TI DSP,MCU 및 Xilinux Zynq FPGA

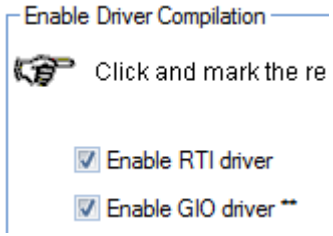
프로그래밍 전문가 과정

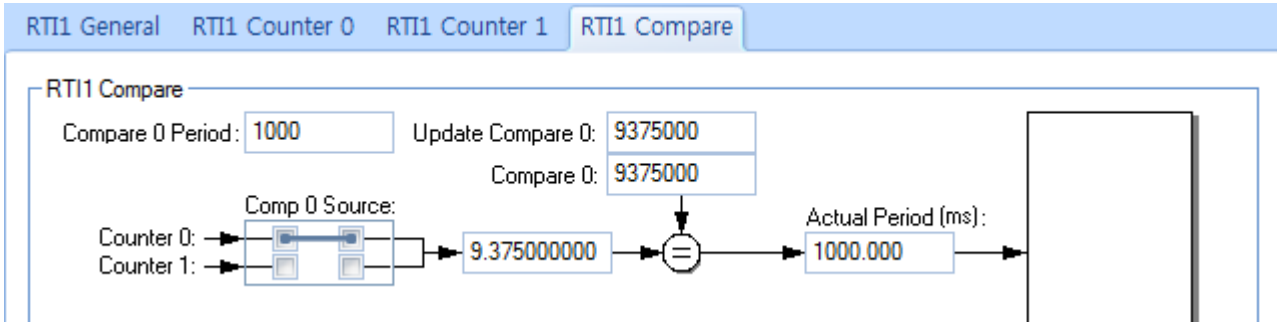
이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/5/15
수업일수	54 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

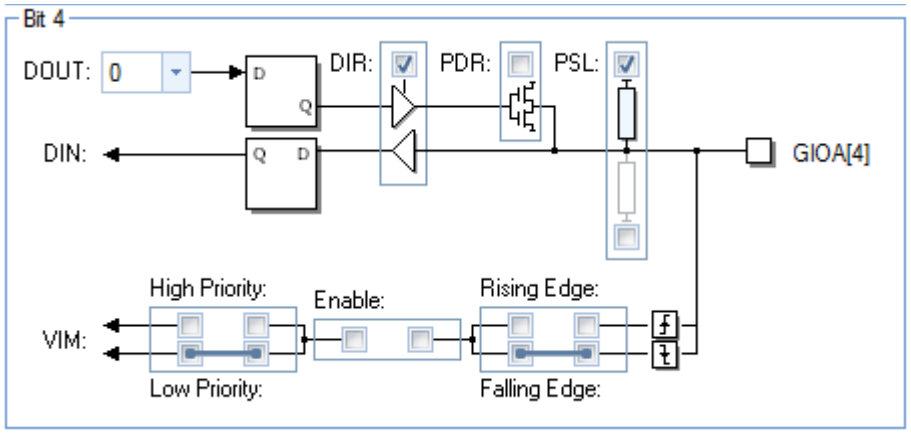
목차

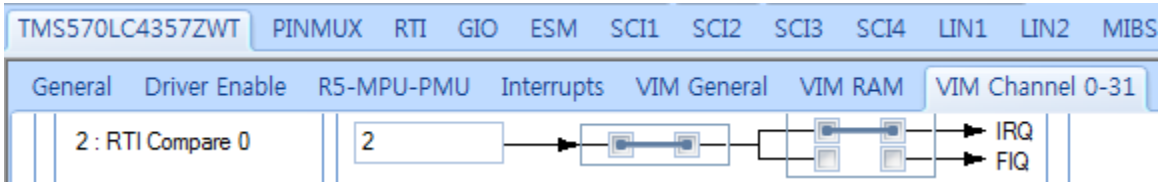
OC_circuit

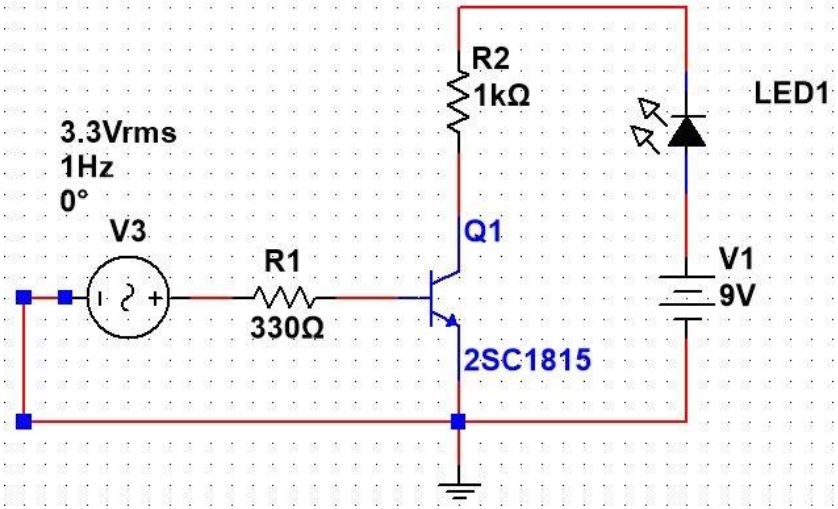
[HALcogen]설정











회로 구성

Open collector사용 이유

1. Open collector를 사용하지 않으면 항상 방전되어 소비전력이 커진다.

2. 유연한 인터럽트 동작을 위해, 하드웨어적으로만 회로를 구성하게 되면 정해진 동작밖에 하지 못하는데 소프트웨어적으로 인터럽트를

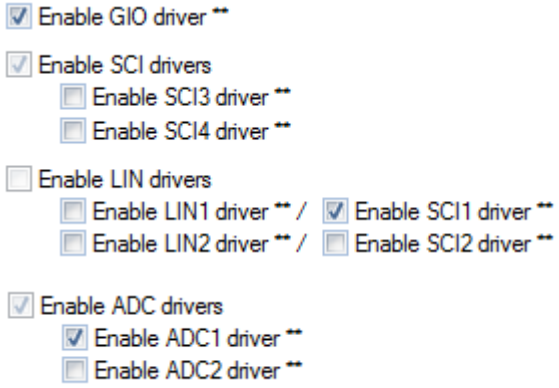
3. 두개의 전원으로 분리시킴

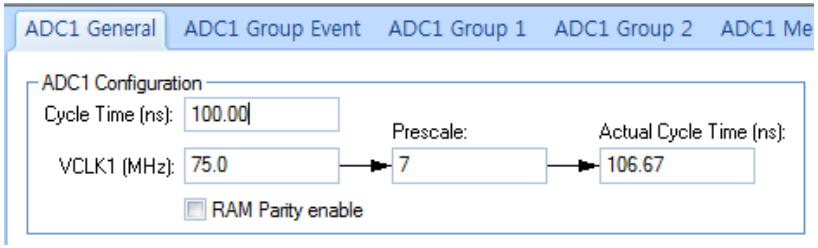
```
#include "HL_sys_common.h"
#include "HL_system.h"
```

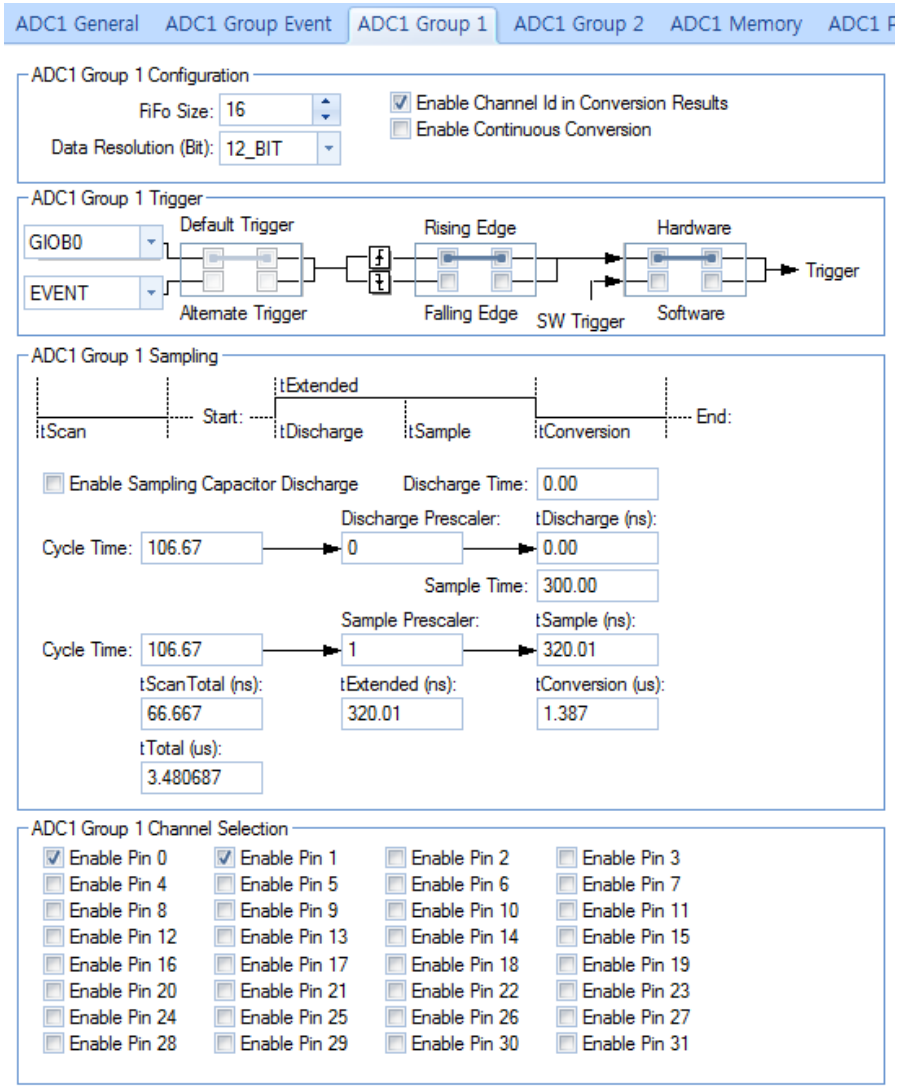
```
#include "HL_sys_core.h"
#include "HL_mibspi.h"
#include "HL_esm.h"
#include "HL_rti.h"
#include "HL_gio.h"
#include "HL_het.h"
#include <stdlib.h>
#include <time.h>

unsigned int i = 0;
int t_flag =0;

int main(void)
{
/* USER CODE BEGIN (3) */
    rtiInit();

    gioSetDirection(gioPORTA, 0xFFFFFFFF);

    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);

    _enable_IRQ_interrupt_();

    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);
        srand(time(NULL));

    while(1);
/* USER CODE END */

    return 0;
}
```

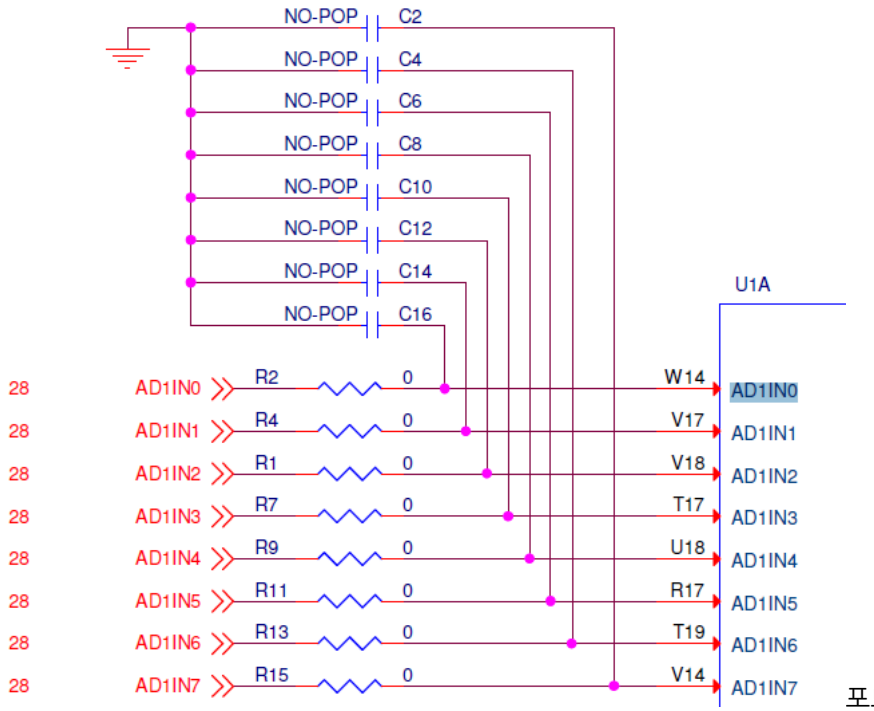
```
void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
	unsigned int val = 0xAA060021;
	unsigned int val2= 0xFFFFFFFF;
	unsigned int a[9] = {0,5,25,18,29,27,17,31};

	/*i++;
	if(i>8)
	{i =0;
	}*/
	i = rand() % 9;

	if(t_flag == 0)
	{
	    t_flag = 1;
	    gioSetPort(hetPORT1, gioGetPort(hetPORT1) & ~(val));

	}
	else{
	    gioSetPort(hetPORT1, gioGetPort(hetPORT1) | 1<<a[i]);
	    t_flag = 0;
	}


}
```

ADC_UART


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 ADC drivers
Enable ADC1 driver **
Enable ADC2 driver **


ADC1 General
ADC1 Group Event
ADC1 Group 1
ADC1 Group 2
ADC1 Me
ADC1 Configuration
Cycle Time (ns): 100.00
Prescale:
Actual Cycle Time (ns):
VCLK1 (MHz): 75.0
7
106.67
RAM Parity enable


ADC1 General
ADC1 Group Event
ADC1 Group 1
ADC1 Group 2
ADC1 Memory
ADC1 Configuration
FiFo Size: 16
Data Resolution (Bit): 12_BIT
Enable Channel Id in Conversion Results
Enable Continuous Conversion
ADC1 Group 1 Trigger
GIOB0
EVENT
Default Trigger
Alternate Trigger
Rising Edge
Falling Edge
Hardware
SW Trigger
Software
Trigger
ADC1 Group 1 Sampling
tScan
Start:
tExtended
tDischarge
tSample
tConversion
End:
Enable Sampling Capacitor Discharge
Discharge Time: 0.00
Discharge Prescaler:
tDischarge (ns):
Cycle Time: 106.67
0
0.00
Sample Time: 300.00
Sample Prescaler:
tSample (ns):
Cycle Time: 106.67
1
320.01
tScanTotal (ns):
tExtended (ns):
tConversion (us):
66.667
320.01
1.387
tTotal (us):
3.480687
ADC1 Group 1 Channel Selection
Enable Pin 0
Enable Pin 1
Enable Pin 2
Enable Pin 3
Enable Pin 4
Enable Pin 5
Enable Pin 6
Enable Pin 7
Enable Pin 8
Enable Pin 9
Enable Pin 10
Enable Pin 11
Enable Pin 12
Enable Pin 13
Enable Pin 14
Enable Pin 15
Enable Pin 16
Enable Pin 17
Enable Pin 18
Enable Pin 19
Enable Pin 20
Enable Pin 21
Enable Pin 22
Enable Pin 23
Enable Pin 24
Enable Pin 25
Enable Pin 26
Enable Pin 27
Enable Pin 28
Enable Pin 29
Enable Pin 30
Enable Pin 31



포트연결 AD1N0

```
#include "HL_system.h"
#include "HL_sys_common.h"

#include "HL_esm.h"
#include "HL_adc.h"
#include "HL_sci.h"
#include "Hl_gio.h"
```

```
#define TSIZE1   12
uint8 TEXT1[TSIZE1] = {'₩r', '₩n', '|', 't', 'C', 'H', '.', 'I','D', '=', '0', 'x'};
#define TSIZE2   9
uint8    TEXT2[TSIZE2] = {'₩t', 'V', 'A', 'L', 'U', 'E', '=', '0', 'x'};

adcData_t adc_data[2];
void sciDisplayText(sciBASE_t *sci, uint8 *text, uint32 length);
void sciDisplayData(sciBASE_t *sci, uint8 *text, uint32 length);
void wait(uint32 time);

int main(void)
{

    uint32 ch_count = 0;
    uint32 id = 0;
    uint32 value = 0;

    gioInit(); //HALcogen설정에 맞춘 gio코드
    gioSetDirection(gioPORTB, 1);

    sciInit(); //HALcogen설정에 맞춘 sci코드

    adcInit(); //HALcogen설정에 맞춘 adc코드
    adcStartConversion(adcREG1, adcGROUP1);

    for(;;)
    {
        gioSetBit(gioPORTB, 0, 1); //시작
        while((adcIsConversionComplete(adcREG1, adcGROUP1)) == 0) //샘플링하는 구간
            ;
        ch_count = adcGetData(adcREG1, adcGROUP1, &adc_data[0]);
```

```
        id = adc_data[0].id; // 데이터 받은 핀
        value = adc_data[0].value; //샘플링 값

        gioSetBit(gioPORTB, 0, 0);//샘플링 끝

        sciDisplayText(sciREG1, &TEXT1[0], TSIZE1);
        sciDisplayData(sciREG1, (uint8 *)&id, 4);
        sciDisplayText(sciREG1, &TEXT2[0], TSIZE2);
        sciDisplayData(sciREG1, (uint8 *)&value, 4);

        /* pin1은 사용하지 않으므로 주석처리
        id = adc_data[1].id;
        value = adc_data[1].value;

        sciDisplayText(sciREG1, &TEXT1[0], TSIZE1);
        sciDisplayData(sciREG1, (uint8 *)&id, 4);
        sciDisplayText(sciREG1, &TEXT2[0], TSIZE2);
        sciDisplayData(sciREG1, (uint8 *)&value, 4);

        wait(0xFFFFF);
        */
    }

    return 0;
}

void sciDisplayData(sciBASE_t *sci, uint8 *text, uint32 length)
{
    uint8 txt = 0;
    uint8 txt1 = 0;
```

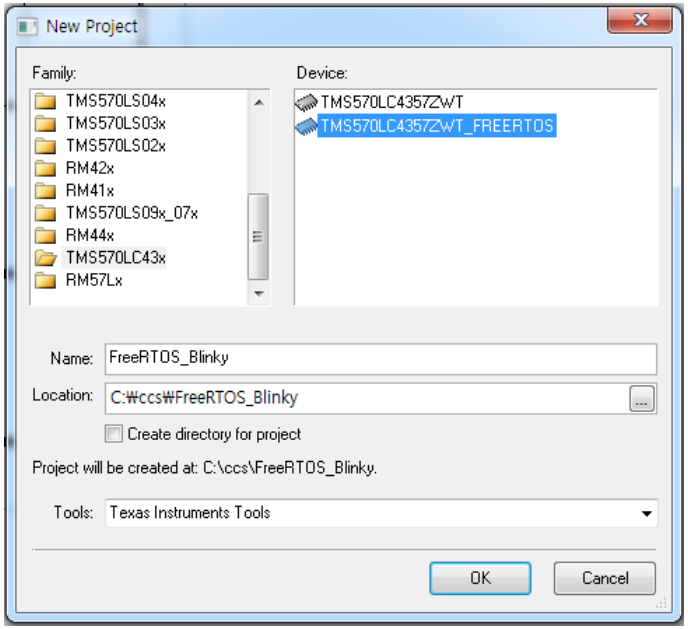
```
#if ((__little_endian__ == 1) || (__LITTLE_ENDIAN__ == 1))
    text = text + (length -1);
#endif

    while(length--)
    {
#if((__little_endian__ == 1) || (__LITTLE_ENDIAN__ == 1))
        txt = *text--;
#else
        txt = *text++;
#endif
        txt1 = txt;

        txt &= ~(0xF0);
        txt1 &= ~(0x0F);
        txt1 = txt1>>4;

        if(txt<=0x9)
        {
            txt += 0x30;
        }
        else if(txt > 0x9 && txt < 0xF)
        {
            txt = 0x37;
        }
        else
        {
            txt = 0x30;
        }
        if(txt1 <= 0x9)
        {
            txt1 += 0x30;
```

```
        }
        else if((txt1 > 0x9) && (txt1 <= 0xF))
        {
            txt += 0x37;
        }
        else
        {
            txt1 = 0x30;
        }

        while((sciREG1->FLR & 0x4) == 4)
            ;
        sciSendByte(sciREG1, txt1);

        while((sciREG1->FLR & 0x4) == 4)
            ;
        sciSendByte(sciREG1, txt);
    }
}

void sciDisplayText(sciBASE_t *sci, uint8 *text, uint32 length)
{
    while(length--)
    {
        while((sciREG1->FLR & 0x4) == 4)
            ;
        sciSendByte(sciREG1, *text++);
    }
}

void wait(uint32 time)
{
```

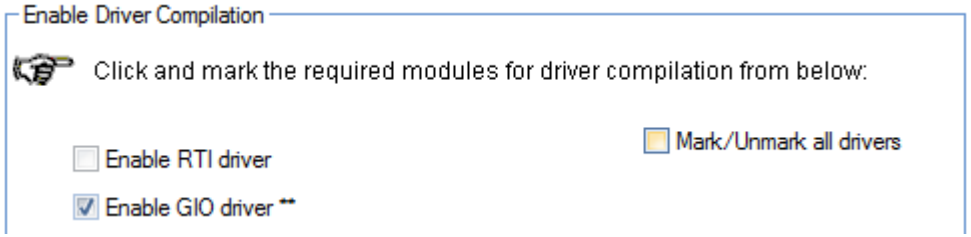
```
    while(time)
    {
        time--;
    }
}
```

FreeRTOS_Blinky

[HALcogen] 설정

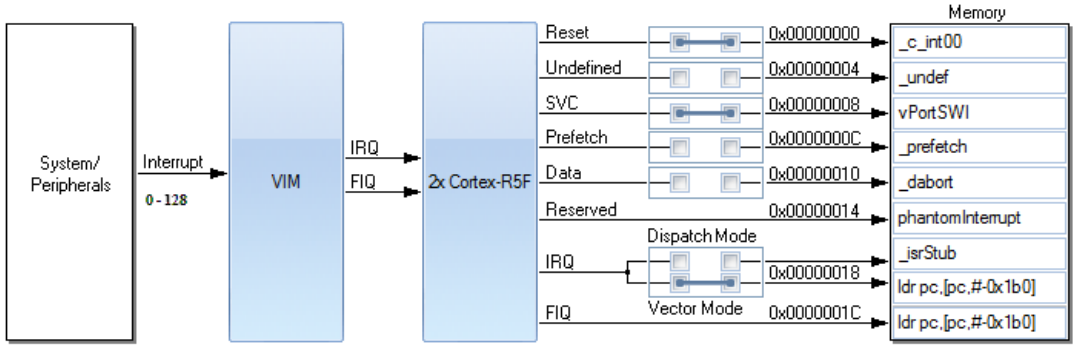


project설정



General | Driver Enable | R5-MPU-PMU | Interrupts | VIM General | VIM RAM | VIM Channel 0-31 | VIM Channel 32-63 | VIM Cl

Interrupt Configuration

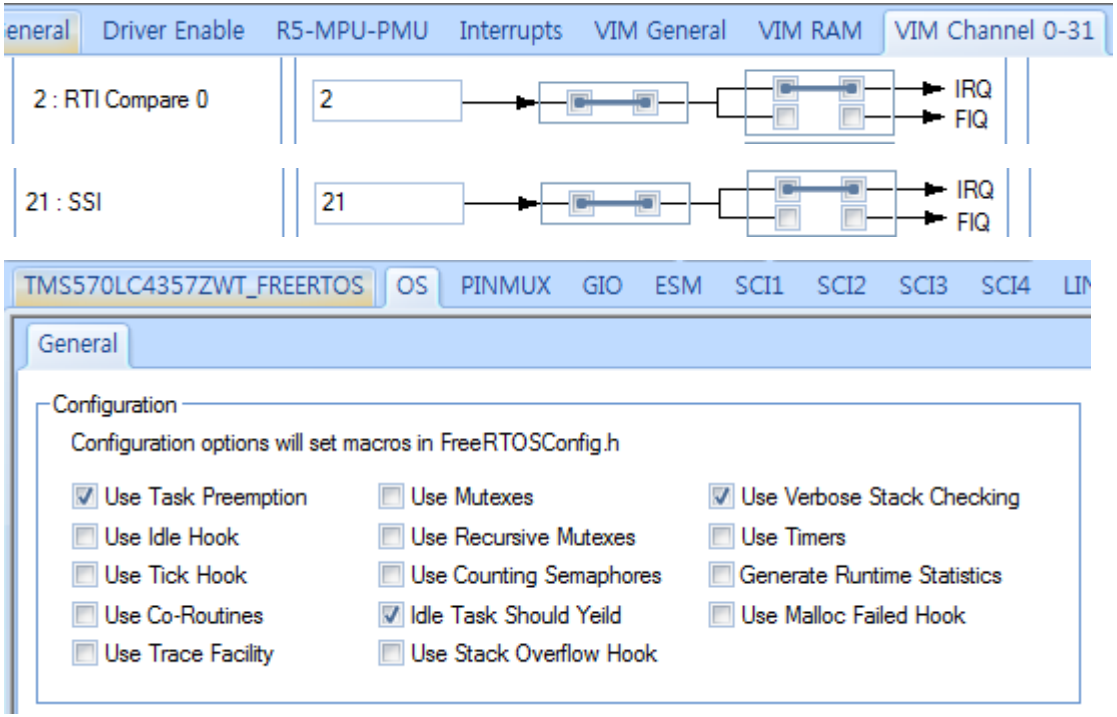


General | Driver Enable | R5-MPU-PMU | Interrupts | VIM General | VIM RAM | VIM

VIM RAM : ISR Assignments

Base Address: 0xFFF82000 ☑ VIM RAM ECC enable

0x00000000:PH	phantomInterrupt	0x00000040:15	adc1Group1Interrupt	0x0000	
0x00000004:00	esmHighInterrupt	0x00000044:16	can1HighLevelInterrupt	0x0000	
0x00000008:01	phantomInterrupt	0x00000048:17	spi2HighLevelInterrupt	0x0000	
0x0000000C:02	vPortPreemptiveTick	0x0000004C:18	phantomInterrupt	0x0000	
0x00000010:03	phantomInterrupt	0x00000050:19	crcInterrupt	0x0000	
0x00000014:04	phantomInterrupt	0x00000054:20	esmLowInterrupt	0x0000	
0x00000018:05	phantomInterrupt	0x00000058:21	vPortYeildWithinAPI	0x0000	
0x0000001C:06	phantomInterrupt	0x0000005C:22	phantomInterrupt	0x0000	
0x00000020:07	phantomInterrupt	0x00000060:23	gioLowLevelInterrupt	0x0000	
0x00000024:08	phantomInterrupt	0x00000064:24	het1LowLevelInterrupt	0x0000	
0x00000028:09	gioHighLevelInterrupt	0x00000068:25	phantomInterrupt	0x0000	
0x0000002C:10	het1HighLevelInterrupt	0x0000006C:26	mibspi1LowLevelInterrup	0x0000	
0x00000030:11	phantomInterrupt	0x00000070:27	lin1LowLevelInterrupt	0x0000	
0x00000034:12	mibspi1HighLevelInterrup	0x00000074:28	adc1Group2Interrupt	0x0000	
0x00000038:13	lin1HighLevelInterrupt	0x00000078:29	can1LowLevelInterrupt	0x0000	
0x0000003C:14	adc1Group0Interrupt	0x0000007C:30	mibspi2LowLevelInterrup	0x0000	


eneral
Driver Enable
R5-MPU-PMU
Interrupts
VIM General
VIM RAM
VIM Channel 0-31
2 : RTI Compare 0
2
IRQ
FIQ
21 : SSI
21
IRQ
FIQ
TMS570LC4357ZWT_FREERTOS
OS
PINMUX
GIO
ESM
SCI1
SCI2
SCI3
SCI4
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

```
#include "HL_sys_common.h"
#include "FreeRTOS.h"
#include "os_task.h"
#include "HL_het.h"
#include "HL_gio.h"

xTaskHandle xTask1Handle;

void vTask1(void *pvParameters) //태스크 생성 성공하면 자식 프로세스가 실행
{
    for(;;)
    {
        gioSetBit(hetPORT1, 17, gioGetBit(hetPORT1, 17) ^ 1 );
        vTaskDelay(100); //지연
    }
}
int main(void)
{
    gioSetDirection(hetPORT1, 0xFFFFFFFF); //hetPORT1사용

    if(xTaskCreate(vTask1, "Task1",configMINIMAL_STACK_SIZE, NULL, 1, &xTask1Handle) != pdTRUE) //태스크 생성 실패시 if문 만족
      /*RTOS의 태스크를 생성(fork와 같은것)
      Task1이라는 태스크가 생성됨.
      최소한의 스택 사이즈를 configMINIMAL_STACK_SIZE*/
    {
        while(1);
    }
    vTaskStartScheduler(); //태스크 생성 성공 시 부모 프로세스가 실행

    return 0;
}
```