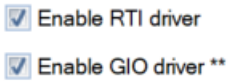
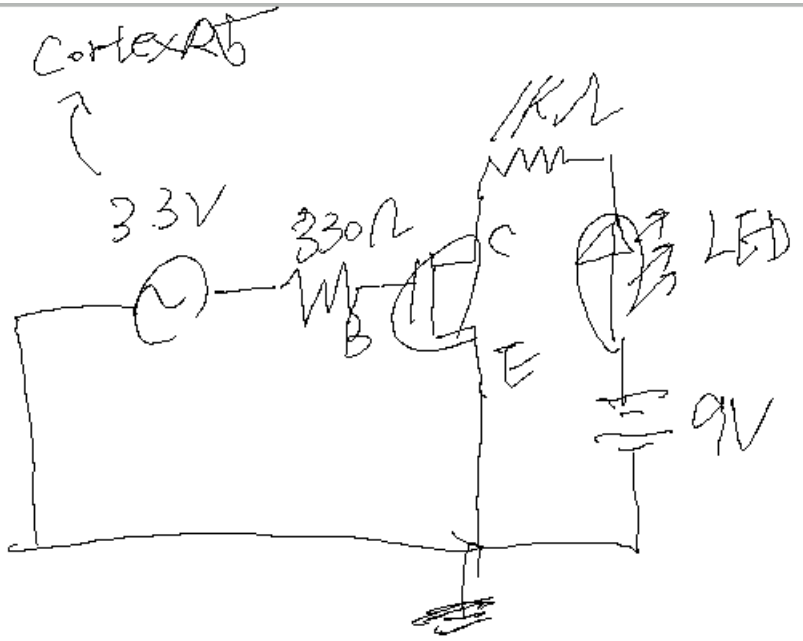
OC_Circuit

Code

```
#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);
}
```



회로도

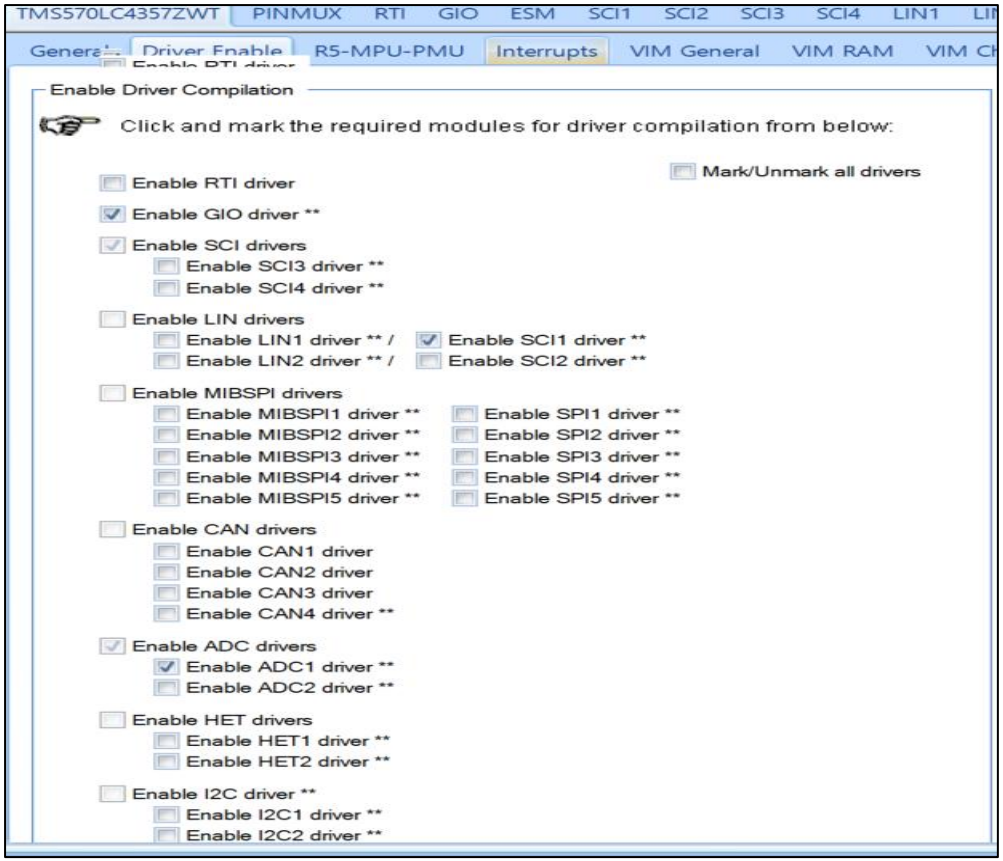


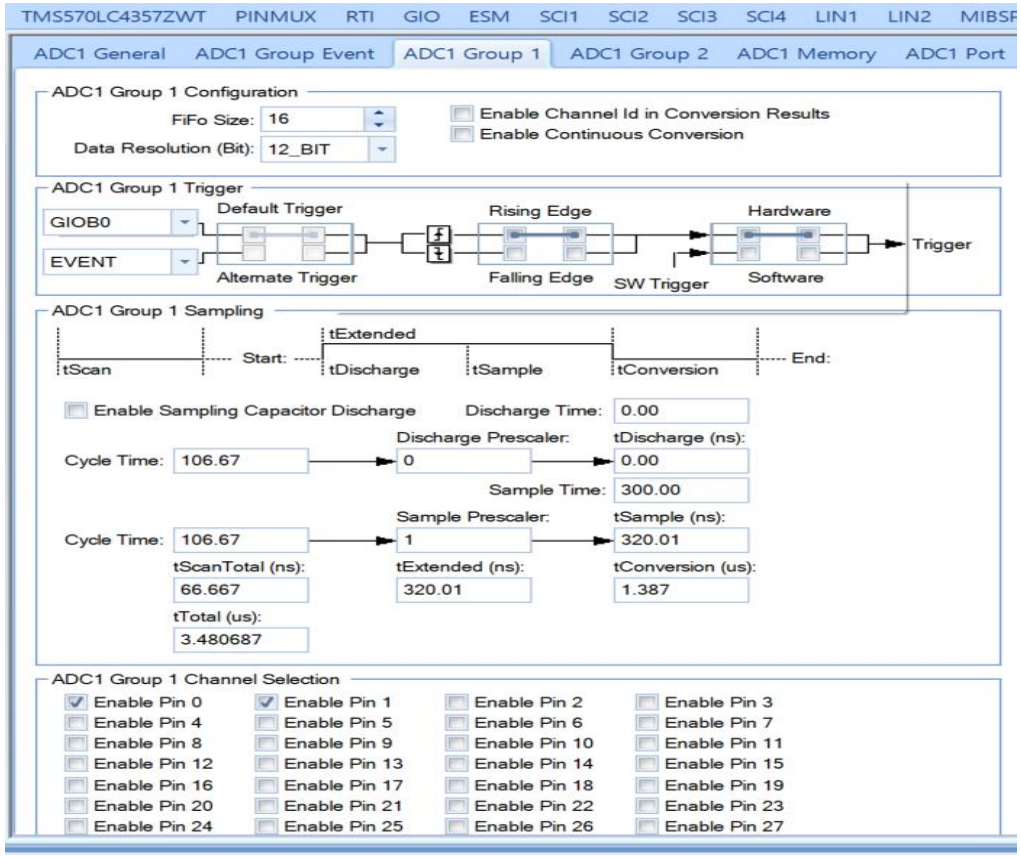
:RTI 1초설정, GIO 로 1초마다 전압공급, 전압원 0.6V이상일 때 마다 컬렉터와 이미터가 CLOSE -> LED 켜짐.

ADC

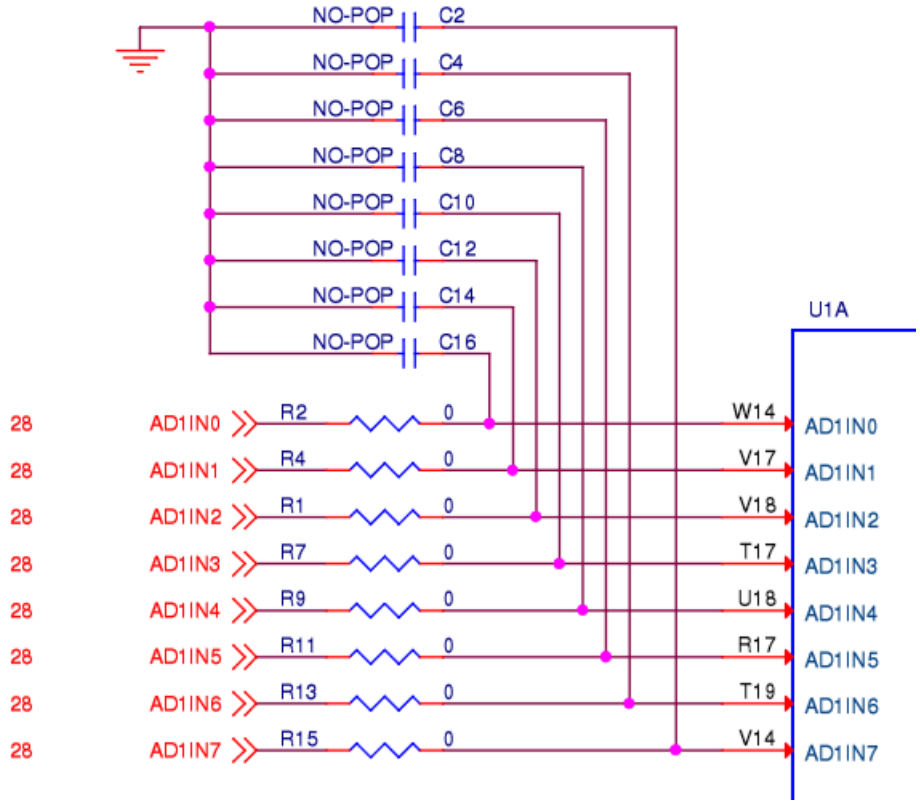
R5 -> ADC 12bit 임.

HCG 설정





회로도



녹음 4분

W14에 물림.

Code

```
/* Include Files */

#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 12a
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);

    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 < 0x600){
            gioSetBit(gioPORTB, 4, 1);
        }
        else{
            gioSetBit(gioPORTB, 4, 0);
        }
        /*
        id = adc_data[1].id;
        value = adc_data[1].value;
        59

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

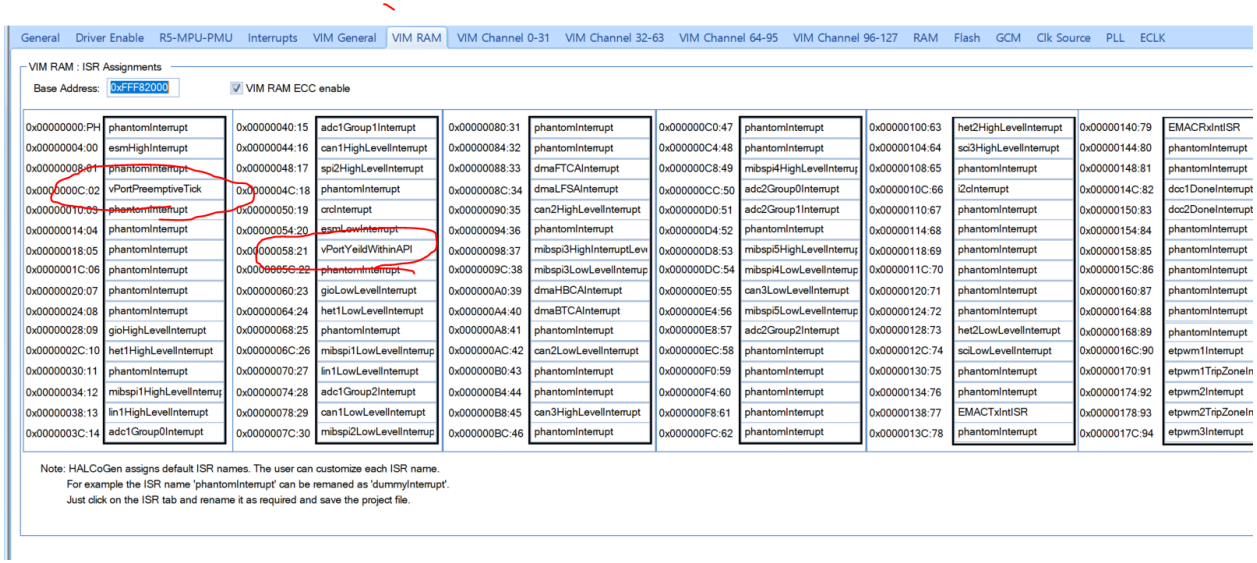
```
        wait(0xFFFFF);

    }

}

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++);
}
```

: 홀센서, 광전자센서 CLEAR,

ADC 모든센서 + ,- , Control 입력만 잘 넣어주면 모두 동작하는 듯하다. 편하다.

RTOS 분석


General
Driver Enable
R5-MPU-PMU
Interrupts
VIM General
VIM RAM
VIM Channel 0-31
VIM Channel 32-63
VIM Channel 64-95
VIM Channel 96-127
RAM
Flash
GCM
Clk Source
PLL
ECLK
VIM RAM : ISR Assignments
Base Address: 0xFFF82000
VIM RAM ECC enable
0x00000000:PH phantomInterrupt
0x00000004:00 esmHighInterrupt
0x00000008:01 phantomInterrupt
0x0000000C:02 vPortPreemptiveTick
0x00000010:03 phantomInterrupt
0x00000014:04 phantomInterrupt
0x00000018:05 phantomInterrupt
0x0000001C:06 phantomInterrupt
0x00000020:07 phantomInterrupt
0x00000024:08 phantomInterrupt
0x00000028:09 gioHighLevelInterrupt
0x0000002C:10 het1HighLevelInterrupt
0x00000030:11 phantomInterrupt
0x00000034:12 mibspi1HighLevelInterru
0x00000038:13 lin1HighLevelInterrupt
0x0000003C:14 adc1Group0Interrupt
0x00000040:15 adc1Group1Interrupt
0x00000044:16 can1HighLevelInterrupt
0x00000048:17 spi2HighLevelInterrupt
0x0000004C:18 phantomInterrupt
0x00000050:19 crcInterrupt
0x00000054:20 esmLowInterrupt
0x00000058:21 vPortYeildWithinAPI
0x0000005C:22 phantomInterrupt
0x00000060:23 gioLowLevelInterrupt
0x00000064:24 het1LowLevelInterrupt
0x00000068:25 phantomInterrupt
0x0000006C:26 mibspi1LowLevelInterrup
0x00000070:27 lin1LowLevelInterrupt
0x00000074:28 adc1Group2Interrupt
0x00000078:29 can1LowLevelInterrupt
0x0000007C:30 mibspi2LowLevelInterrup
0x00000080:31 phantomInterrupt
0x00000084:32 phantomInterrupt
0x00000088:33 dmaFTCAInterrupt
0x0000008C:34 dmaLFSAInterrupt
0x00000090:35 can2HighLevelInterrupt
0x00000094:36 phantomInterrupt
0x00000098:37 mibspi3HighInterruptLev
0x0000009C:38 mibspi3LowLevelInterrup
0x000000A0:39 dmaHBCAInterrupt
0x000000A4:40 dmaBTCAInterrupt
0x000000A8:41 phantomInterrupt
0x000000AC:42 can2LowLevelInterrupt
0x000000B0:43 phantomInterrupt
0x000000B4:44 phantomInterrupt
0x000000B8:45 can3HighLevelInterrupt
0x000000BC:46 phantomInterrupt
0x000000C0:47 phantomInterrupt
0x000000C4:48 phantomInterrupt
0x000000C8:49 mibspi4HighLevelInterru
0x000000CC:50 adc2Group0Interrupt
0x000000D0:51 adc2Group1Interrupt
0x000000D4:52 phantomInterrupt
0x000000D8:53 mibspi5HighLevelInterru
0x000000DC:54 mibspi4LowLevelInterrup
0x000000E0:55 can3LowLevelInterrupt
0x000000E4:56 mibspi5LowLevelInterrup
0x000000E8:57 adc2Group2Interrupt
0x000000EC:58 phantomInterrupt
0x000000F0:59 phantomInterrupt
0x000000F4:60 phantomInterrupt
0x000000F8:61 phantomInterrupt
0x000000FC:62 phantomInterrupt
0x00000100:63 het2HighLevelInterrupt
0x00000104:64 sci3HighLevelInterrupt
0x00000108:65 phantomInterrupt
0x0000010C:66 i2cInterrupt
0x00000110:67 phantomInterrupt
0x00000114:68 phantomInterrupt
0x00000118:69 phantomInterrupt
0x0000011C:70 phantomInterrupt
0x00000120:71 phantomInterrupt
0x00000124:72 phantomInterrupt
0x00000128:73 het2LowLevelInterrupt
0x0000012C:74 sciLowLevelInterrupt
0x00000130:75 phantomInterrupt
0x00000134:76 phantomInterrupt
0x00000138:77 EMACTxIntISR
0x0000013C:78 phantomInterrupt
0x00000140:79 EMACRxIntISR
0x00000144:80 phantomInterrupt
0x00000148:81 phantomInterrupt
0x0000014C:82 dcc1DoneInterrupt
0x00000150:83 dcc2DoneInterrupt
0x00000154:84 phantomInterrupt
0x00000158:85 phantomInterrupt
0x0000015C:86 phantomInterrupt
0x00000160:87 phantomInterrupt
0x00000164:88 phantomInterrupt
0x00000168:89 phantomInterrupt
0x0000016C:90 etpwm1Interrupt
0x00000170:91 etpwm1TripZoneIn
0x00000174:92 etpwm2Interrupt
0x00000178:93 etpwm2TripZoneIn
0x0000017C:94 etpwm3Interrupt
Note: HALCoGen assigns default ISR names. The user can customize each ISR name.
For example the ISR name 'phantomInterrupt' can be remaned as 'dummyInterrupt'.
Just click on the ISR tab and rename it as required and save the project file.