

RTOS





- ▶ RTOS가 무엇인지 설명할 수 있다.
- ▶ Nucleo-F429보드의 FreeRTOS를 사용할 수 있다.



- ▶ RTOS 소개
- ▶ FreeRTOS 사용하기



RTOS란?

RTOS

RTOS가 왜 필요한가?

... LD2와 LD3를 500ms 마다 토글하는 코드를 작성한다고 하자. 그러면 다음과 같다.

```
int main(void)
{
    while(1)
    {
        HAL_GPIO_TogglePin(GPIOB, LD2_Pin);
        HAL_GPIO_TogglePin(GPIOB, LD3_Pin);
        HAL_Delay(500);
    }
}
```

- ... 위와 같이 작성하면 근소한 시간이지만 LD2가 동작한 후 LD3가 동작할 것이다.
- ... LD2와 LD3의 동작이 서로 전혀 상관 관계 없이 개별적으로 동작시키려면 어떻게 할까?

RTOS란?

RTOS

RTOS가 왜 필요한가?

...> LD2와 LD3가 서로 개별적으로 동작하기 위해 아래와 같이 작성하면 어떻게 될까?

```
int main(void)
{
    while(1)
    {
        HAL_GPIO_TogglePin(GPIOB, LD2_Pin);
        HAL_Delay(500);
    }
    while(1)
    {
        HAL_GPIO_TogglePin(GPIOB, LD3_Pin);
        HAL_Delay(500);
    }
}
```

RTOS란?

RTOS

RTOS가 왜 필요한가?

- LD2가 500ms마다 동작하는 것을 하나의 task로 하고 LD3가 500ms마다 동작하는 것을 또 다른 task로 만들어 두 개의 무한 루프(=task)가 서로 번갈아 동작하게 만드는 것이 RTOS임.

```
int main(void)
{
    osThreadDef(LED2, LED_Thread2, osPriorityNormal, 0, configMINIMAL_STACK_SIZE);
    osThreadDef(LED3, LED_Thread3, osPriorityNormal, 0, configMINIMAL_STACK_SIZE);
    LED2_ThreadId = osThreadCreate(osThread(LED2), NULL);
    LED3_ThreadId = osThreadCreate(osThread(LED3), NULL);
}
```

```
static void LED_Thread2(void const *argument)
{
    while(1)
    { HAL_GPIO_TogglePin(GPIOB,LD2_Pin);
      osDelay(500);
    }
}
```

```
static void LED_Thread3(void const *argument)
{
    while(1)
    { HAL_GPIO_TogglePin(GPIOB,LD3_Pin);
      osDelay(500);
    }
}
```

RTOS란?

RTOS

RTOS의 정의

- ... RTOS는 Real Time Operating System의 약자
- ... 실시간 운영체제
- ... 임베디드 시스템은 특정목적에만 사용하는 용도로 설계되었기 때문에 자원이 풍부하지 않아 굳이 운영체제를 사용할 필요가 없었다
- ... 하지만 임베디드 시스템도 계속 고사양, 고도화되면서 OS의 필요성을 느끼게 되어 현재는 수많은 임베디드 시스템 제품에 OS가 사용되고 있다
- ... 미사일, 비행기와 같은 군사용 임베디드 시스템은 매우 정확한 동작이 요구되기 때문에 일반적인 운영 체제는 사용하기 어렵고 매우 빠른 운영체제가 필요
- ... RTOS는 주어진 작업을 정해진 시간 안에 수행 하고 예측 가능하며 일정한 응답 시간을 요구하는 시스템에 적합

RTOS란?

RTOS

RTOS의 특징

... multi-tasking 지원

- 여러 개의 task를 동시에 동작시킬 수 있다.
- 각 task마다 우선순위를 둘 수 있다.

... 짧은 interrupt latency

- Interrupt latency는 실제 인터럽트가 발생한 후 인터럽트 핸들러까지 도착하는 시간

... 작은 Kernel 사이즈

- 일반적인 OS인 윈도우나 리눅스의 경우 설치된 용량을 보면 수십 Mega바이트에서 수 giga 바이트의 크기
- RTOS는 작은 사이즈를 가지고 있으며 수십 Kilo 바이트 수준

RTOS란?

RTOS

유명한 RTOS

...> VxWorks

- 미국의 WindRiver사에서 제조, 판매. 세계 시장에서 점유율이 가장 높음
- 화성 탐사선 스피릿 로버등이 PowerPC 플랫폼에 VxWorks 운영체제를 탑재하여 유명

...> pSOS

- VxWorks가 유명해지기 전 가장 유명한 RTOS
- Integrated Systems사에서 제조 판매, 현재는 WindRiver사에 인수

...> Nucleus

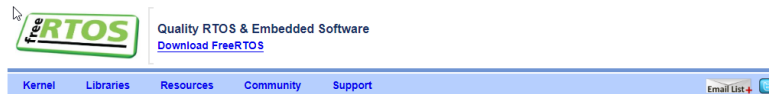
- Accelerated Technology사에서 제조 판매, 현재는 Mentor Graphics사에서 인수
- 일반적인 상용 RTOS는 소스 코드를 공개하지 않지만 Nucleus는 구매 고객에게 소스 코드를 공개
- 우리별 1호, 2호에도 탑재



STM32F429에 사용할 RTOS

Real Time Engineers사에서 개발한 오픈소스 형태의 RTOS

- <http://www.freertos.org/>
- 다양한 플랫폼을 지원하며 특히 ARM계열의 모든 플랫폼을 지원한다. ARM7, ARM9, Cortex-M시리즈, Cortex-A시리즈뿐만 아니라 AVR와 같은 8비트 MCU도 지원
- 라이선스는 MIT open source license



FreeRTOS™

Real-time operating system for microcontrollers

Developed in partnership with the world's leading chip companies over a 15-year period, and now downloaded every 175 seconds, FreeRTOS is a market-leading real-time operating system (RTOS) for microcontrollers and small microprocessors. Distributed freely under the MIT open source license, FreeRTOS includes a kernel and a growing set of libraries suitable for use across all industry sectors. FreeRTOS is built with an emphasis on reliability and ease of use.

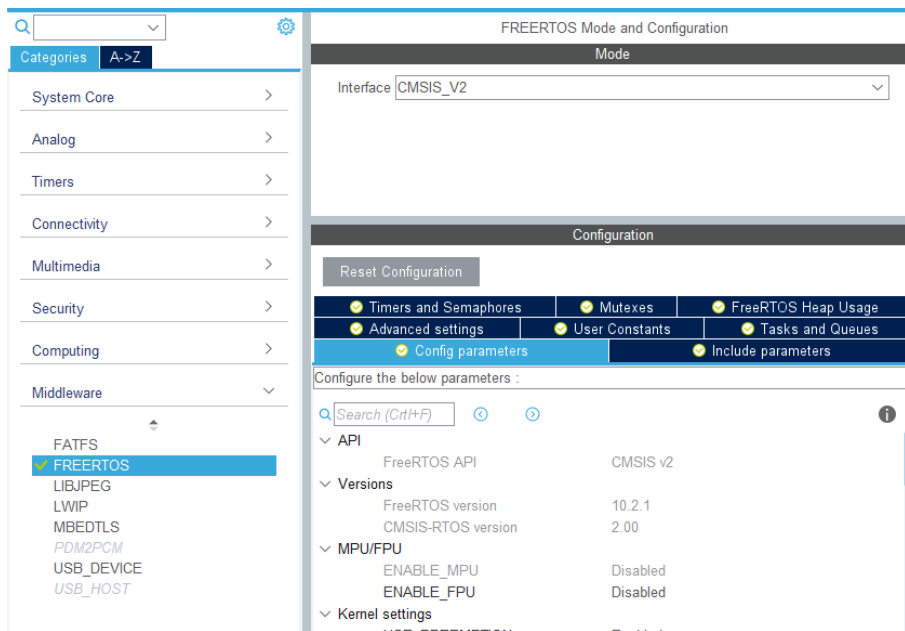
[Download FreeRTOS](#)

[Getting Started](#)

FreeRTOS

FreeRTOS

STM32CubeIDE로 FreeRTOS 생성




- ... Middleware에 FREERTOS선택
- ... Interface에 CMSIS_V1과 CMSIS_V2를 선택가능
- ... 여기서는 CMSIS_V2 선택
- ... CMSIS는 Cortex Microcontroller Software Interface Standard 로 ARM Cortex 시리즈끼리의 호환성을 위해 ARM사에서 정한 소프트웨어 인터페이스 표준

FreeRTOS

STM32CubeIDE로 FreeRTOS 생성

→ RTOS는 기본적인 동작을 위해 timer가 필요

 Warning: Code Generation



WARNINGS:

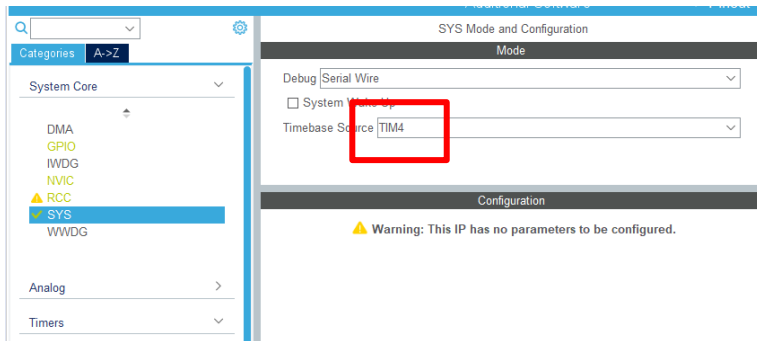
- When FreeRTOS is used, it is strongly recommended to use a HAL timebase source other than the SysTick.
The HAL timebase source can be changed from the Pinout tab under SYS

Do you still want to generate code ?

Yes

No

→ “FreeRTOS의 timebase source를 SysTick으로 사용하는 것을 권장하지 않음”이라는 경고 메시지

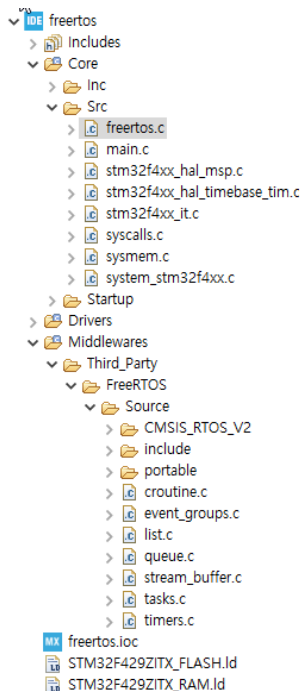


→ SYS블록 설정을 보면 Timbase Source가 나오는데 기존 설정인 SysTick대신 TIM4로 수정. 다른 Timer를 선택해도 무방함.

FreeRTOS

FreeRTOS의 구조

FreeRTOS 소스



- ... Src 디렉토리에 freertos.c와 stm32f4xx_hal_timebase_tim.c가 새로 생성됨.
- ... FreeRTOS의 소스는 Middlewares/Third_Party/FreeRTOS/Source라는 디렉토리에 존재



FreeRTOS의 구조



Src/freertos.c

```
freertos.ioc freertos.c main.c
1  /* USER CODE BEGIN Header */
2  /**
3   *
4   * File Name      : freertos.c
5   * Description    : Code for freertos applications
6   *
7   * @attention
8   *
9   * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
10  * All rights reserved.</center></h2>
11  *
12  * This software component is licensed by ST under Ultimate Liberty license
13  * SLA0044, the "License"; You may not use this file except in compliance with
14  * the License. You may obtain a copy of the License at:
15  * www.st.com/SLA0044
16  *
17  *
18  */
19  /* USER CODE END Header */
20
21  /* Includes -----*/
22  #include "FreeRTOS.h"
23  #include "task.h"
24  #include "main.h"
25
26  /* Private includes -----*/
27  /* USER CODE BEGIN Includes */
28
29  /* USER CODE END Includes */
30
31  /* Private typedef -----*/
32  /* USER CODE BEGIN PTN */
```

- ... freertos의 application code를 추가할 수 있도록 만들어준 파일
- ... USER CODE로 명명된 자리에 필요한 freertos의 application code를 추가



FreeRTOS의 구조



Src/stm32f4xx_hal_timebase_tim.c

```
freertos.ioc  freertos.c  main.c  stm32f4xx_hal_timebase_tim.c
40  * @retval HAL status
41  */
42  HAL_StatusTypeDef HAL_InitTick(uint32_t TickPriority)
43  {
44      RCC_ClkInitTypeDef  clkconfig;
45      uint32_t             uwTimclock = 0;
46      uint32_t             uwPrescalerValue = 0;
47      uint32_t             pFLatency;
48
49      /*Configure the TIM4 IRQ priority */
50      HAL_NVIC_SetPriority(TIM4_IRQn, TickPriority ,0);
51
52      /* Enable the TIM4 global Interrupt */
53      HAL_NVIC_EnableIRQ(TIM4_IRQn);
54
55      /* Enable TIM4 clock */
56      __HAL_RCC_TIM4_CLK_ENABLE();
57
58      /* Get clock configuration */
59      HAL_RCC_GetClockConfig(&clkconfig, &pFLatency);
60
61      /* Compute TIM4 clock */
62      uwTimclock = 2*HAL_RCC_GetPCLK1Freq();
63
64      /* Compute the prescaler value to have TIM4 counter clock equal to 1MHz */
65      uwPrescalerValue = (uint32_t) ((uwTimclock / 1000000) - 1);
66
67      /* Initialize TIM4 */
68      htim4.Instance = TIM4;
69
```

- ... freeRTOS의 timer관련 코드가 존재
- ... TIM4 설정을 반영한 코드가 HAL_InitTick()함수



FreeRTOS의 구조



Src/main.c

```
51 /* Definitions for defaultTask */
52 osThreadId_t defaultTaskHandle;
53 const osThreadAttr_t defaultTask_attributes = {
54     .name = "defaultTask",
55     .priority = (osPriority_t) osPriorityNormal,
56     .stack_size = 128 * 4
57 };
58 /* USER CODE BEGIN PV */
59
114 /* Init scheduler */
115 osKernelInitialize();
116
117 /* USER CODE BEGIN RTOS_MUTEX */
118 /* add mutexes, ... */
119 /* USER CODE END RTOS_MUTEX */
120
121 /* USER CODE BEGIN RTOS_SEMAPHORES */
122 /* add semaphores, ... */
123 /* USER CODE END RTOS_SEMAPHORES */
124
125 /* USER CODE BEGIN RTOS_TIMERS */
126 /* start timers, add new ones, ... */
127 /* USER CODE END RTOS_TIMERS */
128
129 /* USER CODE BEGIN RTOS_QUEUES */
130 /* add queues, ... */
131 /* USER CODE END RTOS_QUEUES */
132
133 /* Create the thread(s) */
134 /* creation of defaultTask */
135 defaultTaskHandle = osThreadNew(StartDefaultTask, NULL, &defaultTask_attributes);
136
137 /* USER CODE BEGIN RTOS_THREADS */
138 /* add threads, ... */
139 /* USER CODE END RTOS_THREADS */
140
141 /* Start scheduler */
142 osKernelStart();
```

- 프로젝트 생성 후 기본적으로 StartDefaultTask라는 기본 태스크가 생성됨.
- defaultTask_attributes는 StartDefaultTask의 속성으로 태스크 이름, 태스크간 우선 순위, 태스크의 stack 사이즈
- osKernelInitialize()는 스케줄러를 초기화
- osThreadNew를 통해 StartDefaultTask를 defaultTask_attributes라는 속성으로 생성
- osKernelStart()는 스케줄러 시작

Q & A
Thank you

