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

#54

2018.05.16

강사:Innova Lee(이 상훈)

학생: 김시윤

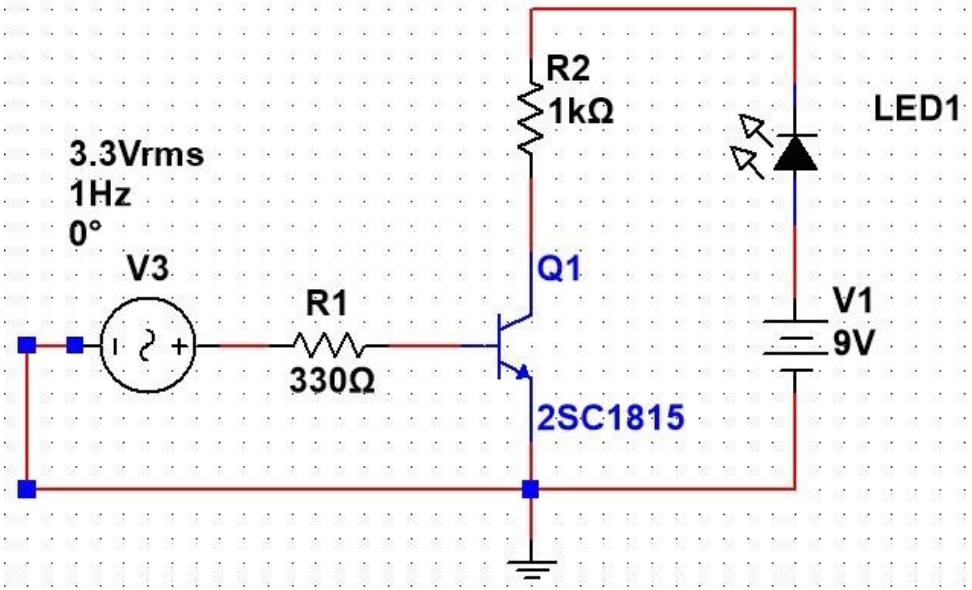
과제

OPEN_COLLECTOR_CIRCUIT

OPEN COLLECTOR 회로를 쓰는 이유!!

오픈 컬렉터 회로는 서로 다른 전압의 회로를 안정적으로 동작시키기 위해 사용한다.
MCU 와 모터의 관계로 예를들면 모터의 동작전압이 12V 일 때, MCU 핀의 전압은 3.3V 이기 때문에 모터의 동작 전압과 맞지 않는다. 하지만 MCU 로 모터를 통제 해야 하기 때문에 외부 전압을 끌고와 사용하기 위해 오픈컬렉터 방식을 이용한다.

오픈컬렉터 방식은 둘다 출력이 1 일때만 출력이 되기 때문에 wired AND 라고도 부른다.



OPEN_COLLECTOR 회로는 RTI_Blink 소스코드로 확인하였다.
5V 아두이노 전원으로 핀에서 나오는 전압의 통제를 받아 동작하는것을 확인하였다.

```
#include "HL_sys_common.h"
#include "HL_gio.h"
#include "HL_rti.h"

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

  gioSetDirection(gioPORTA, 0xffffffff);
  rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);
  gioSetPort(gioPORTA, 0xffffffff);

  _enable_IRQ_interrupt_();

  rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

  for(;;)
    ;

  return 0;
}

void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
  gioToggleBit(gioPORTA, 4);
}
```

```
#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','.','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();
  gioSetDirection(gioPORTB, 0xFF);
  gioSetDirection(gioPORTA, 0xFFFFFFFF);

  sciInit();
  adcInit();
  adcStartConversion(adcREG1, adcGROUP1);

  while(1){
    gioSetBit(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++);
}
```

ADC
조도센서의 아날로그 신호를 디지털신호로 변조해 값을 받아오는 작업을 했었다.
할코젠에서 샘플 타임까지 다 맞춰줘서 편했다…
12 비트 기준.
샘플링 주파수 10MHz
VCLK 클럭주파수 75MHz
1 클록에 7 번 샘플링을 한다.

우리는 조도센서에서 들어오는 아날로그 데이터를 샘플링 해서 그 떄의 값을 UART 로 확인하였다.

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

#include "HL_het.h"
#include "HL_gio.h"

xTaskHandle xTask1Handle;
xTaskHandle xTask2Handle;

void vTask1(void *pvParameters)
{
   for(;;)
   {
      gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0xAA060021);
      //gioSetBit(hetPORT1, 17, gioGetBit(hetPORT1, 17) ^ 1);
      vTaskDelay(100);

   }
}


void vTask2(void *pvParameters)
{
   for(;;)
   {
      gioSetBit(gioPORTA, 4, gioGetBit(gioPORTA, 4) ^ 1);
      vTaskDelay(100);
   }

}

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

if(xTaskCreate(vTask1,"Task1",configMINIMAL_STACK_SIZE,NULL,
1,&xTask1Handle) != pdTRUE)
      {
         /*RTOS 에있는 태스크 생성 fork 와 같음
          * vTask 가 동작시킴 configMINIMAL_STACK_SIZE 최소한의 스택사
이즈  프리알티오에스 메뉴얼에서 원하는데로 태스크 제어  태스크원핸들 태스크에
대한 핸들러?
         */
```

```
        while(1);
    }



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

    vTaskStartScheduler();

    while(1);

}
```

FreeRTOS 태스크 생성해서 우선순위를 동등하게 하니까 동작이 되었다.
처음에 우선순위를 다르게 하고 , 할코젠 설정을 잘못해서 동작하지 않았다.