIO_INPUT_USER_BUTTON);

    /* create binary semaphore */
    xCountingSemaphore = xSemaphoreCreateCounting( 10, 0 );

    uint32_t count=0;
    while(1)
    {
            xSemaphoreTake( xCountingSemaphore, portMAX_DELAY );
            count++;
            printf( "Handler task - Processing event count: %d\r\n", count );
    }
}
```

FreeRTOS - Timer

Software Timer

미래에 설정된 시간 또는 주기적으로 기능 실행을 예약하는 데 사용
FreeRTOS 커널에의해서 실행 (not use hardware timer)
Software Timer는 콜백(callback)함수를 호출

FreeRTOS의 옵션으로 사용 설정은
FreeRTOS/Source/timers.c 를 포함

Software Timer Callback 함수

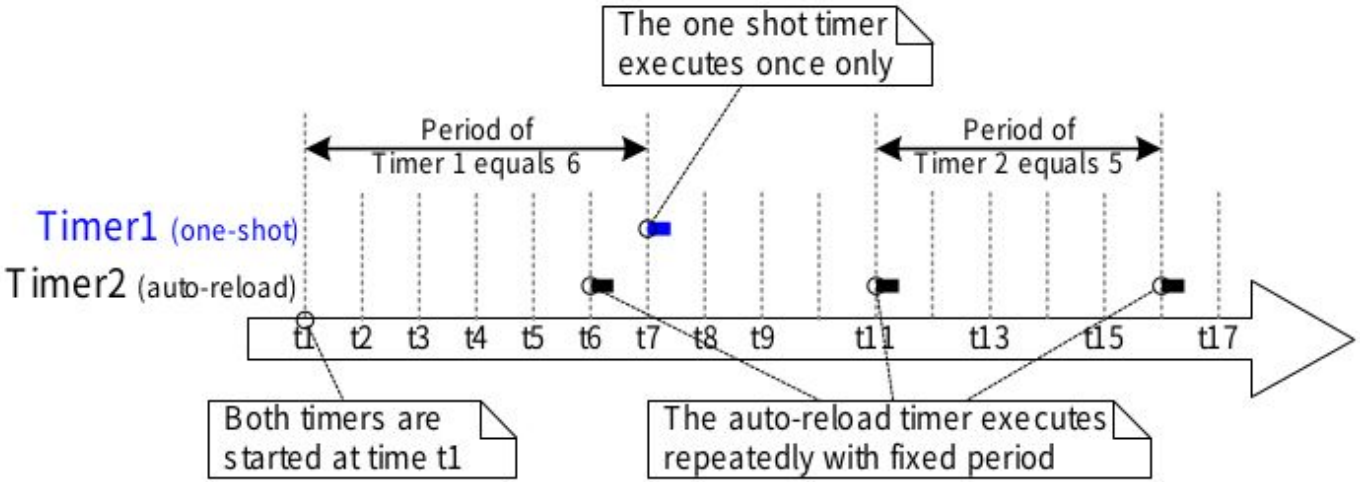
```
void ATimerCallback( TimerHandle_t xTimer );
```

FreeRTOS - Timer

Software Timer

One-shot, Auto-reload Timers

- one-shot timer : 콜백함수를 한번만 실행
- auto-reload timer: 타이머가 실행되면, 주어진 주기내에 반복적으로 실행



소프트웨어 타이머 상태

- Dormant(수면)
- Running(실행)

FreeRTOS - Timer

RTOS Daemon Task

모든 Software timer callback은 RTOS daemon task의 context에서 실행
daemon task는 스케줄러가 시작되면 자동으로 생성

Timer Command Queue

Software timer 함수를 호출한 task로 부터 daemon task로 timer command queue에 명령을 보냄
(Start timer, stop timer, reset timer 등)
timer command queue는 스케줄러가 시작될때 자동으로 생성

FreeRTOS - Timer

xTimerCreate( ) : Software timer 생성

TimerHandle_t xTimerCreate(const char * const pcTimerName, TickType_t xTimerPeriodInTicks, UBaseType_t uxAutoReload, void * pvTimerID, TimerCallbackFunction_t pxCallbackFunction);	
pcTimerName	타이머의 이름
xTimerPeriodInTicks	tick 단위로 지정된 타이머의 기간, pdMS_TO_TICKS()로 ms 사용가능
uxAutoReload	pdTRUE : auto-reload timer pdFALSE : one-shot timer
pvTimerID	각 소프트웨어 타이머의 ID 값 이 ID는 하나이상의 소프트웨어 타이머에서 같은 콜백을 사용하는 경우 유용하게 사용된다.
pxCallbackFunction	소프트웨어 타이머의 콜백함수
return value	NULL : 실패 non-NULL : 소프트웨어 타이머의 handle

FreeRTOS - Timer

xTimerStart( ) : Software timer 시작

BaseType_t xTimerStart(TimerHandle_t xTimer, TickType_t xTicksToWait);	
xTimer	소프트웨어 타이머의 handle
xTicksToWait	xTimerStart()함수는 'start a timer' 명령을 command queue를 사용해서 daemon task로 전달한다. 따라서 command queue가 full이면 block state가 된다. task가 Blocked state에 있을 시간의 최대값 0이면 바로 리턴 portMAX_DELAY은 무한대기
return value	pdPASS : 성공 pdFALSE : 실패(시간초과)

xTimerStop( ) : Running state안의 software timer 중지

BaseType_t xTimerStop(TimerHandle_t xTimer, TickType_t xTicksToWait);	
xTimer	소프트웨어 타이머의 handle
xTicksToWait	task가 Blocked state에 있을 시간의 최대값 0이면 바로 리턴 portMAX_DELAY은 무한대기
return value	pdPASS : 성공 pdFALSE : 실패(시간초과)

Example13 – one-shot, auto-reload timer

```
#include <stdio.h>
#include <string.h>

#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/timers.h"

#define mainONE_SHOT_TIMER_PERIOD        ( pdMS_TO_TICKS( 3333UL ) )
#define mainAUTO_RELOAD_TIMER_PERIOD ( pdMS_TO_TICKS( 500UL ) )

/* The Software timer callback functions. */
static void prvOneShotTimerCallback( TimerHandle_t xTimer );
static void prvAutoReloadTimerCallback( TimerHandle_t xTimer );
```

```
static void prvOneShotTimerCallback( TimerHandle_t xTimer )
{
        static TickType_t xTimeNow;

        /* Obtain the current tick count. */
        xTimeNow = xTaskGetTickCount();

        printf( "One-shot timer callback executing %d\r\n", xTimeNow );
}

static void prvAutoReloadTimerCallback( TimerHandle_t xTimer )
{
        static TickType_t xTimeNow;

        /* Obtain the current tick count. */
        xTimeNow = xTaskGetTickCount();

        printf( "Auto-reload timer callback executing %d\r\n", xTimeNow );
}
```

Example13 – one-shot, auto-reload timer

```
void app_main(void)
{
    TimerHandle_t xAutoReloadTimer, xOneShotTimer;
    BaseType_t xTimer1Started, xTimer2Started;

    /* Create the one shot software timer */
    xOneShotTimer = xTimerCreate( "OneShot",
mainONE_SHOT_TIMER_PERIOD, pdFALSE, 0,
prvOneShotTimerCallback );

    /* Create the auto-reload software timer */
    xAutoReloadTimer = xTimerCreate( "AutoReload",
mainAUTO_RELOAD_TIMER_PERIOD, pdTRUE, 0,
prvAutoReloadTimerCallback );

    /* Check the timers were created. */
    if( ( xOneShotTimer != NULL ) && ( xAutoReloadTimer != NULL ) )
    {
        /* Start the software timers */
        xTimer1Started = xTimerStart( xOneShotTimer, 0 );
        xTimer2Started = xTimerStart( xAutoReloadTimer, 0 );

        if( ( xTimer1Started == pdPASS ) && ( xTimer2Started == pdPASS ) )
        {
            printf("Start Software timer\r\n");
        }
    }

    while(1)
    {
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

FreeRTOS - Timer

Timer ID

각 소프트웨어 타이머는 ID를 갖는다.
ID 값은 void pointer값으로 사용자가 어떤 용도로도 활용할수 있다.
초기값은 타이머가 생성될때 할당

vTimerSetTimerID( ) : ID 값 변경

void vTimerSetTimerID(const TimerHandle_t xTimer, void *pvNewID);	
xTimer	소프트웨어 타이머 handle
pvNewID	변경할 timer id의 값

pvTimerGetTimerID( ) : ID 값 쿼리

void *pvTimerGetTimerID(TimerHandle_t xTimer);	
xTimer	소프트웨어 타이머 handle
return value	timer id의 값

Example14 – software timer ID

```
#include <stdio.h>
#include <string.h>

#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/timers.h"

#define mainONE_SHOT_TIMER_PERIOD        ( pdMS_TO_TICKS( 3333UL ) )
#define mainAUTO_RELOAD_TIMER_PERIOD ( pdMS_TO_TICKS( 500UL ) )

/* The Software timer callback functions. */
static void prvTimerCallback( TimerHandle_t xTimer );

TimerHandle_t xAutoReloadTimer, xOneShotTimer;


static void prvTimerCallback( TimerHandle_t xTimer )
{
        TickType_t xTimeNow;
        uint32_t ulExecutionCount =0;

        /* Obtain the ID */
        ulExecutionCount = ( uint32_t ) pvTimerGetTimerID( xTimer );
        ulExecutionCount++;
        vTimerSetTimerID( xTimer, ( void * ) ulExecutionCount );

        /* Obtain the current tick count. */
        xTimeNow = xTaskGetTickCount();

        if( xTimer == xOneShotTimer )
        {
                printf( "One-shot timer callback executing %d\r\n", xTimeNow );
        }
        else
        {
                printf( "Auto-reload timer callback executing %d\r\n", xTimeNow );

                if( ulExecutionCount == 5 )
                {
                        xTimerStop( xTimer, 0 );
                }
        }
}
```

Example14 – software timer ID

```
void app_main(void)
{
	BaseType_t xTimer1Started, xTimer2Started;
	/* Create the one shot software timer */
	xOneShotTimer = xTimerCreate( "OneShot", mainONE_SHOT_TIMER_PERIOD, pdFALSE, 0,  prvTimerCallback);

	/* Create the auto-reload software timer */
	xAutoReloadTimer = xTimerCreate( "AutoReload", mainAUTO_RELOAD_TIMER_PERIOD, pdTRUE, 0,  prvTimerCallback);
	/* Check the timers were created. */
	if( ( xOneShotTimer != NULL ) && ( xAutoReloadTimer != NULL ) )
	{
		/* Start the software timers */
		xTimer1Started = xTimerStart( xOneShotTimer, 0 );
		xTimer2Started = xTimerStart( xAutoReloadTimer, 0 );

		if( ( xTimer1Started == pdPASS ) && ( xTimer2Started == pdPASS ) )
		{
			printf("Start Software timer\r\n");
		}
	}
   while(1)
   {
      vTaskDelay(1000 / portTICK_PERIOD_MS);
   }
}
```

FreeRTOS - Software Timer Management

xTimerChangePeriod( ) : software timer 간격(period)을 변경

BaseType_t xTimerChangePeriod(TimerHandle_t xTimer, TickType_t xNewTimerPeriodInTicks, TickType_t xTicksToWait);	
xTimer	소프트웨어 타이머 handle
xNewTimerPeriodInTicks	tick단위의 새로운 소프트웨어 타이머의 간격, pdMS_TO_TICKS() // ms
xTicksToWait	task가 Blocked state에 있을 시간의 최대값 0이면 바로 리턴 portMAX_DELAY은 무한대기
return value	pdPASS : 성공 pdFALSE : 실패(시간초과)

Example15 – software timer period 변경

```
#include <stdio.h>
#include <string.h>
#include "sdkconfig.h"
#include "driver/gpio.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/timers.h"


#define GPIO_INPUT_USER_BUTTON          0
#define GPIO_INPUT_BUTTON_SEL
(1ULL<<GPIO_INPUT_USER_BUTTON)
#define ESP_INTR_FLAG_DEFAULT           0

#define HEALTHY_PERIOD      ( pdMS_TO_TICKS( 2000UL ) )
#define ERROR_PERIOD        ( pdMS_TO_TICKS( 300UL ) )

/* The Software timer callback functions. */
static void prvTimerCallback( TimerHandle_t xTimer );

uint8_t count_i;
TimerHandle_t xAutoReloadTimer;
```

```
static void IRAM_ATTR gpio_isr_handler(void* arg)
{
        count_i = (count_i+1)%2;
}

static void prvTimerCallback( TimerHandle_t xTimer )
{
        if(count_i)
        {
                xTimerChangePeriod(xAutoReloadTimer, ERROR_PERIOD, 0);
        }
        else
        {
                xTimerChangePeriod(xAutoReloadTimer, HEALTHY_PERIOD, 0);
        }

        printf("Timer Call \n");
}
```

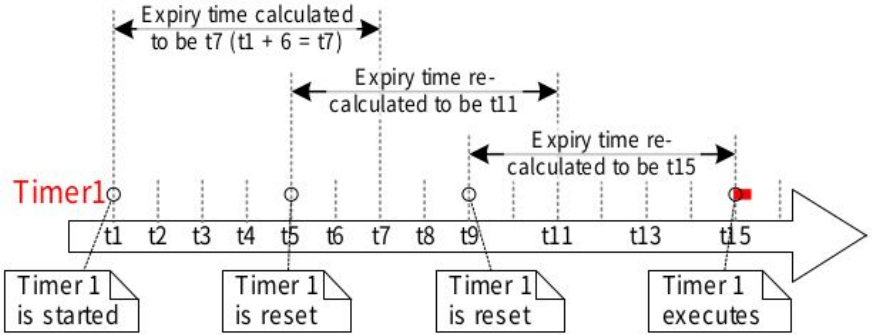
Example15 – software timer period 변경

```
void app_main(void)
{
    gpio_config_t io_conf;

    //interrupt of rising edge
    io_conf.intr_type = GPIO_PIN_INTR_POSEDGE;
    //bit mask of the pins, use GPIO4/5 here
    io_conf.pin_bit_mask = GPIO_INPUT_BUTTON_SEL;
    //set as input mode
    io_conf.mode = GPIO_MODE_INPUT;
    //enable pull-up mode
    io_conf.pull_up_en = 1;
    gpio_config(&io_conf);

    //install gpio isr service
    gpio_install_isr_service(ESP_INTR_FLAG_DEFAULT);
    //hook isr handler for specific gpio pin
    gpio_isr_handler_add(GPIO_INPUT_USER_BUTTON,
gpio_isr_handler, (void*) GPIO_INPUT_USER_BUTTON);
    BaseType_t xTimerStarted;

    /* Create the auto-reload software timer */
    xAutoReloadTimer = xTimerCreate( "AutoReload", HEALTHY_PERIOD, pdTRUE, 0,
prvTimerCallback);
    /* Check the timers were created. */
    if( xAutoReloadTimer != NULL )
    {
            /* Start the software timers */
            xTimerStarted = xTimerStart( xAutoReloadTimer, 0 );
            if(  xTimerStarted == pdPASS )
            {
                printf("Start Software timer\r\n");
            }
    }

    while(1)
    {
       vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

FreeRTOS - Software Timer Management

xTimerReset( ) : Softwrae timer reset

BaseType_t xTimerReset(TimerHandle_t xTimer, TickType_t xTicksToWait);	
xTimer	소프트웨어 타이머 handle
xTicksToWait	task가 Blocked state에 있을 시간의 최대값 0이면 바로 리턴 portMAX_DELAY은 무한대기
return value	pdPASS : 성공 pdFALSE : 실패(시간초과)



Example16 – software timer reset

```
#include <stdio.h>
#include <string.h>
#include "sdkconfig.h"
#include "driver/gpio.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/timers.h"

#define GPIO_INPUT_USER_BUTTON          0
#define GPIO_INPUT_BUTTON_SEL
(1ULL<<GPIO_INPUT_USER_BUTTON)
#define ESP_INTR_FLAG_DEFAULT           0
#define mainBACKLIGHT_TIMER_PERIOD      (
pdMS_TO_TICKS( 5000UL ) )

/* The Software timer callback functions. */
static void prvBacklightTimerCallback( TimerHandle_t xTimer );

uint8_t count_i;
static BaseType_t xSimulatedBacklightOn = pdFALSE;
static TimerHandle_t xBacklightTimer = NULL;
```

```
static void IRAM_ATTR gpio_isr_handler(void* arg)
{
    count_i = (count_i+1)%2;
}

static void prvBacklightTimerCallback( TimerHandle_t xTimer )
{
    TickType_t xTimeNow = xTaskGetTickCount();

    /* The backlight timer expired, turn the backlight off. */
    xSimulatedBacklightOn = pdFALSE;
    /* Print the time at which the backlight was turned off. */
    printf( "Timer expired, turning backlight OFF at time %d\r\n", xTimeNow );
}
```

Example16 – software timer reset

```
void app_main(void)
{
    gpio_config_t io_conf;
    //interrupt of rising edge
    io_conf.intr_type = GPIO_PIN_INTR_POSEDGE;
    //bit mask of the pins, use GPIO4/5 here
    io_conf.pin_bit_mask = GPIO_INPUT_BUTTON_SEL;
    //set as input mode
    io_conf.mode = GPIO_MODE_INPUT;
    //enable pull-up mode
    io_conf.pull_up_en = 1;
    gpio_config(&io_conf);

    //install gpio isr service
    gpio_install_isr_service(ESP_INTR_FLAG_DEFAULT);
    //hook isr handler for specific gpio pin
    gpio_isr_handler_add(GPIO_INPUT_USER_BUTTON, gpio_isr_handler, (void*) GPIO_INPUT_USER_BUTTON);
    xSimulatedBacklightOn = pdFALSE;
    xBacklightTimer = xTimerCreate( "Backlight", mainBACKLIGHT_TIMER_PERIOD, pdFALSE, 0, prvBacklightTimerCallback );
    /* Start the timer. */
    xTimerStart( xBacklightTimer, 0 );

    const TickType_t xShortDelay = pdMS_TO_TICKS( 500 );
    TickType_t xTimeNow;
    printf( "Press a key to turn the backlight on.\r\n" );
```

Example16 – software timer reset

```
    for( ;; )
    {
        /* Has a key been pressed? */
        if(count_i != 0 )
        {
            /* Record the time at which the key press was noted. */
            xTimeNow = xTaskGetTickCount();

            /* A key has been pressed. */
            if( xSimulatedBacklightOn == pdFALSE )
            {
                xSimulatedBacklightOn = pdTRUE;
                printf( "Key pressed, turning backlight ON at time %d\r\n", xTimeNow );
            }
            else
            {
                printf( "Key pressed, resetting software timer at time %d\r\n", xTimeNow );
            }
            xTimerReset( xBacklightTimer, xShortDelay );
        }

        vTaskDelay( xShortDelay );
    }
}
```

FreeRTOS - Event Group

Event Group

event작업을 전달 할수 있는 FreeRTOS의 또 다른 기능
여러 task를 동기화, event를 둘 이상의 task로 broadcasting
한 set의 event가 발생할 때마다 blocked state로 대기하는 작업을 허용하고 여러 작업이 완료 될 때까지 blocked state로 task를 대기 할수 있다.
많은 binary semaphore를 단일 event group로 대체 가능 하기 때문에 Ram의 사용량을 줄일수 있다.

semaphore와 queue와 다른점
-event group을 사용하면 하나 이상의 event 조합을 task가 blocked state로 대기
-event group은 event가 발생할 때 동일한 event 또는 event조합을 기다리고 있던 모든 task를 unblock

FreeRTOS - Event Group

Event Group의 특성

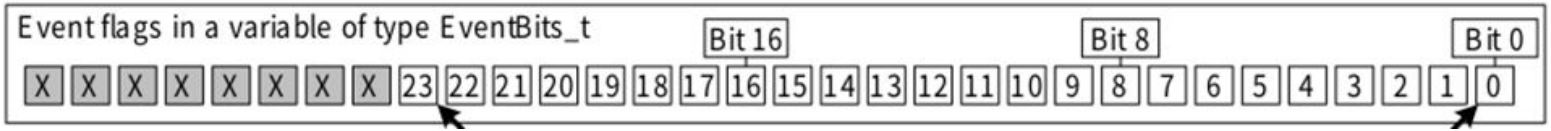
Event Flags : event가 발생유무를 나타내는데 사용되는 Boolean(1,0)값

Event Groups : event flags의 set

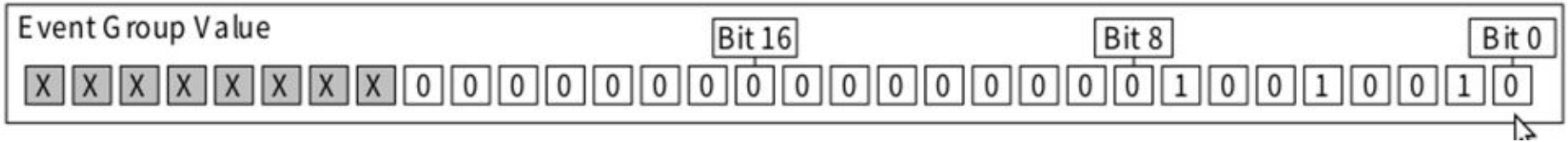
event group안의 event flag의 상태는 “EventBits_t” 타입 변수의 단일 bit로 표시 (event flags는 event ‘bits’라고도 한다.)

EventBits_t : 각 bit는 해당 비트가 나타내는 event의 발생 유무(1,0)를 표시

EventBits_t 타입의 변수 mapping



예) 1,4,7 bit가 set 된 상태 (0x92)



FreeRTOS - Event Group

Event Group를 사용한 Event 관리

xEventGroupCreate() : Event group 생성

EventGroupHandle_t xEventGroupCreate(void);	
return value	NULL: 생성실패 non-NULL: event group생성, EventGroupHandle_t 반환

FreeRTOS - Event Group

xEventGroupSetBits() : event group에 하나 이상의 bit를 설정
설정중인 bit가 나타내는 event가 발생했음을 task에 알리는데 사용

EventBits_t xEventGroupSetBits(EventGroupHandle_t xEventGroup, const EventBits_t uxBitsToSet);	
xEventGroup	event group의 handle
uxBitsToSet	event group에서 1로 설정할 event bit를 지정하는 bit mask event group값은 기존 event group값을 bit OR하여 갱신
return value	event group 값

xEventGroupSetBitsFromISR() : xEventGroupSetBits()의 interrupt safe version

BaseType_t xEventGroupSetBitsFromISR(EventGroupHandle_t xEventGroup, const EventBits_t uxBitsToSet, BaseType_t *pxHigherPriorityTaskWoken);	
xEventGroup	event group의 handle
uxBitsToSet	event group에서 1로 설정할 event bit를 지정하는 bit mask event group값은 기존 event group값을 bit OR하여 갱신
pxHigherPriorityTaskWoken	event bit를 ISR내에서 직접 설정하지 않고 timer command queue에 명령을 전송 daemon task에 대한 작업을 연기 daemon task는 timer command queue가 사용가능할때 까지 Blocked state에 있고, timer command queue에 기록하면 daemon task가 blocked state를 떠나게 된다. daemon task의 우선순위가 현재 interrupt를 실행한 task보다 높다면 pdTRUE가 된다.
return value	pdPASS : 성공 (timer command queue에 'set bits' 전달) pdFALSE: 실패(timer command queue full)

FreeRTOS - Event Group

xEventGroupWaitBits() : event group의 값을 읽는다.
event bit가 아직 설정되어 있지 않은 경우,
event group안의 하나 이상의 event bit가 set되기를 Blocked state안에서 선택적으로 기다린다.

EventBits_t xEventGroupWaitBits(const EventGroupHandle_t xEventGroup, const EventBits_t uxBitsToWaitFor, const BaseType_t xClearOnExit, const BaseType_t xWaitForAllBits, TickType_t xTicksToWait);	
xEventGroup	event group의 handle
uxBitsToWaitFor	event group안의 특정 event bit에 대한 bit mask
xClearOnExit	호출하는 task의 'unblock condtion'이 충족되고 xClearOnExit이 pdTRUE로 설정된 경우 uxBitsToWaitFor로 지정된 event bit는 호출 task가 xEventGroupWaitBits() 함수를 종료하기 전에 event group에서 다시 0으로 지워진다. xClearOnExit가 pdFALSE로 설정된 경우 event group의 event bit상태는 xEventGroupWaitBits() 함수에 의해서 수정되지 않는다.
xWaitForAllBits	함수를 호출하는 task의 상태를 uxBitsToWaitFor값과 비교해서 Blocked state, unblock될지를 결정하는 연산 방법을 정의한다. pdFALSE : uxBitsToWaitFor값과 event group값을 OR 연산 pdTRUE : uxBitsToWaitFor값과 event group값을 AND 연산
xTicksToWait	task가 Blocked state에 있을 시간의 최대값 0이면 바로 리턴 portMAX_DELAY은 무한대기
return value	호출 task가 unblock condition이 충족 되었을 때 리턴된 경우, 호출 task의 unblock condition의 조건이 충족 될 때 event group의 값 xTicksToWait값에 의한 시간 초과로 리턴된 경우, block시간이 만기된 시간의 event group값

FreeRTOS - Event Group

`unblock condition'은 uxBitsToWaitFor 및 xWaitForAllBits의 parameter값을 조합으로 지정:

- uxBitsToWaitFor는 test할 event group안의 event bit를 지정
- xWaitForAllBits는 bit OR 또는 bit AND test 여부를 지정

Existing Event Group Value	uxBitsToWaitFor value	xWaitForAllBits value	Resultant Behavior
0000	0101	pdFALSE	bit0또는 bit2가 event group에 설정되어 있지 않으므로 -> Blocked state bit0 또는 bit2가 설정되면 -> unblock
0100	0101	pdTRUE	bit0 과 bit2가 event group에 설정되어 있지 않으므로 -> Blocked state bit0와 bit2가 모두 event gropu에 설정되면 -> unblock
0100	0110	pdFALSE	xWaitForAllBits가 pdFALSE이고 uxBitsToWaitFor에 지정된 두 bit중 하나가 event group에 이미 설정되어 있기 때문에 호출 task는 Blocked state 상태로 들어 가지 않는다.
0100	0110	pdTRUE	xWaitForAllBits가 pdTRUE이고 uxBitsToWaitFor에 지정된 두 bit중 하나만 event group에 이미 설정되어 있기 때문에 호출 task는 Blocked state. bit2와 bit1둘다 event group에 설정된 경우만 unblocked 된다.

uxBitsToWaitFor를 사용하여 test할 bit를 지정, 호출 task는 'unblock condition'이 충족된 후 이 bit를 0으로 다시 clear해야 한다.
Event bit는 xEventGroupClearBits() 함수를 사용하여 지울수 있다,

Example17 – event group

```
#include <stdio.h>
#include <string.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"

/* Definitions for the event bits in the event group. */
#define mainFIRST_TASK_BIT  ( 1UL << 0UL )#define
mainSECOND_TASK_BIT         ( 1UL << 1UL )

/* task function */
static void vEventBitSettingTask( void *pvParameters );

/* Declare the event group */
EventGroupHandle_t xEventGroup;

/* task function */
static void vEventBitSettingTask( void *pvParameters );

/* Declare the event group */
EventGroupHandle_t xEventGroup;
```

```
static void vEventBitSettingTask( void *pvParameters )
{
   const TickType_t xDelay200ms = pdMS_TO_TICKS( 200UL );

   for( ;; )
   {
         vTaskDelay( xDelay200ms );
         printf( "Bit setting task -\t about to set bit 0.\r\n" );
         /* set event bit 0. */
         xEventGroupSetBits( xEventGroup, mainFIRST_TASK_BIT );

         vTaskDelay( xDelay200ms );
         printf( "Bit setting task -\t about to set bit 1.\r\n" );
         /* set event bit 1. */
         xEventGroupSetBits( xEventGroup, mainSECOND_TASK_BIT );
   }
}
```

Example17 – event group

```
void app_main(void)
{
    /* Create event group */
    xEventGroup = xEventGroupCreate();

    xTaskCreate( vEventBitSettingTask, "BitSetter", 1024, NULL, 1, NULL );

    const EventBits_t xBitsToWaitFor = (mainFIRST_TASK_BIT | mainSECOND_TASK_BIT);
    EventBits_t xEventGroupValue;

    for( ;; )
    {
        /* Block to wait for event bits to become set within the event group. */
        xEventGroupValue = xEventGroupWaitBits(xEventGroup, xBitsToWaitFor, pdTRUE, pdFALSE, portMAX_DELAY );

        /* Print a message for each bit that was set. */
        if( ( xEventGroupValue & mainFIRST_TASK_BIT ) != 0 )
        {
            printf( "Bit reading task -\t event bit 0 was set\r\n" );
        }

        if( ( xEventGroupValue & mainSECOND_TASK_BIT ) != 0 )
        {
            printf( "Bit reading task -\t event bit 1 was set\r\n" );
        }
    }
}
```

ESP-IDF FreeRTOS

SMP(symmetric multiprocessing)지원을 위해 FreeRTOS수정버전

- CPU0(PRO_CPU), CPU1(APP_CPU) 듀얼코어 지원
- FreeRTOS는 단일 코어에서 실행
- FreeRTOS v8.2.0기반에 V9.0.0기능을 백 포트

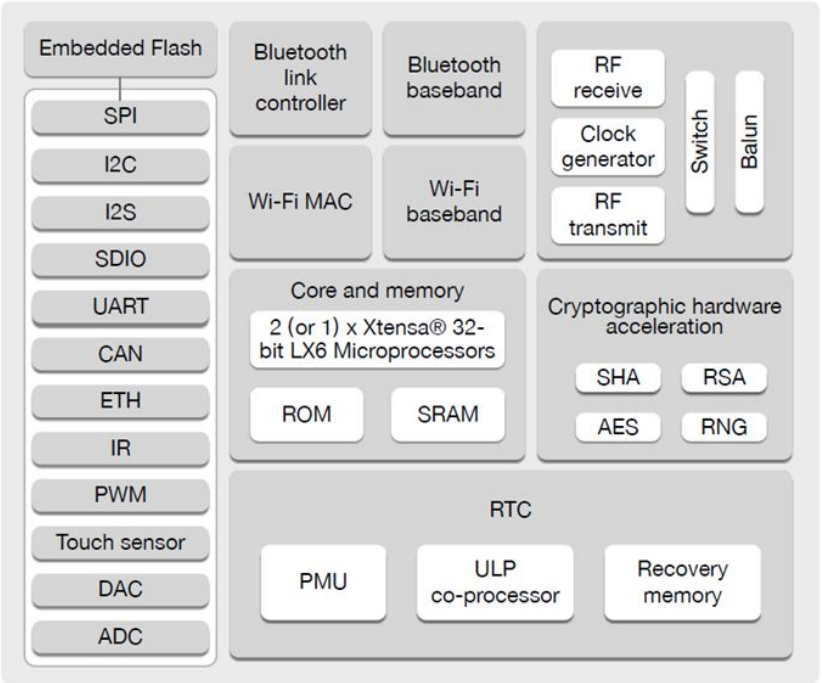


Figure 1: Functional Block Diagram

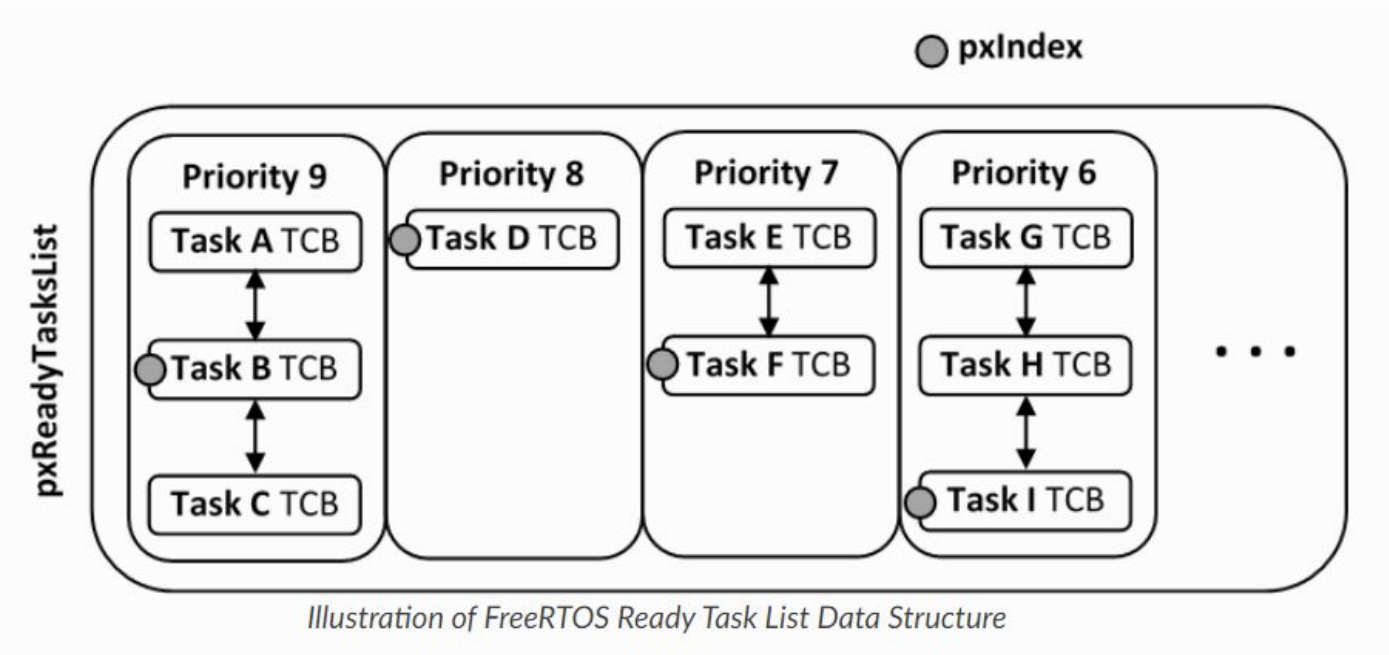
ESP-IDF FreeRTOS

SMP 시스템에서 선택한 코어에 Task 생성

BaseType_t xTaskCreatePinnedToCore(TaskFunction_t pvTaskCode, const char *constpcName, const uint32_t usStackDepth, void *const pvParameters, UBaseType_t uxPriority, TaskHandle_t *const pvCreatedTask, const BaseType_t xCoreID)	
pvTaskCode	Task 함수
pcName	Task 이름 configMAX_TASK_NAME_LEN: 이름의 최대 길이
usStackDepth	Task의 stack크기(bytes)
pvParameters	Task 함수에 전달되는 파라미터
uxPriority	Task 실행 우선순위
pvCreatedTask	Task Handle
xCoreID	Task를 실행할 core의 ID 0, 1 : core id taskNO_AFFINITY: 코어 고정없이 스케줄러가 사용 가능한 모든코어에서 실행
return	pdPASS: 성공 다른 값: 실패(projdefs.h 에 정의)

FreeRTOS - Scheduling

스케줄링 알고리즘
Running state로 전환할 Ready state task를 결정하는 소프트웨어 루틴



pxIndex는 마지막 반환된 TCB를 표시
위 색인을 이용해서 Round Robin 스케줄링 구현

ESP-IDF FreeRTOS - Scheduling

PRO_CPU: 파란색, 보라색 APP_CPU: 주황색, 보라색

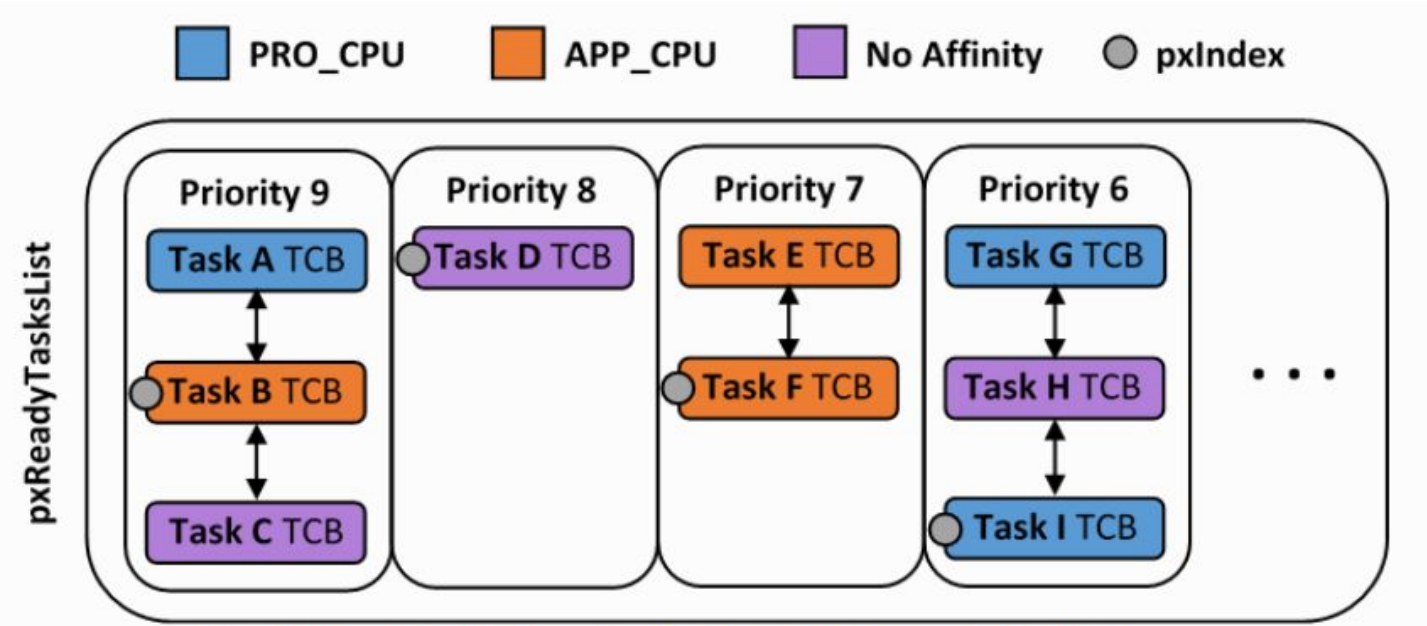


Illustration of FreeRTOS Ready Task List Data Structure in ESP-IDF

ESP-IDF FreeRTOS - Scheduling에서 Task Skip

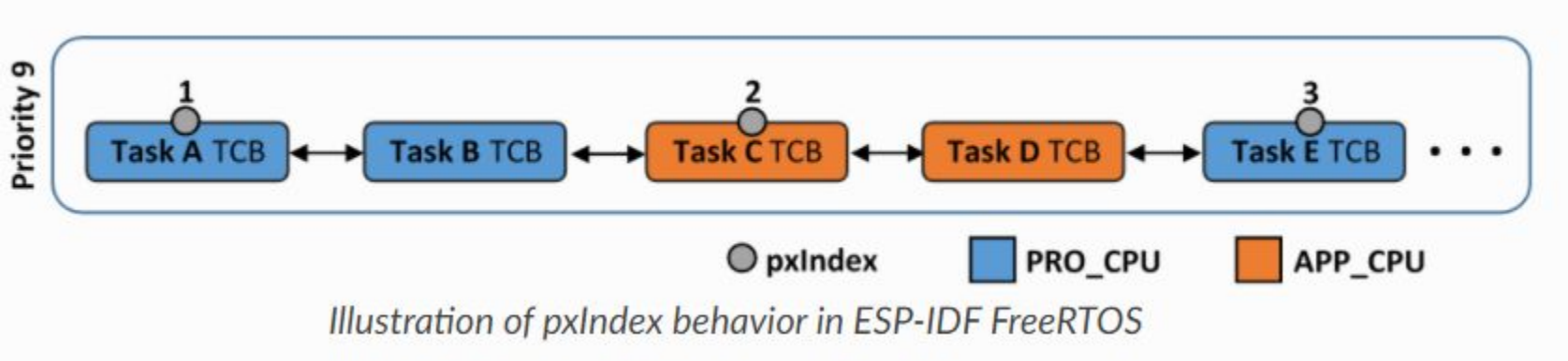


Illustration of pxIndex behavior in ESP-IDF FreeRTOS

1. PRO_CPU는 스케줄러 호출하고 Task A선택, pxIndex → A
2. APP_CPU는 스케줄러 호출하고 Task C선택으로 Task B는 Skip, pxIndex → C
3. PRO_CPU는 스케줄러 호출하고 Task E선택으로 Task D는 Skip, pxIndex → E

해결 방안

1. 모든 Task를 Blocked만들어 Ready Task List를 제거
2. 같은 우선순위에 다른 코어에 고정된 여러 Task가 할당되지 않도록 우선순위를 분산

Example18 – SMP System내 Task생성

```
#include <stdio.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

void vTask1(void *pvParameters)
{
    for(;;)
    {
        printf("VTask1\n");
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}

void vTask2(void *pvParameters)
{
    for(;;)
    {
        printf("VTask2\n");
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}
```

```
void vTask3(void *pvParameters)
{
    for(;;)
    {
        printf("VTask3\n");
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}

void app_main(void)
{
    int i = 0;

    printf("Hello world!\n");

    xTaskCreatePinnedToCore(vTask1, "Task 1", 1024, NULL, 3, NULL, 0);
    xTaskCreatePinnedToCore(vTask2, "Task 2", 1024, NULL, 3, NULL, 1);
    xTaskCreatePinnedToCore(vTask3, "Task 3", 1024, NULL, 3, NULL, tskNO_AFFINITY);

    while(1)
    {
        printf("Restarting in %d seconds...\n", i);
        i++;
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

ESP-IDF FreeRTOS - Scheduler Suspension

ESP-IDF FreeRTOS의 vTaskSuspendAll()은 호출한 코어의 스케줄러에만 영향
예) APP_CPU에서 xTaskSuspendAll() 호출은 PRO_CPU코어의 Task 스케줄러에는 영향이 없음

공유자원의 대한 보호는 세마포어를 사용

Example19 – SMP System나 vTaskSuspendAll

```
#include <stdio.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

int i = 0;

void vTask1(void *pvParameters)
{
    for(;;)
    {
        printf("VTask1\n");
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}
```

```
void vTask2(void *pvParameters)
{
    int j;
    for(;;)
    {
            printf("VTask2\n");
            if(i == 5)
            {
                vTaskSuspendAll();
                for(j=0; j<0x2FFFFFF; j++);
                xTaskResumeAll();
                printf("Scheduling Start\n");
            }
            vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}
```

Example19 – SMP System내 vTaskSuspendAll

```
void vTask3(void *pvParameters)
{
    for(;;)
    {
        printf("VTask3\n");
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}

void app_main(void)
{
    printf("Hello world!\n");

    xTaskCreatePinnedToCore(vTask1, "Task 1", 1024, NULL, 3, NULL, 0);
    xTaskCreatePinnedToCore(vTask2, "Task 2", 1024, NULL, 3, NULL, 1);
    xTaskCreatePinnedToCore(vTask3, "Task 3", 1024, NULL, 3, NULL, tskNO_AFFINITY);

    while(1)
    {
            printf("Restarting in %d seconds...\n", i);
        i++;
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

감사합니다.