


Xilinx Zynq FPGA,TI DSP, MCU 기반의 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – 정한별

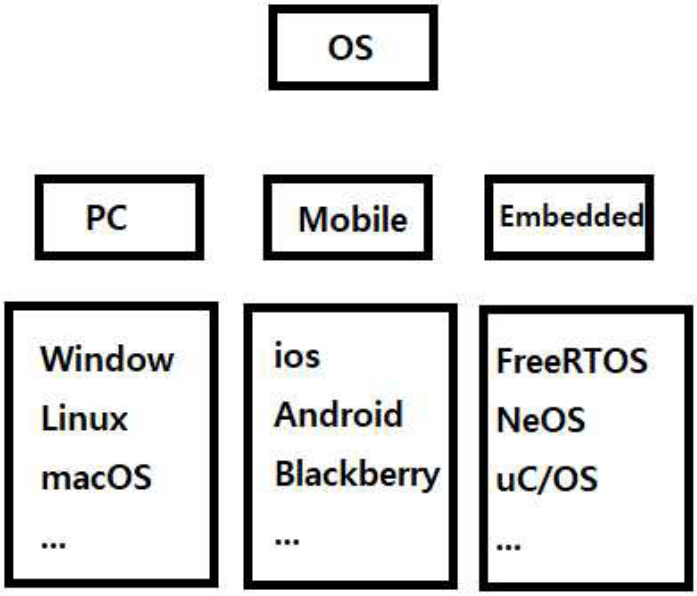
hanbulkr@gmail.com

FreeRTOS (Free Real Time OS) 프로그래밍 2018. 7. 31. 21:42
https://blog.naver.com/hanbulkr/221330288779

김형주라는 친구의 도움으로 만들었습니다.

What is FreeRTOS?

1.멀티 태스킹
FreeRTOS는 Amazon사에서 개발한 RTOS의 한 종류이고, RTOS는 OS중 특수한 OS를 말한다. OS라는 것은 우리가 흔히 사용하는 윈도우, 리눅스, 안드로이드, IOS 등을 말한다.



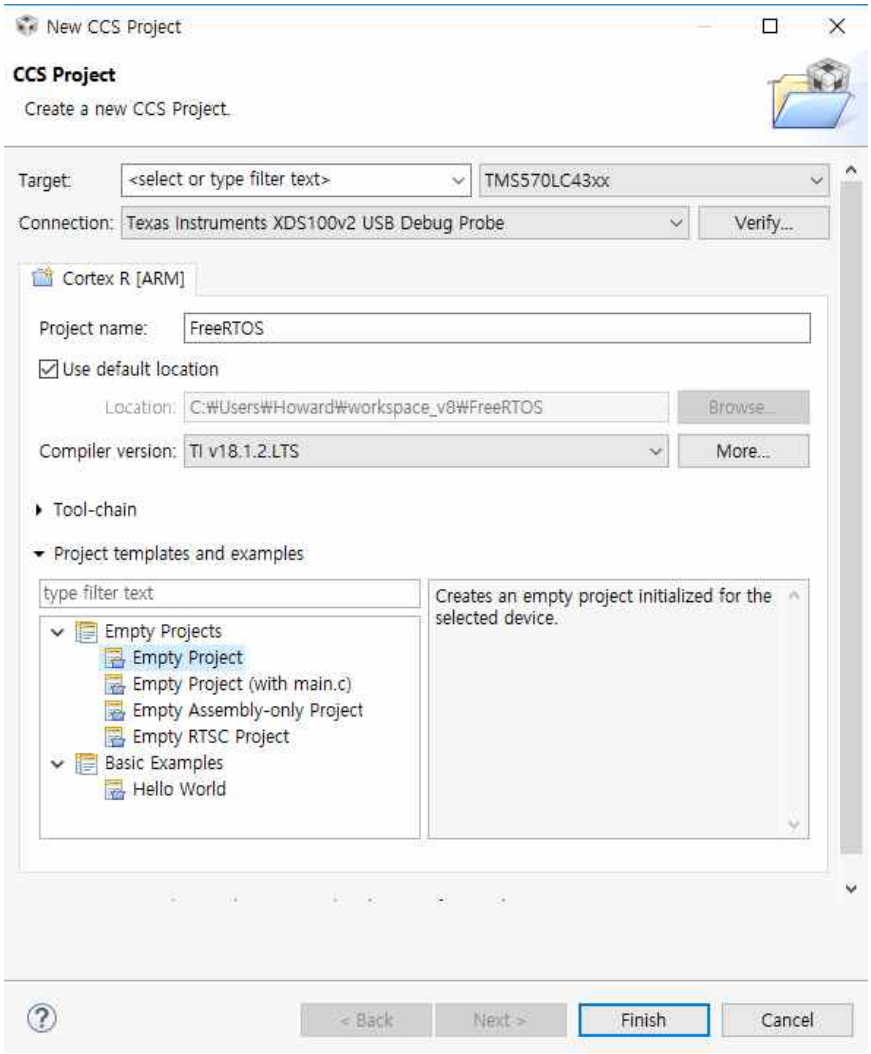
OS는 크게 5가지 기능을 가진다. 파일시스템 관리, 메모리 관리, 프로세스 관리, 입출력 장치 관리, 네트워크 관리를 담당한다. 임베디드 시스템에 일반 OS를 올리기에는 너무 무겁고, 비효율적이며 MMU를 탑재하고 있지 않아 독립적인 가상 메모리 공간을 가질 수 없어 멀티프로세싱이 불가능하다.

이런 종류의 제약이 너무 많기 때문에, RTOS는 OS의 기능 중 태스크 관리(프로세스 관리)만을 가져와 사용하는 OS라고 볼 수 있다.

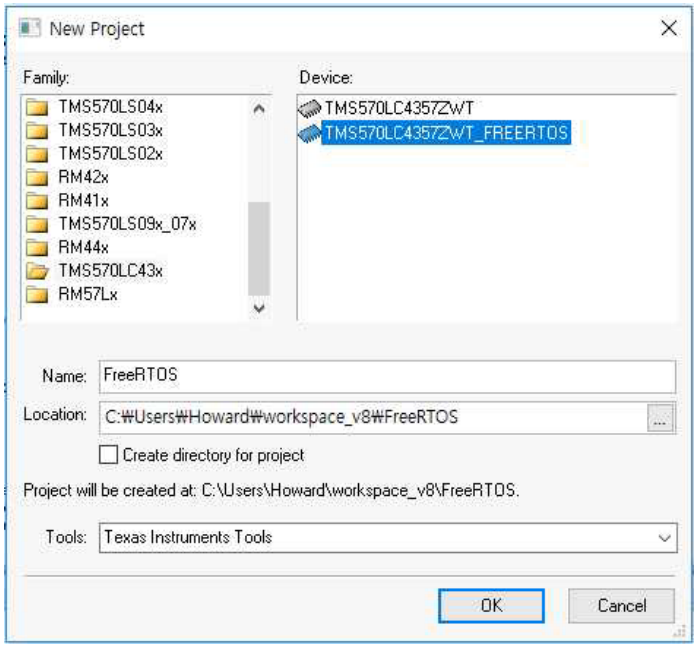
그렇기 때문에 여러가지 프로그램을 동시에 돌아가는 멀티 **프로세스 기술, 멀티 쓰레드 기능**이라고 생각하면 된다.

동시에 돌아가는 것이 아니지만 동시에 돌아가는 것처럼 보이게 해준다고 보면 된다.

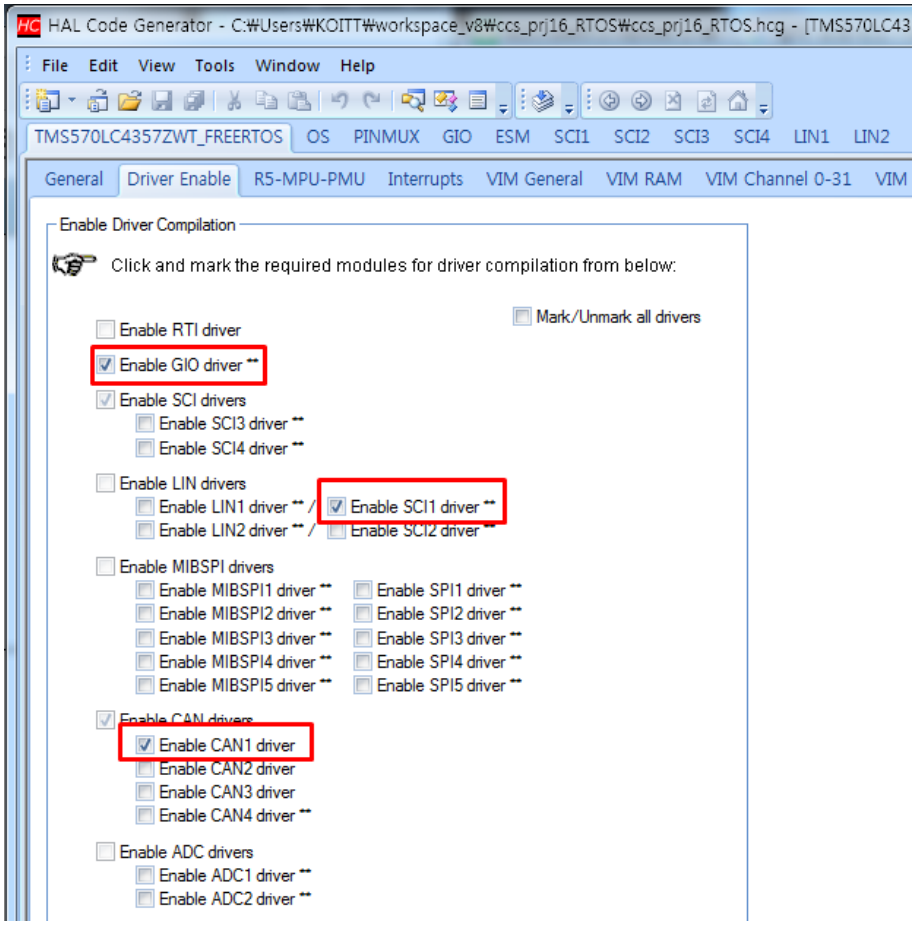
FreeRTOS _ HALCoGEN 설정 및 프로그램밍
(TI Hercules MCU - TMS570LC43xx Cortex R5F)

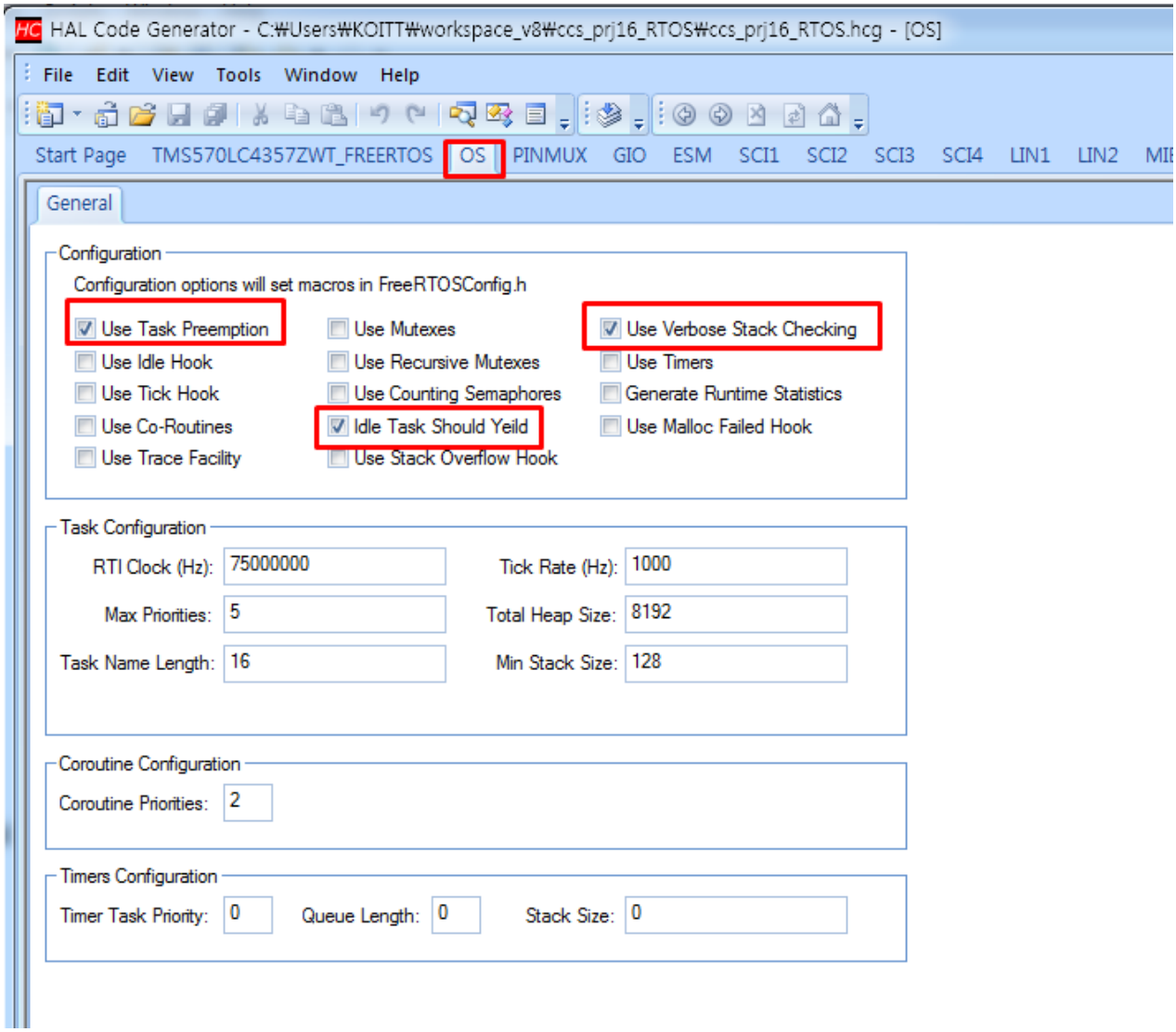


여기서 중요 하다 평소와 다르게 FREERTOS를 check를 해준다.

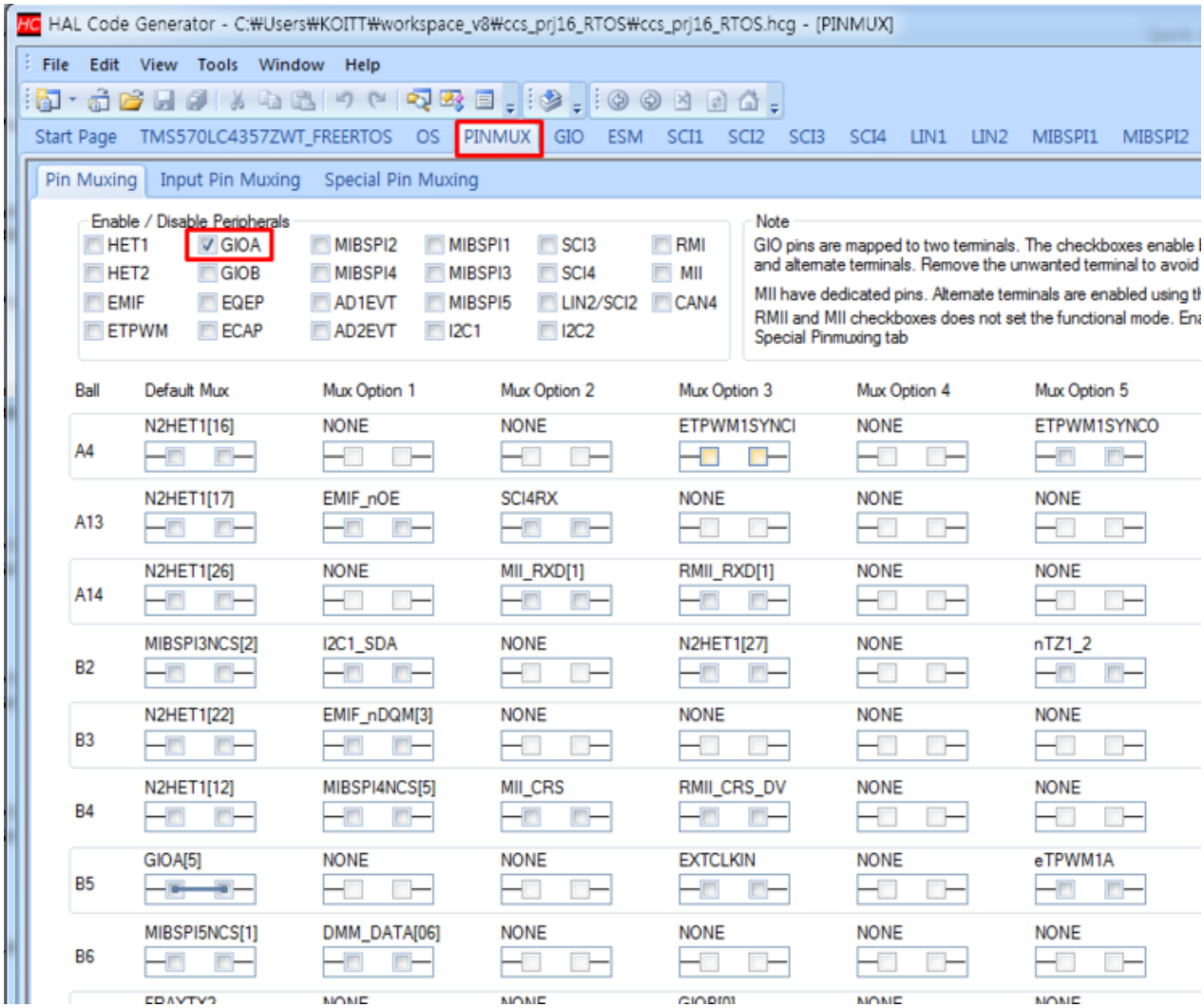


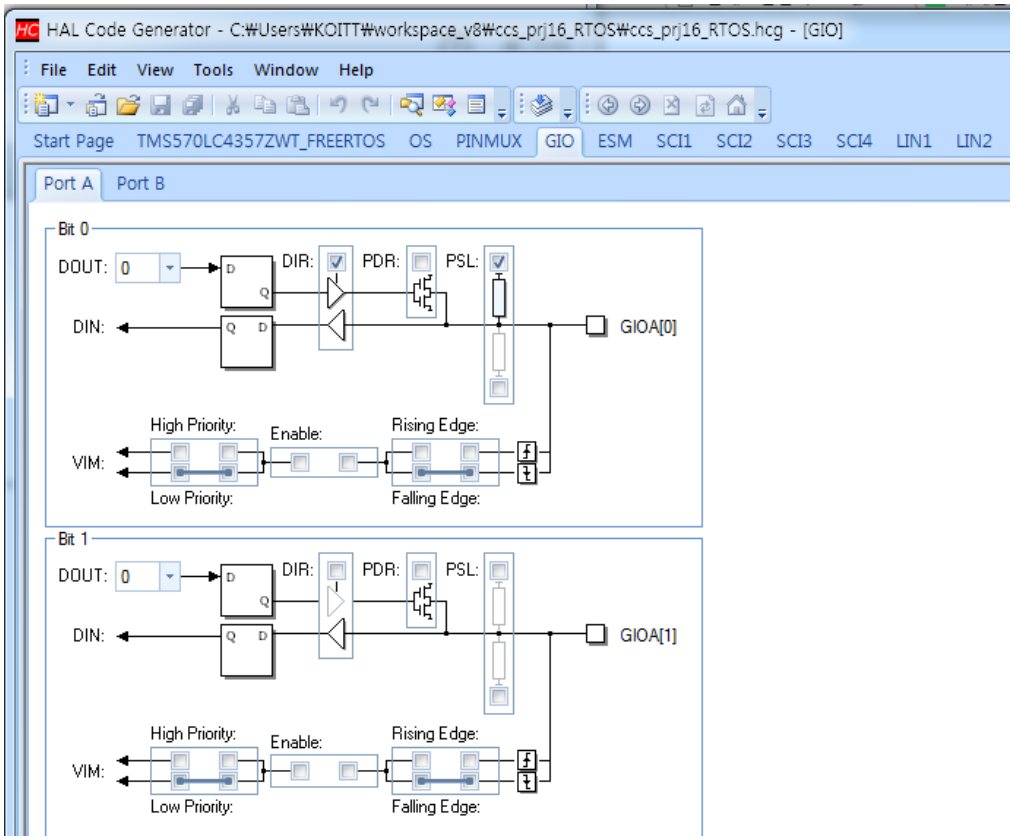
RTI가 set 되어서 선택 자체가 비활성화 된다. 이유는 RTOS를 선택하면 RTI기반으로 돌아가기 때문에 자동적로 setting이 무조건 되어 있어야 한다.

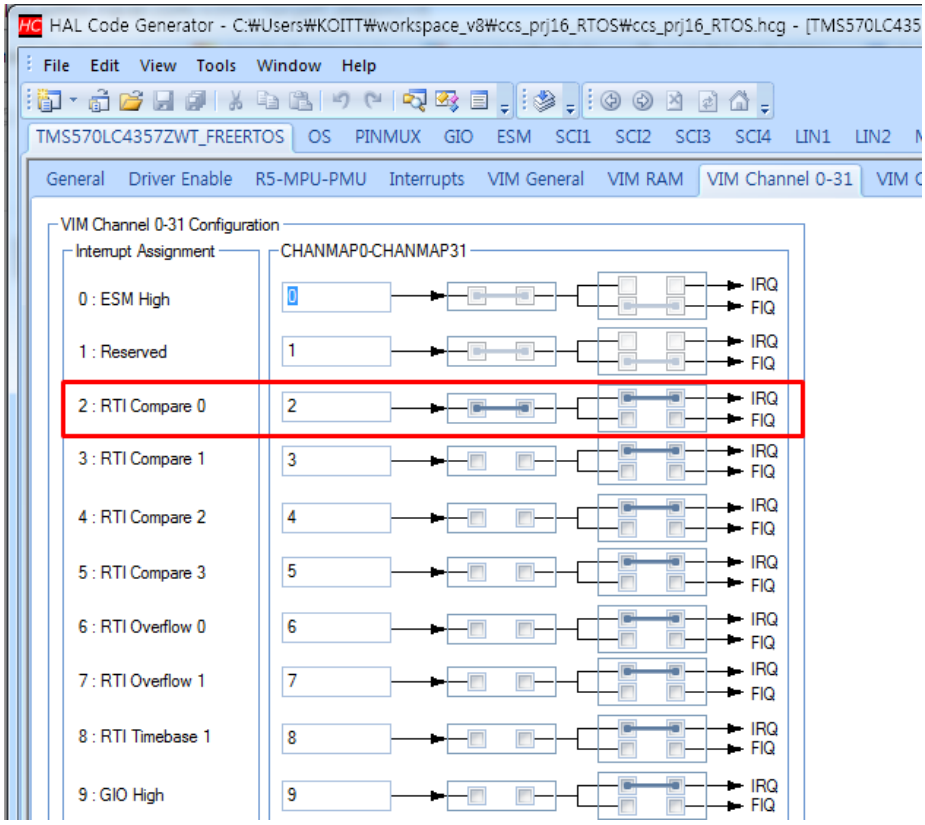

HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj16_RTOS₩ccs_prj16_RTOS.hcg - [TMS570LC43
File Edit View Tools Window Help
TMS570LC4357ZWT_FREERTOS OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2
General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31 VIM
Enable Driver Compilation
Click and mark the required modules for driver compilation from below:
Mark/Unmark all drivers
Enable RTI driver
Enable GIO driver **
Enable SCI drivers
Enable SCI3 driver **
Enable SCI4 driver **
Enable LIN drivers
Enable LIN1 driver ** / Enable SCI1 driver **
Enable LIN2 driver ** / Enable SCI2 driver **
Enable MIBSPI drivers
Enable MIBSPI1 driver ** Enable SPI1 driver **
Enable MIBSPI2 driver ** Enable SPI2 driver **
Enable MIBSPI3 driver ** Enable SPI3 driver **
Enable MIBSPI4 driver ** Enable SPI4 driver **
Enable MIBSPI5 driver ** Enable SPI5 driver **
Enable CAN drivers
Enable CAN1 driver
Enable CAN2 driver
Enable CAN3 driver
Enable CAN4 driver **
Enable ADC drivers
Enable ADC1 driver **
Enable ADC2 driver **

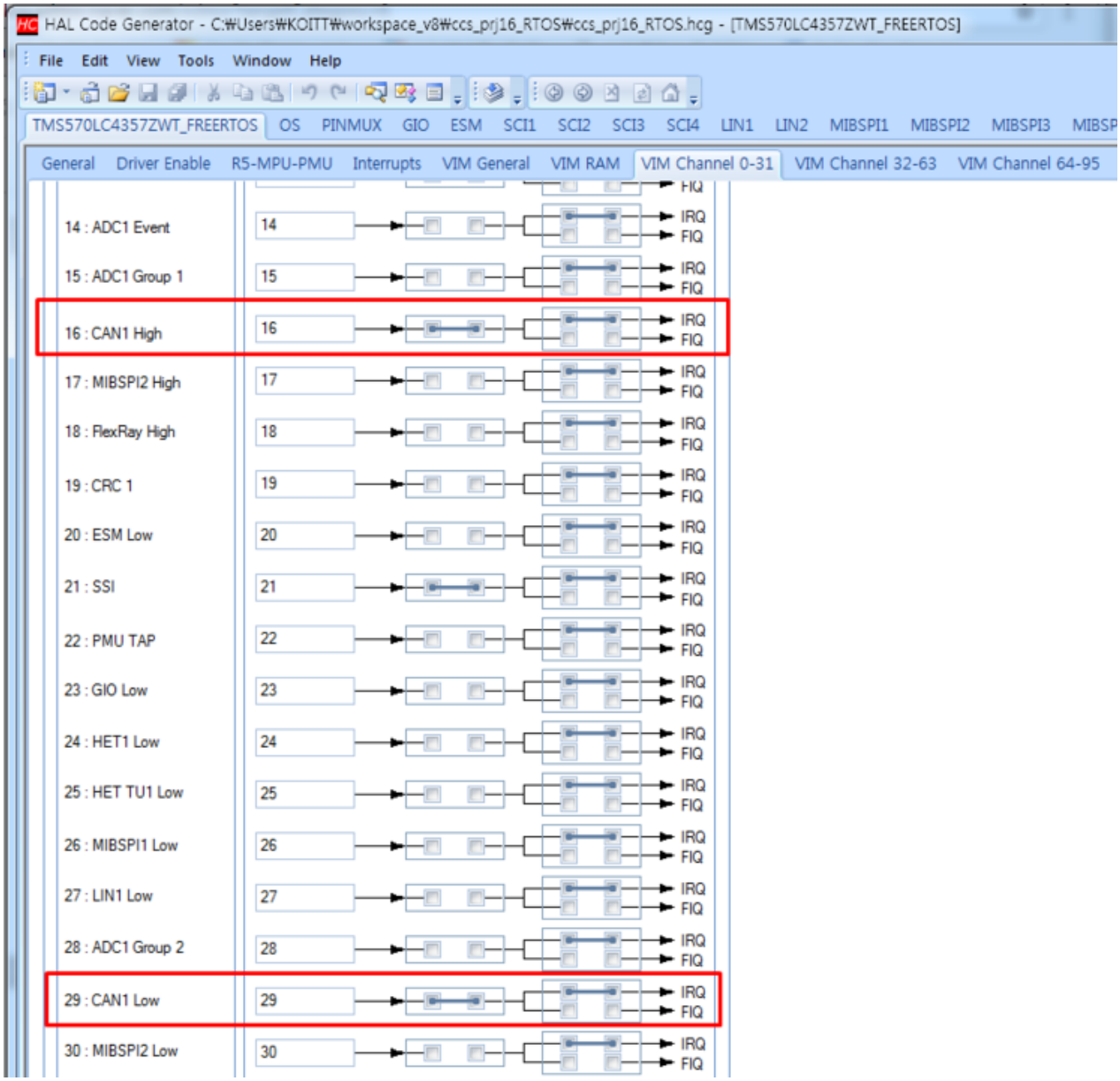

HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj16_RTOS₩ccs_prj16_RTOS.hcg - [OS]
File Edit View Tools Window Help
Start Page TMS570LC4357ZWT_FREERTOS OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2
General
Configuration
Configuration options will set macros in FreeRTOSConfig.h
Use Task Preemption
Use Idle Hook
Use Tick Hook
Use Co-Routines
Use Trace Facility
Use Mutexes
Use Recursive Mutexes
Use Counting Semaphores
Idle Task Should Yeild
Use Stack Overflow Hook
Use Verbose Stack Checking
Use Timers
Generate Runtime Statistics
Use Malloc Failed Hook
Task Configuration
RTI Clock (Hz): 75000000
Tick Rate (Hz): 1000
Max Priorities: 5
Total Heap Size: 8192
Task Name Length: 16
Min Stack Size: 128
Coroutine Configuration
Coroutine Priorities: 2
Timers Configuration
Timer Task Priority: 0
Queue Length: 0
Stack Size: 0

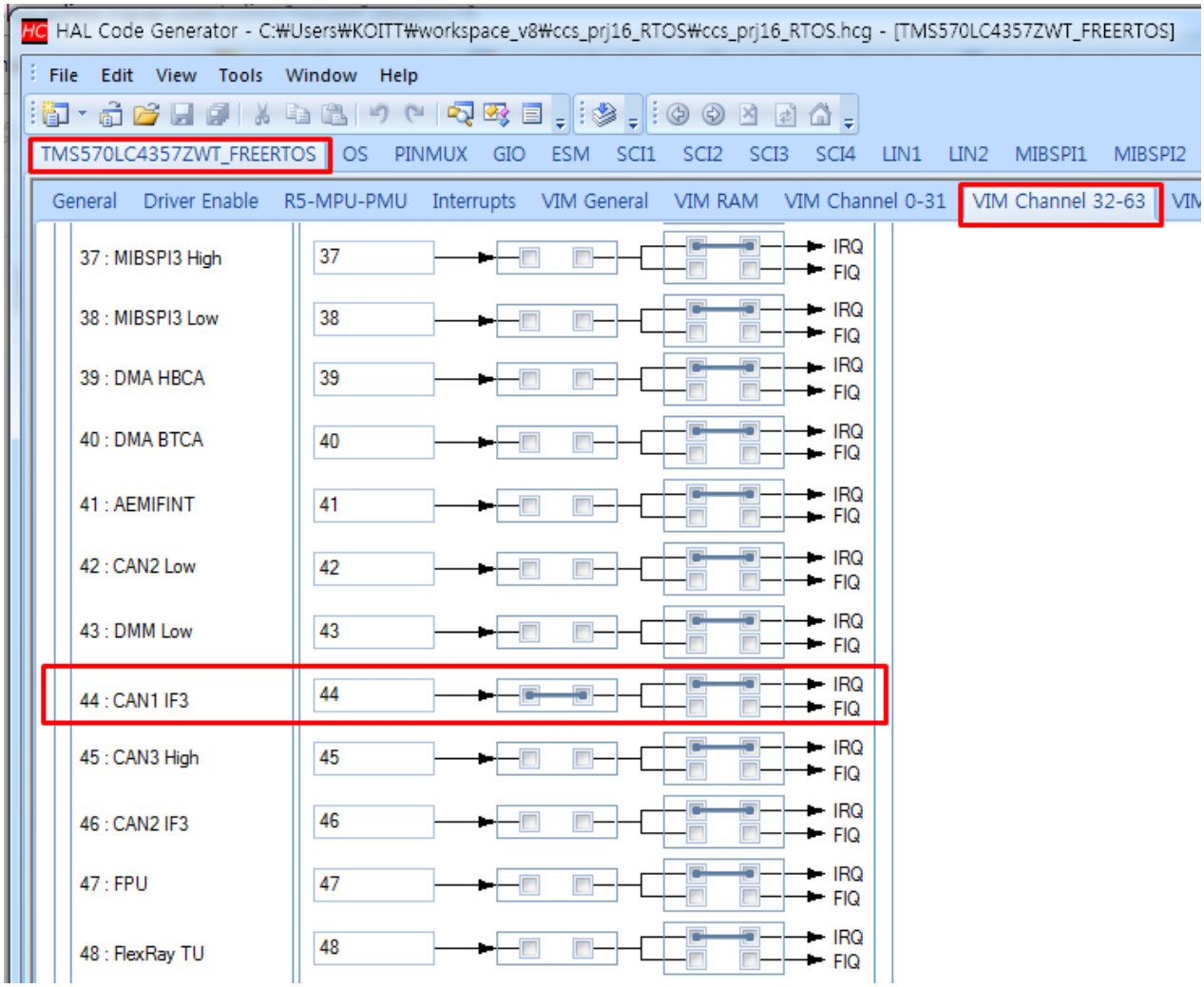
여기서 보면 RTI 0가 저절로 세팅 되어 있는 것을 알 수 있다. 그 이유는 RTOS에서 RTI를 써서 멀티태스킹처럼 돌아가게 구현을 했기 때문이다.

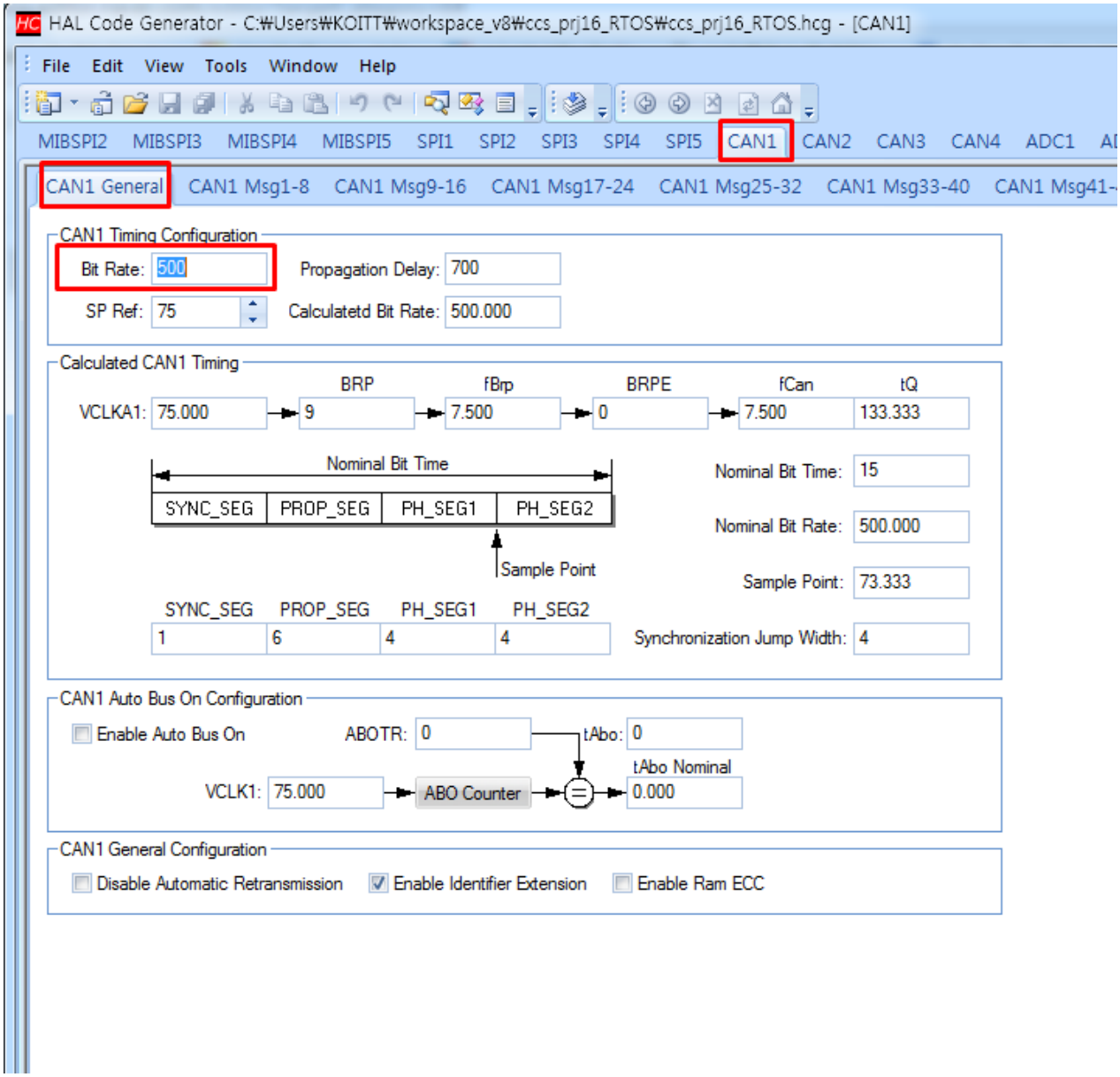


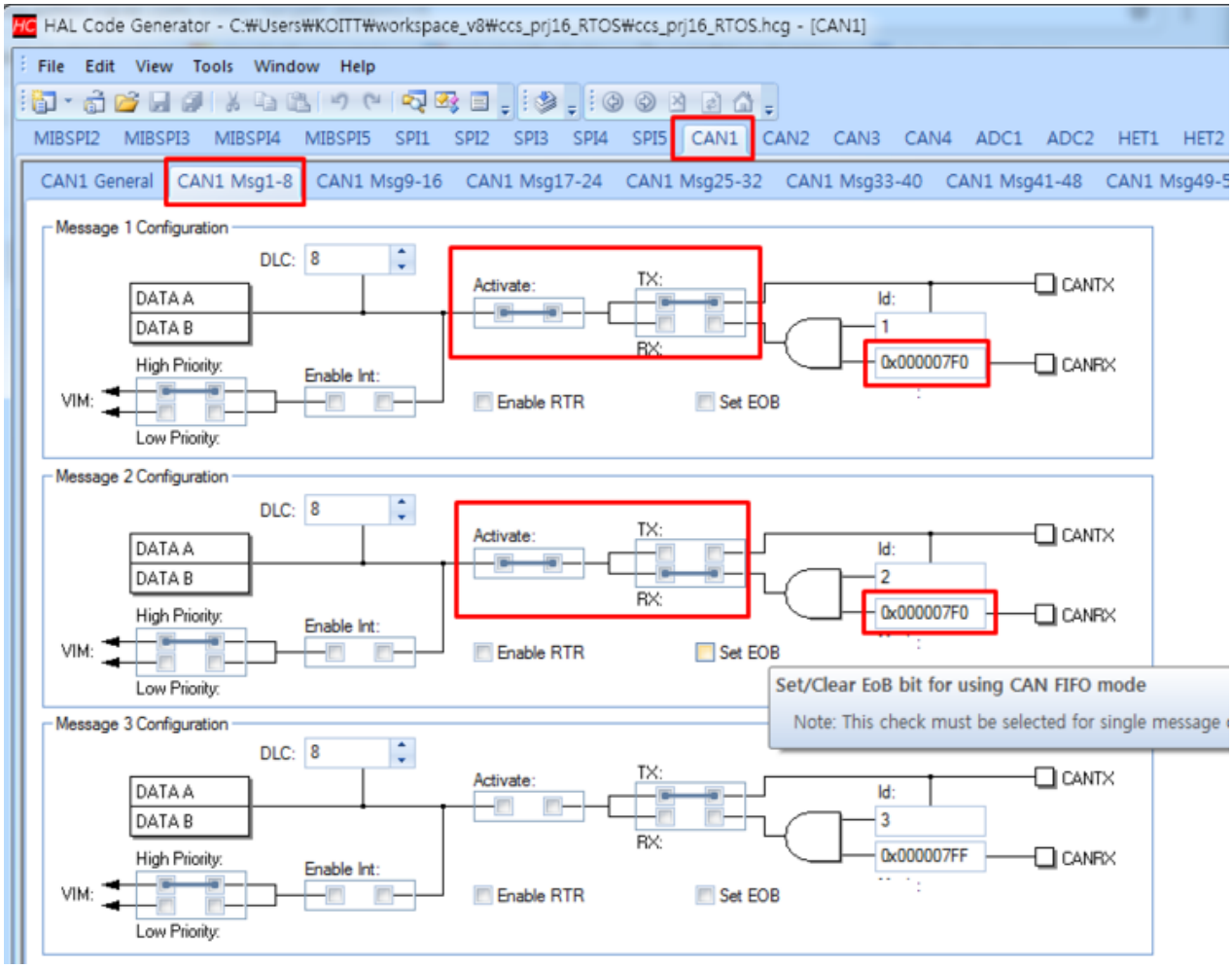



HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj16_RTOS₩ccs_prj16_RTOS.hcg - [TMS570LC435
File Edit View Tools Window Help
TMS570LC4357ZWT_FREERTOS OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2
General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31
VIM Channel 0-31 Configuration
Interrupt Assignment
CHANMAP0-CHANMAP31
0 : ESM High
1 : Reserved
2 : RTI Compare 0
3 : RTI Compare 1
4 : RTI Compare 2
5 : RTI Compare 3
6 : RTI Overflow 0
7 : RTI Overflow 1
8 : RTI Timebase 1
9 : GIO High
IRQ
FIQ


HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj16_RTOS₩ccs_prj16_RTOS.hcg - [TMS570LC4357ZWT_FREERTOS]
File Edit View Tools Window Help
TMS570LC4357ZWT_FREERTOS OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3
General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31 VIM Channel 32-63 VIM Channel 64-95
14 : ADC1 Event
15 : ADC1 Group 1
16 : CAN1 High
17 : MIBSPI2 High
18 : FlexRay High
19 : CRC 1
20 : ESM Low
21 : SSI
22 : PMU TAP
23 : GIO Low
24 : HET1 Low
25 : HET TU1 Low
26 : MIBSPI1 Low
27 : LIN1 Low
28 : ADC1 Group 2
29 : CAN1 Low
30 : MIBSPI2 Low
IRQ
FIQ


HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj16_RTOS₩ccs_prj16_RTOS.hcg - [TMS570LC4357ZWT_FREERTOS]
File Edit View Tools Window Help
TMS570LC4357ZWT_FREERTOS OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2
General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31 VIM Channel 32-63 VIM
37 : MIBSPI3 High 37 IRQ FIQ
38 : MIBSPI3 Low 38 IRQ FIQ
39 : DMA HBCA 39 IRQ FIQ
40 : DMA BTCA 40 IRQ FIQ
41 : AEMIFINT 41 IRQ FIQ
42 : CAN2 Low 42 IRQ FIQ
43 : DMM Low 43 IRQ FIQ
44 : CAN1 IF3 44 IRQ FIQ
45 : CAN3 High 45 IRQ FIQ
46 : CAN2 IF3 46 IRQ FIQ
47 : FPU 47 IRQ FIQ
48 : FlexRay TU 48 IRQ FIQ


HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj16_RTOS₩ccs_prj16_RTOS.hcg - [CAN1]
File Edit View Tools Window Help
MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5 SPI1 SPI2 SPI3 SPI4 SPI5 CAN1 CAN2 CAN3 CAN4 ADC1
CAN1 General CAN1 Msg1-8 CAN1 Msg9-16 CAN1 Msg17-24 CAN1 Msg25-32 CAN1 Msg33-40 CAN1 Msg41-
CAN1 Timing Configuration
Bit Rate: 500
Propagation Delay: 700
SP Ref: 75
Calculatetd Bit Rate: 500.000
Calculated CAN1 Timing
BRP fBrp BRPE fCan tQ
VCLKA1: 75.000 9 7.500 0 7.500 133.333
Nominal Bit Time
SYNC_SEG PROP_SEG PH_SEG1 PH_SEG2
Sample Point
Nominal Bit Time: 15
Nominal Bit Rate: 500.000
Sample Point: 73.333
SYNC_SEG PROP_SEG PH_SEG1 PH_SEG2
1 6 4 4
Synchronization Jump Width: 4
CAN1 Auto Bus On Configuration
Enable Auto Bus On
ABOTR: 0
tAbo: 0
tAbo Nominal
VCLK1: 75.000
ABO Counter
0.000
CAN1 General Configuration
Disable Automatic Retransmission
Enable Identifier Extension
Enable Ram ECC



F5를 눌러 code generation을 해준다.

〈Program code〉

- 밑에 프로그램을 보면 for문이나 while문이 많은 것을 볼 수가 있다.

1. vTask1, vTask2, vTask3 함수들 안에 있는 for문이 들어 있지 않으면 함수가 1번 동작하고 멈춰 버린다고 생각하면 된다. 그렇기 때문에 필수적으로 필요하다.(실험해 보아야함)

2. main 문 안에 있는 xTaskCreate 안에도 while문이 있는 것을 볼 수 있는데 이렇게 하면 while이 계속 동작 하고 있어 나중에 프로그램 동작에 많은 부담을 준다. 그런데도 while문이 있는 이유는 Task가 생성되지 않았을 예외경우를 처리 하기 위해서 이다.
(마치 컴퓨터에서 동영상을 여러개 튼것과 같은 것이다.)

3. main에 들어가는 while 문은 필수적인 녀석이다. 무조건 적으로 있어야 동작을 한다.

〈RTOS 를 하면서 모르겠는 부분이 있으면 이 사이트에 들어가서 확인한다.〉
https://www.freertos.org/FreeRTOS_Support_Forum_Archive/September_2013/freertos_Concerns_about_the_atomicity_of_vTaskSuspendAll_d165e9c3j.html


```
#include <FreeRTOS.h>
#include <FreeRTOSConfig.h>
#include <HL_can.h>
```

```
#include <HL_gio.h>
#include <HL_hal_stdtypes.h>
#include <HL_reg_sci.h>
#include <HL_sci.h>
#include <os_mpu_wrappers.h>
#include <os_projdefs.h>
#include <os_task.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define D_COUNT 8
#define D_SIZE 8

xTaskHandle xTask1Handle;
xTaskHandle xTask2Handle;
xTaskHandle xTask3Handle;

uint32 cnt = 0;
uint32 error = 0;
uint32 tx_done = 0;

uint8 msg[] = { 0 };
uint8 tx_data[D_COUNT] = { 1, 2, 3, 4, 4, 3, 2, 1 };
uint8 rx_data[D_COUNT] = { 0 };

void send_data(sciBASE_t* sci, uint8* msg, int length);

// 하나 하나가 메인이 라고 보면 된다. Task 마다 for문이 있는데 안 넣어주면 한번 동작을 하고 멈춘다.
// 함수의 들어가는 파라미터는 create 할때 지정해 준다?
void vTask1(void *pbParameters) {
    for (;;) {
        gioSetBit(gioPORTA, 0, gioGetBit(gioPORTA, 0) ^ 1);
        // vTaskDelay는 단위가 ms이다. halcogen에서 tick을 설정한 것에 따라서 기준이 달라진다.
        // 내부를 분석을 하면 interrupt를 끄지 않고 task만 설정한 시간 만큼 중단해 놓고 시간이 지나면 다시 task가 동작하도록 하는 동
        vTaskDelay(1000);
    }
}

void vTask2(void *pbParameters) {
    uint8 msg[32] = { 'T', 'a', 's', 'k', '2', '\r', '\n' };
    for (;;){
        send_data(sciREG1, msg, strlen((const char *) msg));
        canTransmit(canREG1, canMESSAGE_BOX1, (const uint8 *) &tx_data[0]);
        vTaskDelay(1000);
    }
}

void vTask3(void *pbParameters) {
    uint8 msg[32] = { 'T', 'a', 's', 'k', '3', '\r', '\n' };
    for (;;){
        send_data(sciREG1, msg, strlen((const char *) msg));
        vTaskDelay(1000);
    }
}
```

```
void main(void) {
    int i;

    gioInit();
    sciInit();
    canInit();

    gioSetDirection(gioPORTA, 1);

    // start message
    sciSend(sciREG1, 7, "start\n\r");

    //이 기능은 오류 수준에 대한 경고 기능을 활성화 합니다.
    canEnableErrorNotification(canREG1);

    /*
     *  BaseType_t xTaskCreate( TaskFunction_t pxTaskCode,
                            const char * const pcName,
                            const uint16_t usStackDepth,
                            void * const pvParameters,
                            UBaseType_t uxPriority,
                            TaskHandle_t * const pxCreatedTask )
                            /*lint !e971 Unqualified char types are allowed for strings and single characters only. */


    // 그냥하면 테스크 실패가 있을 수가 있어서 IF를 썼다. 중간에 보이는 파라미터 1은 우선 순위를 나타낸다.
    // xTaskCreate( 함수이름 , 테스크이름, 스텍의 크기, 만들어진 함수 인자들이 들어간다 (파라미터가 뭐가 들어가는지), 우선순위, 스텍이
    if (xTaskCreate(vTask1, "Task1", configMINIMAL_STACK_SIZE, NULL, 1, &xTask1Handle) != pdTRUE) {
        while (1) {
            if (xTaskCreate(vTask1, "Task1", configMINIMAL_STACK_SIZE, NULL, 1, &xTask1Handle) == pdTRUE)
                break;
        }
    }

    if (xTaskCreate(vTask2, "Task2", configMINIMAL_STACK_SIZE, NULL, 1, &xTask2Handle) != pdTRUE) {
        while (1) {
            if (xTaskCreate(vTask2, "Task2", configMINIMAL_STACK_SIZE, NULL, 1, &xTask2Handle) == pdTRUE)
                break;
        }
    }

    if (xTaskCreate(vTask3, "Task3", configMINIMAL_STACK_SIZE, NULL, 1, &xTask3Handle) != pdTRUE) {
        while (1) {
            if (xTaskCreate(vTask3, "Task3", configMINIMAL_STACK_SIZE, NULL, 1, &xTask3Handle) == pdTRUE)
                break;
        }
    }

    // 위에서 Create로  만들어 놓은 Task들을  우선순위를 고려한 방식으로 스케쥴링 되어 동작하게 한다.
    vTaskStartScheduler();
    while (1)
        ;
}
void send_data(sciBASE_t* sci, uint8* msg, int length) {
```

```
    int i;
    for (i = 0; i < length; i++)
        sciSendByte(sci, msg[i]);
}
```

<코드 설명>

vTaskDelay(1000) : _(현재 tick이 1000Hz이기 때문에 1ms 단위로 setting 되어 있다.)
내부를 분석을 하면 interrupt를 끄지 않고 task만 설정한 시간 만큼 중단해 놓고 시간(1000ms)이 지나면 다시 task가 동작하도록 하는 동작방식이다.

if (xTaskCreate(vTask1, "Task1", configMINIMAL_STACK_SIZE, NULL, 1, &xTask1Handle) != pdTRUE) {
// 그냥하면 테스크 실패가 있을 수가 있어서 IF를 썼다. 중간에 보이는 파라미터 1은 우선 순위를 나타낸다. xTaskCreate(함수 이름 , 테스크이름, 스텍의 크기, 만들어진 함수 인자들이 들어간다 (파라미터가 뭐가 들어가는지), 우선순위, 스텍이 할당된 주소의 처음 위치)

void vTask2(void *pbParameters) :
하나 하나가 메인이 라고 보면 된다. Task 마다 for문이 있는데 안 넣어주면 한번 동작을 하고 멈춘다.
함수의 들어가는 파라미터는 create 할때 지정해 준다?

vTaskStartScheduler();
// 위에서 Create로 만들어 놓은 Task들을 우선순위를 고려한 방식으로 스케쥴링 되어 동작하게 한다.