Xilinx Zynq FPGA, TI DSP, MCU 기반의 회로 설계 및 임베디드 전문가 과정

강사 - Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 - TaeYoung Eun(은태영)

zero_bird@naver.com

1.8 이벤트 그룹 API

1.8.1 이벤트 그룹 생성 및 삭제

1.8.1.1 xEventGroupCreate : 이벤트 그룹 동적 생성

```
#include “FreeRTOS.h”
#include “event_qroup.h”

EventGroupHandle_t xEventGroupCreate ( void );
```

새 이벤트 그룹을 생성하고 생성된 이벤트 그룹을 참조할 핸들을 반환한다. 각 이벤트 그룹에는 이벤트 그룹의 상태를 유지하는데 사용되는 매우 적은 양의 RAM 이 필요하다. 이벤트 그룹이 xEventGroupCreate()를 사용하여 만들어진다면 RTOS 힙 메모리에서 자동으로 할당된다. 이벤트 그룹이 xEventGroupCreateStatic()을 사용하여 생성되면 컴파일 시 RAM 을 정적으로 할당할 수 있도록 응용프로그램 작성자가 RAM 을 제공한다. 이벤트 그룹은 EventGroupHandle_t 유형의 변수에 저장된다.

이벤트 그룹 내에서 구현되는 비트(플래그) 수는 configUSE_16_BIT_TICKS 가 1 로 설정된 경우 8 비트이며, 0 으로 설정된 경우 24 비트이다. configUSE_16_BIT_TICKS 에 대한 의존성은 RTOS task 의 내부 구현에서 thread local storage 에 사용되는 데이터 유형의 결과이다.

해당 함수는 인터럽트에서 호출할 수 없다.

FreeRTOSConfig .h 의 configSUPPORT_DYNAMIC_ALLOCATION 가 1 로 설정되어 있어야 하고(정의되지 않은 채로 남겨두면 기본 값이 1 이다.), RTOS 소스 파일 FreeRTOS / source / event_groups.c 가 빌드에 포함되어 있어야 한다.

1.8.1.1.1 반환 값

NULL : 사용 가능한 FreeRTOS 힙 메모리가 충분하지 않아서 이벤트 그룹을 생성할 수 없다.

다른 값 : 이벤트 그룹이 생성되고 반환된 값은 생성된 이벤트 그룹의 핸들이다.

1.8.1.1.2 기타

```
// 생성될 이벤트 그룹을 저장할 변수를 선언한다.
EventGroupHandle_t xCreatedEventGroup;

// 이벤트 그룹을 생성한다.
xCreatedEventGroup = xEventGroupCreate();

// 이벤트 그룹의 성공 유무를 확인한다.
if( xCreatedEventGroup == NULL ) {
   // FreeRTOS 힙 메모리가 부족하여 이벤트 그룹 생성에 실패했다.
} else {
   // 이벤트 그룹이 생성되었다.
}
```

1.8.1.2 xEventGroupCreateStatic : 이벤트 그룹 정적 생성

```
#include “FreeRTOS.h”
#include “event_qroup.h”

EventGroupHandle_t xEventGroupCreateStatic ( StaticEventGroup_t * pxEventGroupBuffer );
```

새 이벤트 그룹을 생성하고 생성된 이벤트 그룹을 참조할 핸들을 반환한다. 각 이벤트 그룹에는 이벤트 그룹의 상태를 유지하는데 사용되는 매우 적은 양의 RAM 이 필요하다. 이벤트 그룹이 xEventGroupCreate()를 사용하여 만들어진다면 RTOS 힙 메모리에서 자동으로 할당된다. 이벤트 그룹이 xEventGroupCreateStatic()을 사용하여 생성되면 컴파일 시 RAM 을 정적으로 할당할 수 있도록 응용프로그램 작성자가 RAM 을 제공한다. 이벤트 그룹은 EventGroupHandle_t 유형의 변수에 저장된다.

이벤트 그룹 내에서 구현되는 비트(플래그) 수는 configUSE_16_BIT_TICKS 가 1 로 설정된 경우 8 비트이며, 0 으로 설정된 경우 24 비트이다. configUSE_16_BIT_TICKS 에 대한 의존성은 RTOS task 의 내부 구현에서 thread local storage 에 사용되는 데이터 유형의 결과이다.

해당 함수는 인터럽트에서 호출할 수 없다.

FreeRTOSConfig .h 의 configSUPPORT_DYNAMIC_ALLOCATION 가 1 로 설정되어 있어야 하고(정의되지 않은 채로 남겨두면 기본 값이 1 이다.), RTOS 소스 파일 FreeRTOS / source / event_groups.c 가 빌드에 포함되어 있어야 한다.

1.8.1.2.1 매개 변수

pxEventGroupBuffer : 이벤트 그룹의 데이터 구조가 저장될 StaticEventGroup_t 유형의 변수를 가리킨다.

1.8.1.2.2 반환 값

NULL : pxEventGroupBuffer 가 NULL 이기 때문에 이벤트 그룹을 생성할 수 없다.

다른 값 : 이벤트 그룹이 작성되고 반환된 값은 생성된 이벤트 그룹의 핸들이다.

1.8.1.2.3 기타

```
// 생성된 이벤트 그룹을 저장할 핸들을 변수 선언한다.
EventGroupHandle_t xEventGroupHandle;

// 생성된 이벤트 그룹과 연관된 데이터를 보유할 변수를 선언한다.
StaticEventGroup_t xCreatedEventGroup;

void vAFunction( void ) {
    // 이벤트 그룹을 생성한다.
    xEventGroupHandle = xEventGroupCreate( &xCreatedEventGroup );

    // pxEventGroupBuffer 가 NULL 이 아니기 때문에 이벤트 그룹이 생성될 것으로 예측한다.
    configASSERT( xEventGroupHandle );
}
```

1.8.1.3 vEventGroupDelete : 이벤트 그룹의 삭제

```
#include “FreeRTOS.h”
#include “event_qroup.h”

void vEventGroupDelete ( EventGroupHandle_t * xEventGroup );
```

xEventGroupCreate()를 호출하여 이전에 생성된 이벤트 그룹을 삭제한다. 삭제되는 이벤트 그룹의 차단 상태 task 는 차단 생태에서 해제되고 이벤트 그룹 값에 0 을 보고한다. 이 함수는 인터럽트에서 호출되면 안된다.

vEventGroupDelete()함수를 사용할 수 있도록 RTOS 소스 파일 FreeRTOS / source / event_groups.c 를 빌드에 포함해야 한다

1.8.1.3.1 매개 변수

xEventGroup : 삭제할 이벤트 그룹이다.

1.8.2 이벤트 그룹 기능 함수

1.8.2.1 xEventGroupGetBits : 이벤트 그룹의 플래그 확인

```
#include “FreeRTOS.h”
#include “event_qroup.h”

EventBits_t xEventGroupGetBits ( EventGroupHandle_t xEventGroup );
```

이벤트 그룹의 이벤트 비트(플래그)의 현재 값을 반환한다.

이 함수는 인터럽트에서 사용할 수 없다. 인터럽트에서 사용할 수 있는 버전은 xEventGroupGetBitsFromISR()를 참조한다.

xEventGroupGetBits()함수를 사용할 수 있도록 RTOS 소스 파일 FreeRTOS / source / event_groups.c 를 빌드에 포함해야 한다.

1.8.2.1.1 매개 변수

xEventGroup : 확인하고자 하는 이벤트 그룹이다. 이벤트 그룹은 xEventGroupCreate()를 호출하여 미리 만들어져 있어야 한다.

1.8.2.1.2 반환 값

모든 값 : xEventGroupGetBits()에서 호출한 이벤트 그룹의 이벤트 비트 값이다.

1.8.2.2 xEventGroupGetBitsFromISR : ISR 에서 이벤트 그룹의 플래그 확인

```
#include “FreeRTOS.h”
#include “event_qroup.h”

EventBits_t xEventGroupGetBitsFromISR ( EventGroupHandle_t xEventGroup );
```

인터럽트에서 호출할 수 있는 xEventGroupGetBits()의 버전이다.

1.8.2.2.1 매개 변수

xEventGroup : 확인하고자 하는 이벤트 그룹이다. 이벤트 그룹은 xEventGroupCreate()를 호출하여 미리 만들어져 있어야 한다.

1.8.2.2.2 반환 값

모든 값 : xEventGroupGetBitsFromISR()에서 호출한 이벤트 그룹의 이벤트 비트 값이다.

1.8.2.3 xEventGroupSetBits : 이벤트 그룹의 비트 설정

```
#include “FreeRTOS.h”
#include “event_qroup.h”

EventBits_t xEventGroupSetBits ( EventGroupHandle_t xEventGroup, Const EventBit_t uxBitsToSet );
```

RTOS 이벤트 그룹 내의 비트(플래그)를 설정한다.

이 함수는 인터럽트에서 호출할 수 없다. 인터럽트에서 호출할 수 있는 버전은 xEventGroupSetBitsFromISR()을 참조한다.

이벤트 그룹의 비트 설정은 비트 설정을 대기하는 동안 차단 상태인 task 를 자동으로 차단 해제한다.

xEventGroupSetBits()함수가 사용 가능하도록, RTOS 소스 파일 FreeRTOS / source / event_groups.c 를 빌드에 포함시켜야 한다.

1.8.2.3.1 매개 변수

xEventGroup : 비트가 설정되는 이벤트 그룹이다. 이벤트 그룹은 xEventGroupCreate()를 호출하여 미리 만들어져 있어야 한다.

uxBitsToSet : 이벤트 그룹에 설정할 비트 값이다. 예를 들어 3 비트를 설정하려면 uxBitsToSet 를 0x08 로 설정한다. 3 비트와 0 비트를 설정하려면 uxBitsToSet 를 0x09 로 설정한다.

1.8.2.3.2 반환 값

xEventGroupSetBits()의 호출이 반환된 이벤트 그룹의 비트 값이다. 반환 값이 uxBitsToSet 매개 변수에 의해 지정된 비트를 지우는 경우는 두가지가 있다.

1. 비트를 설정하면, 차단 상태에서 벗어나는 비트를 기다리고 있는 task 가 발생하면서 해당 비트가 자동으로 지워질 수 있다. (xEventGroupWaitBits()의 xClearBitsOnExit 매개 변수를 참조한다.)

2. xEventGroupSetBits()를 호출한 task 의 우선 순위보다 높은 우선 순위의 비트(또는 준비 상태의 task)로 인하여, 차단 상태를 유지하던 task 가 실행되기 전에 해당 task 를 반환하고 이벤트를 변경할 수 있다.

1.8.2.3.3 기타

```
#define BIT_0 ( 1 << 0 )
#define BIT_4 ( 1 << 4 )

void aFunction( EventGroupHandle_t xEventGroup ) {
    EventBits_t uxBits;

    // xEventGroup 으로 0 비트와 4 비트가 설정된다.
    uxBits = xEventGroupSetBits( xEventGroup, BIT_0 | BIT_4 );

    // 비트가 세팅되었다.
    if( ( uxBits & ( BIT_0 | BIT_4 ) ) == ( BIT_0 | BIT_4 ) ) {
      // 함수가 반환될 때 0 비트와 4 비트 모두 설정되어 있다.
    } else if( ( uxBits & BIT_0 ) != 0 ) {
      /* 0 비트는 함수가 반환될 때 설정되어 있지만 4 비트는 지워졌다.
      4 비트를 기다리고 있던 task 가 차단 상태에서 해제되어 4 비트가 자동으로 지워질 수도 있다. */
    } else if( ( uxBits & BIT_4 ) != 0 ) {
      /* 4 비트는 함수가 반환될 때 설정되어 있지만 0 비트는 지워졌다.
      0 비트를 기다리고 있던 task 가 차단 상태에서 해제되어 0 비트가 자동으로 지워질 수도 있다. */
    } else {
      /* 0 비트와 4 비트가 설정되지 않았다.
      task 가 두 비트를 모두 설정할 때까지 대기중인 task 가 차단 상태에서 해제되어 비트가 지워질 수도 있다. */
    }
}
```

1.8.2.4 xEventGroupSetBitsFromISR : ISR 에서 이벤트 그룹의 비트 설정

```
#include “FreeRTOS.h”
#include “event_qroup.h”

BaseType_t xEventGroupSetBitsFromISR ( EventGroupHandle_t xEventGroup, Const EventBit_t uxBitsToSet,
                                       BaseType_t * pxHigherPriorityTaskWoken );
```

이벤트 그룹 내에서 비트를 설정한다. 인터럽트 서비스 루틴(ISR)에서 호출할 수 있는 xEventGroupSetBits()버전이다.

이벤트 그룹의 비트 설정은 비트 설정을 대기하는 동안 차단 상태의 task 를 자동으로 차단 해제한다. 비트 또는 비트가 설정되기를 기다리는 task 의 숫자를 알 수 없으므로, 이벤트 그룹의 비트 설정을 결정하여 작동시키지 않는다. FreeRTOS 는 인터럽트나 크리티컬 섹션에서 결정적이지 않는 연산의 수행을 허용하지 않는다. 따라서 xEventGroupSetBitsFromISR()은 RTOS 데몬 task 에 메시지를 보내 데몬 task 의 컨텍스트에서 Set 작업을 수행한다. (크리티컬 섹션 대신 스케줄러 차단을 사용한다.)

데몬 task 의 우선순위는 FreeRTOSConfig.h 의 configTIMER_TASK_PRIORITY 에 의해 설정된다.

xEventGroupSetBitsFromISR()함수가 사용 가능하도록, RTOS 소스 파일 FreeRTOS / source / event_groups.c 를 빌드에 포함시켜야 한다. 또한, FreeRTOSConfig.h 의 INCLUDE_xEventGroupSetBitsFromISR, configUSE_TIMERS, INCLUDE_xTimerPendFunctionCall 을 모두 1 로 설정해야 한다.

1.8.2.4.1 매개 변수

xEventGroup : 비트가 설정되는 이벤트 그룹이다. 이벤트 그룹은 xEventGroupCreate()를 호출하여 미리 만들어져 있어야 한다.

uxBitsToSet : 이벤트 그룹에 설정할 비트 값이다. 예를 들어 3 비트를 설정하려면 uxBitsToSet 를 0x08 로 설정한다. 3 비트와 0 비트를 설정하려면 uxBitsToSet 를 0x09 로 설정한다.

pxHigherPriorityTaskWoken : xEventGroupSetBitsFromISR()을 호출하면 메시지가 RTOS 데몬 Task 로 전송된다. 데몬 task 의 우선순위가 현재 실행중인 task 의 우선 순위(인터럽트 된 task)보다 높으면 * pxHigherPriorityTaskWoken 는 xEventGroupSetBitsFromISR()에 의해 pdTRUE 로 설정되어, 인터럽트가 종료되기 전에 컨텍스트 스위치를 요청해야 하는 것을 알린다. 이러한 이유로 * xEventGroupSetBitsFromISR 은 pdFALSE 로 초기화 해야 한다.

1.8.2.4.2 반환 값

pdPASS : 메시지가 RTOS 데몬 task 에 전송되었다.

pdFAIL : timer command queue 가 가득 차서 메시지를 RTOS 데몬 task(또는 타이머 서비스 task)로 보낼 수 없다. 대기열의 길이는 FreeRTOSConfig.h 의 configTIMER_QUEUE_LENGTH 설정에 의해 결정된다.

1.8.2.4.3 기타

```
#define BIT_0 ( 1 << 0 )
#define BIT_4 ( 1 << 4 )

// 이벤트 그룹은 xEventGroupCreate()에 대한 호출에 의해 이미 생성되었다고 가정한다.
EventGroupHandle_t xEventGroup;

void anInterruptHandler( void ) {
    BaseType_t xHigherPriorityTaskWoken, xResult;
    // xHigherPriorityTaskWoken 을 pdFALSE 로 초기화 한다.
    xHigherPriorityTaskWoken = pdFALSE;

    // xEventGroup 에서 0 비트와 4 비트를 설정한다.
    xResult = xEventGroupSetBitsFromISR( xEventGroup, BIT_0 | BIT_4, &xHigherPriorityTaskWoken );

    // 메시지 전송 성공 여부를 체크한다.
    if( xResult != pdFAIL ) {
      /* xHigherPriorityTaskWoken 이 pdTRUE 로 설정되면 컨텍스트 스위치가 요청되야 한다. 사용된 매크로는 특정 포트에
      따라 다르며, portYIELD_FROM_ISR() 또는 END_SWITCHING_ISR() 이다. 사용중인 포트의 설명서 페이지를 참조한다. */
      portYIELD_FROM_ISR( xHigherPriorityTaskWoken );
    }
}
```

1.8.2.5 xEventGroupClearBits : 이벤트 그룹의 비트 클리어

```
#include “FreeRTOS.h”
#include “event_qroup.h”

EventBits_t xEventGroupClearBits ( EventGroupHandle_t xEventGroup, Const EventBit_t uxBitsToClear );
```

RTOS 이벤트 그룹 내의 비트(플래그)를 클리어 한다. 이 함수는 인터럽트에서 호출할 수 없다.

인터럽트에서 호출할 수 있는 버전은 xEventGroupClearBitsFromISR 를 참조한다.

RTOS 소스 파일인 FreeRTOS / source / event_groups.c 는 xEventGroupClearBits()함수가 사용 가능하도록 빌드에 포함되야 한다.

1.8.2.5.1 매개 변수

xEventGroup : 비트가 지워지는 이벤트 그룹이다. 이벤트 그룹은 xEventGroupCreate()에 대한 호출을 사용하여 먼저 만들어야 한다.

uxBitsToClear : 이벤트 그룹에서 지울 비트를 나타내는 비트 값이다.

예를 들어 비트 3 을 지우려면 0x08 로 설정한다. 비트 3 과 비트 0 을 지우려면 0x09 로 설정한다.

1.8.2.5.2 반환 값

비트가 제거되기 전에 이벤트 그룹의 비트 값이다.

1.8.2.5.3 기타

```
#define BIT_0 ( 1 << 0 )
#define BIT_4 ( 1 << 4 )

void aFunction( EventGroupHandle_t xEventGroup ) {
    EventBits_t uxBits;

    // xEventGroup 의 비트 0 과 4 를 지운다.
    uxBits = xEventGroupClearBits( xEventGroup, BIT_0 | BIT_4 );

    // 비트가 지워진다.
    if( ( uxBits & ( BIT_0 | BIT_4 ) ) == ( BIT_0 | BIT_4 ) ) {
      // xEventGroupClearBits()전에 0 비트와 4 비트 모두 설정되었다.
    } else if( ( uxBits & BIT_0 ) != 0 ) {
      // xEventGroupClearBits()가 호출되기 전에 0 비트가 설정되었다.
    } else if( ( uxBits & BIT_4 ) != 0 ) {
      // xEventGroupClearBits()가 호출되기 전에 4 비트가 호출되었다.
    } else {
      // 처음에는 0 비트나 4 비트가 설정되지 않았다.
    }
}
```

1.8.2.6 xEventGroupClearBitsFromISR : ISR 에서 이벤트 그룹의 비트 클리어

```
#include “FreeRTOS.h”
#include “event_qroup.h”

EventBits_t xEventGroupClearBitsFromISR ( EventGroupHandle_t xEventGroup, Const EventBit_t uxBitsToClear );
```

인터럽트에서 호출할 수 있는 xEventGroupClearBits()버전이다.

xEventGroupClearBitsFromISR()는 RTOS 데몬 task 에 메시지를 보내 데몬 task 컨텍스트에서 클리어 연산을 수행한다.

데몬 task 의 우선 순위는 FreeRTOSConfig.h 의 configTIMER_TASK_PRIORITY 에 의해 설정된다.

RTOS 소스 파일인 FreeRTOS / source / event_groups.c 는 xEventGroupClearBitsFromISR()함수가 사용 가능하도록 빌드에 포함되어야 한다.

1.8.2.6.1 매개 변수

xEventGroup : 비트가 지워지는 이벤트 그룹이다. 이벤트 그룹은 xEventGroupCreate()에 대한 호출을 사용하여 먼저 만들어야 한다.

uxBitsToClear : 이벤트 그룹에서 지울 비트를 나타내는 비트 값이다. 예를 들어 비트 3 을 지우려면 0x08 로 설정한다. 비트 3 과 비트 0 을 지우려면 0x09 로 설정한다.

1.8.2.6.2 반환 값

pdPASS : 메시지가 RTOS 데몬 task 로 전송되었다.

pdFAIL : timer command queue 가 꽉 찼기 때문에 메시지를 RTOS 데몬 task(타이머 서비스 task 라고도 한다)로 보낼 수 없다. 대기열의 길이는 FreeRTOSConfig.h 의 configTIMER_QUEUE_LENGTH 설정에 의해 정해진다

1.8.2.6.3 기타

```
#define BIT_0 ( 1 << 0 )
#define BIT_4 ( 1 << 4 )

// 이 코드는 xEventGroup 변수가 참조하는 이벤트 그룹이 xEventGroupCreate()를 호출하여 이미 생성되어 있다고 가정한다.
void anInterruptHandler( void ) {
    BaseType_t xSuccess;

    // xEventGroup 의 비트 0 과 비트 4 를 지운다.
    xSuccess = xEventGroupClearBitsFromISR( xEventGroup, BIT_0 | BIT_4 );

    // 비트가 지워진다
    if( xSuccess == pdPASS ) {
      // 클리어 비트 메시지가 데몬 task 로 전송되었다.
    } else {
      // 클리어 비트 메시지가 데몬 task 로 전송되지 않았다.
    }
}
```

1.8.2.7 xEventGroupSync : 이벤트 그룹 설정 비트 대기

```
#include “FreeRTOS.h”
#include “event_qroup.h”

EventBits_t xEventGroupSync ( EventGroupHandle_t xEventGroup, Const EventBit_t uxBitsToSet
                              Const EventBit_t uxBitsToWaitFor, TickType_t xTicksToWait );
```

이벤트 그룹 내에서 원자적으로 비트(플래그)를 설정한 다음, 동일한 이벤트 그룹 내에서 비트 조합을 설정할 때까지 기다린다. 이 기능은 일반적으로 여러 task(task rendezvous)를 동기화 하는 작업에 사용된다. 이 task 에서는 계속 진행하기 전에 각 task 가 동기화 지점에 도달할 때까지 기다려야 한다. uxBitsToWaitFor 매개 변수로 지정된 비트가 설정되거나, 해당 시간 내에 설정되면, 차단 시간이 만료되기

전에 함수가 반환된다. 이 경우 uxBitsToWaitFor 로 지정된 모든 비트는 함수가 반환되기 전에 자동으로 지워진다.

이 함수는 인터럽트에서 사용할 수 없다.

xEventGroupSync() 함수를 사용하기 위해 RTOS 소스 파일 FreeRTOS / source / event_groups.c 를 빌드에 포함해야 한다.

1.8.2.7.1 매개 변수

xEventGroup : 비트가 설정되고 테스트되는 이벤트 그룹이다. 이벤트 그룹은 xEventGroupCreate()에 대한 호출을 사용하여 미리 만들어야 한다.

uxBitsToSet : uxBitsToWaitFor 매개 변수로 지정된 모든 비트가 설정되어 있는지(또는 대기 중인지)를 결정하는 이벤트 그룹에 설정할 비트 값이다. 예를 들어 uxBitsToSet 을 0x04 로 설정하고, 이벤트 그룹 내에서 2 비트를 설정한다.

uxBitsToWaitFor : 이벤트 그룹 내에서 테스트 할 비트 또는 비트를 나타내는 비트 값이다. 예를 들어 0 비트와 2 비트를 기다리려면 uxBitsToWaitFor 를 0x05 로 설정해야 한다. uxBitsToWaitFor 를 0x07 로 설정하여 0 비트와 1 비트, 2 비트를 기다린다.

xTicksToWait : uxBitsToWaitFor 매개 변수 값으로, 지정된 모든 비트가 설정 될 때까지 대기하는 최대 시간(tick 으로 지정)이다.

1.8.2.7.2 반환 값

모든 값 : 대기중인 비트가 설정되거나, 블록 시간이 만기 된 시점의 이벤트 그룹 값이다. 반환 값을 테스트하여 어떤 비트가 설정되었는지 확인한다. 타임 아웃이 만료되어 xEventGroupSync()가 반환된 경우 반환된 값에 대기중인 모든 비트가 설정되지 않는다. 대기중인 모든 비트가 설정 되었기 때문에 xEventGroupSync()가 리턴 된 경우, 비트가 자동으로 지워지기 전에 반환 된 값이다.

1.8.2.7.3 기타

```
// 세가지 task 에서 사용되는 비트다.
#define TASK_0_BIT ( 1 << 0 )
#define TASK_1_BIT ( 1 << 1 )
#define TASK_2_BIT ( 1 << 2 )
#define ALL_SYNC_BITS ( TASK_0_BIT | TASK_1_BIT | TASK_2_BIT )

// 이벤트 그룹을 사용하여 세가지 task 를 동기화 한다. 이 이벤트 그룹은 이미 다른곳에서 작성되었다고 가정한다.
EventGroupHandle_t xEventBits;

void vTask0( void *pvParameters ) {
    EventBits_t uxReturn;
    TickType_t xTicksToWait = pdMS_TO_TICKS( 100 );

    for( ;; ) {
      // 여기에 작업 기능을 수행한다.

      /* 이 task 가 동기화 지점에 도달했음을 알리기 위해 이벤트 그룹에서 0 비트을 설정한다. 다른 두 task 는 ALL_SYNC_BITS 에
      의해 정의된 다른 두 비트를 설정한다. ALL_SYNC_BITS 비트가 모두 설정되면 세 task 모두 동기화 지점에 도달한다.
      최대 100ms 동안 기다린다. */
      // 이벤트 그룹, 설정할 빝트, 기다릴 비트, 대기 시간
      uxReturn = xEventGroupSync( xEventBits, TASK_0_BIT, ALL_SYNC_BITS, xTicksToWait );

      if( ( uxReturn & ALL_SYNC_BITS ) == ALL_SYNC_BITS ) {
        // xEventGroupSync()호출이 시간 초과되기 전에 세 task 모두 동기화 지점에 도달했다.
      }
    }
}

void vTask1( void *pvParameters ) {
    for( ;; ) {
      // 여기서 task 기능을 수행한다.
```

```
        /* 이 task 가 동기화 지점에 도달했음을 알리기 위해 이벤트 그룹에서 1 비트를 설정한다. 다른 두 task 는 ALL_SYNC_BITS 에
        의해 정의된 다른 두 비트를 설정한다. ALL_SYNC_BITS 의 모든 비트가 설정되면 세 task 모두 동기화 지점에 도달한다.
        이 일이 끝나기를 기다린다. */
        xEventGroupSync( xEventBits, TASK_1_BIT, ALL_SYNC_BITS, portMAX_DELAY );

        /* xEventGroupSync()가 불확실한 차단 시간으로 호출되었다.
        이 task 는 세 task 모두에 의해 동기화가 이루어진 경우에만 여기에 도달하므로 반환값을 테스트할 필요가 없다. */
    }
}

void vTask2( void *pvParameters ) {
    for( ;; ) {
      // 여기서 task 기능을 수행한다.

        /* 이 task 가 동기화 지점에 도달했음을 알리기 위해 이벤트 그룹에서 2 비트를 설정한다. 다른 두 task 는 ALL_SYNC_BITS 에
        의해 정의된 다른 두 비트를 설정한다. ALL_SYNC_BITS 가 모두 설정되면 세 task 모두 동기화 지점에 도달한다.
        이 일이 끝나기를 기다린다. */
        xEventGroupSync( xEventBits, TASK_2_BIT, ALL_SYNC_BITS, portMAX_DELAY );

        /* xEventGroupSync()가 불확실한 차단 시간으로 호출되었다.
        이 task 는 세 task 모두에 의해 동기화가 이루어진 경우에만 여기에 도달하므로 반환값을 테스트할 필요가 없다. */
    }
}
```

1.8.2.8 xEventGroupWaitBits : 이벤트 그룹 설정 비트 고급 대기

```
#include “FreeRTOS.h”
#include “event_qroup.h”

EventBits_t xEventGroupWaitBits ( Const EventGroupHandle_t xEventGroup, Const EventBit_t uxBitsToWaitFor,
                                  Const BaseType_t xClearOnExit, Const BaseType_t xWaitForAllBits,
                                  TickType_t xTicksToWait );
```

비트 또는 비트 그룹이 설정될 때까지 기다리는 경우, 차단 상태(제한 시간 포함)를 선택적으로 입력하여 RTOS 이벤트 그룹 내의 비트를 읽는다. 이 함수는 인터럽트에서 호출할 수 없다.

xEventGroupWaitBits()함수를 사용하기 위해선 RTOS 소스 파일 FreeRTOS / source / event_groups.c 을 빌드에 포함시켜야 한다.

1.8.2.8.1 매개 변수

xEventGroup : 비트가 테스트 되는 이벤트 그룹이다. 이벤트 그룹은 xEventGroupCreate()을 사용하여 이전에 만들어 져 있어야 한다.

uxBitsToWaitFor : 이벤트 그룹 내에서 테스트할 비트 또는 비트를 나타내는 비트 값이다. 예를들어 0 비트 또는 2 비트를 기다리려면 uxBitsToWaitFor 을 0x07 로 설정한다. uxBitsToWaitFor 은 0 으로 설정하면 안된다.

xClearOnExit : xClearOnExit 이 pdTRUE 로 설정된 경우, uxBitsToWaitFor 매개 변수로 전달된 값에 설정된 비트가 xEventGroupWaitBits()이 시간 초과 이외의 다른 이유로 반환된다면, xEventGroupWaitBits()가 반환하기 전에 이벤트 그룹에서 지워진다. 제한 시간은 xTicksToWait 매개 변수에 의해 설정된다.

xWaitForAllBits : xWaitForAllBits 는 다음과 같이 논리 AND 테스트(모든 비트를 설정해야 한다.) 또는 논리 OR 테스트(하나 이상의 비트를 설정해야 한다.)를 만드는데 사용한다. xWaitForAllBits 를 pdTRUE 로 설정하면 uxBitsToWaitFor 매개 변수로 전달된 값에 설정된 모든 비트가 이벤트 그룹에 설정되거나 지정된 차단 시간이 만기될 때, xEventGroupWaitBits()가 반환한다.

xWaitForAllBits 가 pdFALSE 로 설정된 경우, uxBitsToWaitFor 매개 변수로 전달된 값에 설정된 비트 중 하나가 이벤트 그룹에 설정되거나 지정된 블록 시간이 만료되면 xEventGroupWaitBits()를 반환한다.

xTicksToWait : uxBitsToWaitFor 에 의해 지정된 비트중 하나 또는 모두(xWaitForAllBits 값)를 기다리는 최대 시간(tick)이다.

1.8.2.8.2 반환 값

모든 값 : 대기중인 이벤트 비트가 설정되거나, 차단 시간이 만기된 시점의 이벤트 그룹의 값이다. 우선 순위가 높은 task 또는 인터럽트가 호출된 task 와, 차단 상태에서 xEventGroupWaitBits()함수를 종료하는 사이에 이벤트 비트의 값을 변경하면 이벤트 그룹의 이벤트 비트는 현재 값과 리턴된 값이 다를 수 있다. 반환 값을 테스트하여 어떤 비트가 설정되었는지 확인해야 한다.

시간이 만료되어 xEventGroupWaitBits()가 반환되면 대기중인 모든 비트가 설정되지 않는다. xEventGroupWaitBits()의 반환된 비트가 대기중일 때, xClearOnExit 매개 변수가 pdTRUE 로 설정된 경우 비트가 자동으로 지워지기 전에 리턴된 값이 이벤트 그룹 값이다.

1.8.2.8.3 기타

```
#define BIT_0 ( 1 << 0 )
#define BIT_4 ( 1 << 4 )

void aFunction( EventGroupHandle_t xEventGroup ) {
    EventBits_t uxBits;
    const TickType_t xTicksToWait = pdMS_TO_TICKS( 100 );

    // 이벤트 그룹 내에서 0 비트 또는 4 비트가 설정될 때까지 최대 100ms 동안 대기한다. 종료하기 전에 비트를 지운다.
    // 이벤트 그룹, 반환할 이벤트 그룹 내의 비트, BIT_0, BIT_4 를 지운다, 두 비트를 모두 기다리지 않고 처리한다. 100ms 대기한다.
    uxBits = xEventGroupWaitBits( xEventGroup, BIT_0 | BIT_4, pdTRUE, pdFALSE, xTicksToWait );

    if( ( uxBits & ( BIT_0 | BIT_4 ) ) == ( BIT_0 | BIT_4 ) ) {
      // 두 비트가 모두 설정되었기 때문에 xEventGroupWaitBits()가 반환된다.
    } else if( ( uxBits & BIT_0 ) != 0 ) {
      // BIT_0 이 설정되어 xEventGroupWaitBits()가 반환된다.
    } else if( ( uxBits & BIT_4 ) != 0 ) {
      // BIT_4 이 설정되어 xEventGroupWaitBits()가 반환된다.
    } else {
      //  xEventGroupWaitBits()는 BIT_0 또는 BIT_4 가 설정되지 않고 xTicksToWait 을 초과하여 반환된다.
    }
}
```