

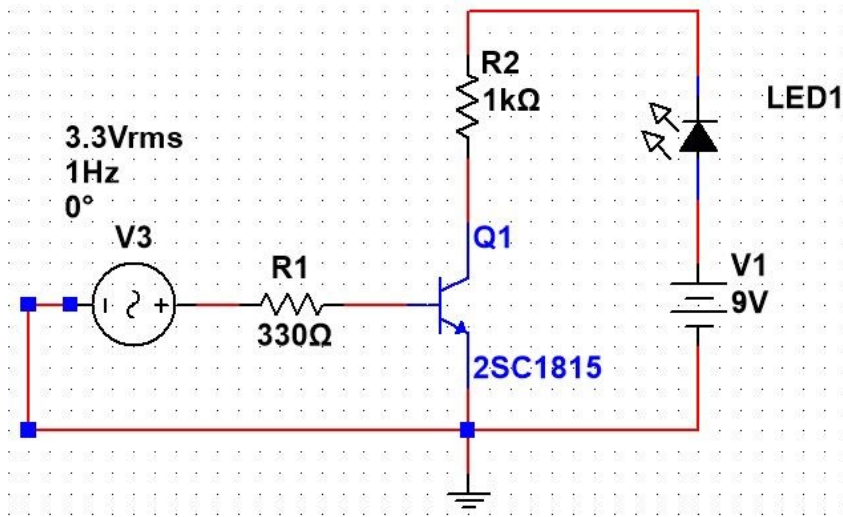
TI DSP, MCU, Xilinx Zynq FPGA ***기반의 프로그래밍 전문가 과정***

<펌웨어 프로그래밍>
2018.05.15 – 54일차

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – 안상재
sangjae2015@naver.com

1. Open Collector (O.C) 회로 제어하기



1-1. 회로 분석

- 1) V3의 구형파(0V, 3.3V의 PWM 신호)에서 3.3V가 입력되면, Q1 TR의 베이스가 ON 되고, 전류가 흐른다.
- 2) Q1의 베이스에 일정 전류 이상이 흐르면 컬렉터와 이미터 사이가 도통된다.
- 3) V1 전원에서 컬렉터 -> 이미터로 전류가 흐른다.
- 4) LED1가 켜진다.
- 5) V3의 구형파에서 0V가 입력이 되면 Q1이 OFF 되고 LED1은 꺼진다.
- 6) V3의 구형파가 입력이 되면 LED1이 반짝반짝 거린다.

1-2. 소스 코드

```
#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"

void main(void)
{
    rtiInit();
    gioInit();

    gioSetDirection(gioPORTA, 0x00000001); // gio PORTA의 0번핀을 출력으로 설정

    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0); // rti 인터럽트를 enable 시킴
    _enable_IRQ_interrupt();
    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0); // counter 를 시작함

    srand(time(NULL));

    while(1);
}
```

```
// rti 핸들러 등록
void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
    gpioSetPort(gioPORTA, gpioGetPort(gioPORTA) ^ 0x00000001);    /* rti 인터럽트가 발생할 때마다
                                                                    PORTA의 0번 핀을 토글 */
}
```

2. ADC_UART

- 조도 센서에서 읽은 아날로그 값을 ADC 를 통해 디지털 값으로 만들고, 디지털 값을 SCI 를 통해 PC 의 터미널창에 출력함.
- 조도 센서에 비추는 빛의 밝기에 따라 값이 변함.

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

#include "HL_sci.h"
#include "HL_esm.h"
#include "HL_adc.h"
#include "HL_gio.h"

#define TSIZE1 12
uint8 TEXT1[TSIZE1]={"\r","\n","|","\t",'C','H',' ','T','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;

    gpioInit();
    gpioSetDirection(gioPORTB, 0xFF);
    gpioSetDirection(gioPORTA, 0xFFFFFFFF);

    sciInit();

    adcInit();
    adcStartConversion(adcREG1, adcGROUP1);

    while(1){
        gpioSetBit(gioPORTB, 0, 1);
        while((adcIsConversionComplete(adcREG1, adcGROUP1))==0);

        ch_count = adcGetData(adcREG1, adcGROUP1, &adc_data[0]);

        id = adc_data[0].id;
        value = adc_data[0].value;
```

```

gioSetBit(gioPORTB, 0, 0);

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

if(value > 0xD00){
    gioSetBit(gioPORTA, 4, 1);
}
else{
    gioSetBit(gioPORTA, 4, 0);
}
wait(0xFFFFF);
}
}

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

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)){
    txt1 += 0x37;
}
else{
    txt1 = 0x30;
}

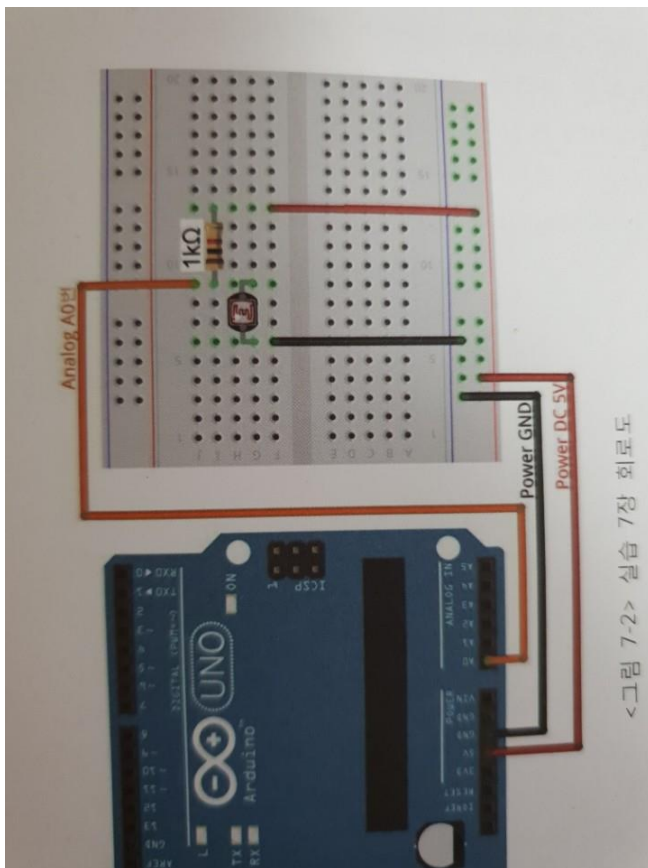
while((sciREG1->FLR & 0x4)==4);
sciSendByte(sciREG1, txt1);

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

void wait(uint32 time){
    int i;
    for(i=0; i<time; i++);
}

```

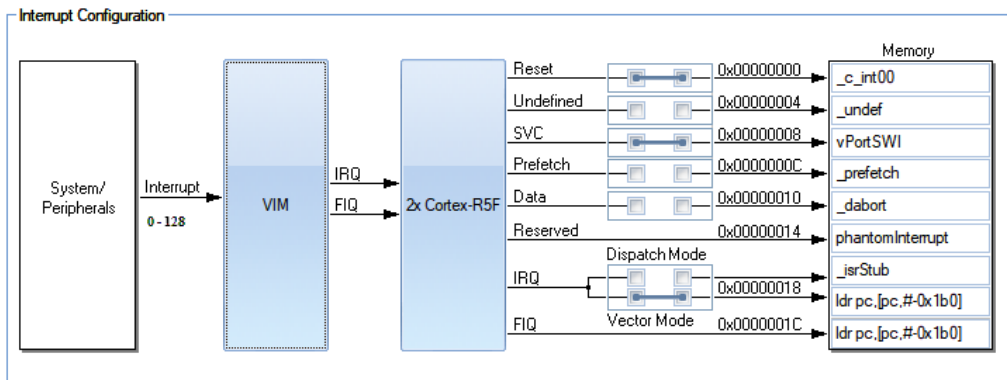
2-1. 회로도



<그림 7-2> 실습 7장 회로도

3. FreeRtos

3-1. FreeRtos 설정 (HAL Code Generator)



VIM RAM : ISR Assignments

Base Address: 0xFFFF82000 ☒ VIM RAM ECC enable

0x00000000:PH	phantomInterrupt	0x00000040:15	adc1Group1Interrupt
0x00000004:00	esmHighInterrupt	0x00000044:16	can1HighLevelInterrupt
0x00000008:01	phantomInterrupt	0x00000048:17	spi2HighLevelInterrupt
0x0000000C:02	vPortPreemptive Tick	0x0000004C:18	phantomInterrupt
0x00000010:03	phantomInterrupt	0x00000050:19	crcInterrupt
0x00000014:04	phantomInterrupt	0x00000054:20	esmLowInterrupt
0x00000018:05	phantomInterrupt	0x00000058:21	vPortYeildWithinAPI
0x0000001C:06	phantomInterrupt	0x0000005C:22	phantomInterrupt
0x00000020:07	phantomInterrupt	0x00000060:23	gioLowLevelInterrupt
0x00000024:08	phantomInterrupt	0x00000064:24	het1LowLevelInterrupt
0x00000028:09	gioHighLevelInterrupt	0x00000068:25	phantomInterrupt
0x0000002C:10	het1HighLevelInterrupt	0x0000006C:26	mibspi1LowLevelIntemp
0x00000030:11	phantomInterrupt	0x00000070:27	lin1LowLevelInterrupt
0x00000034:12	mibspi1HighLevelInterrupt	0x00000074:28	adc1Group2Interrupt
0x00000038:13	lin1HighLevelInterrupt	0x00000078:29	can1LowLevelInterrupt
0x0000003C:14	adc1Group0Interrupt	0x0000007C:30	mibspi2LowLevelInterrupt

TMS570LC4357ZWT_FREERTOS OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN

General

Configuration

Configuration options will set macros in FreeRTOSConfig.h

☒ Use Task Preemption ☐ Use Mutexes ☒ Use Verbose Stack Checking

☐ Use Idle Hook ☐ Use Recursive Mutexes ☐ Use Timers

☐ Use Tick Hook ☐ Use Counting Semaphores ☐ Generate Runtime Statistics

☐ Use Co-Routines ☒ Idle Task Should Yield ☐ Use Malloc Failed Hook

☐ Use Trace Facility ☐ Use Stack Overflow 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

3-2. 소스 코드 분석

- 스레드 생성 (FREERTOS 의 API 사용)

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

xTaskHandle xTask1Handle;

void vTask1(void *pvParameters) // 자식 스레드는 LED 토글
{
    for(;;)
    {
        gioSetBit(hetPORT1, 17, gioGetBit(hetPORT1,17)^1);
        vTaskDelay(100);
    }
}

void main(void)
{
    gioSetDirection(hetPORT1, 0xFFFFFFFF);

    // 자식 스레드 생성
    if(xTaskCreate(vTask1, "Task1", configMINIMAL_STACK_SIZE, NULL,1,&xTask1Handle) != pdTRUE)
    {
        while(1); // 스레드 생성에 실패할 경우
    }

    vTaskStartScheduler(); // 스레드 우선순위 스케줄링

    while(1);
}
```