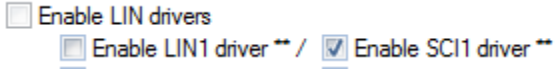
TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

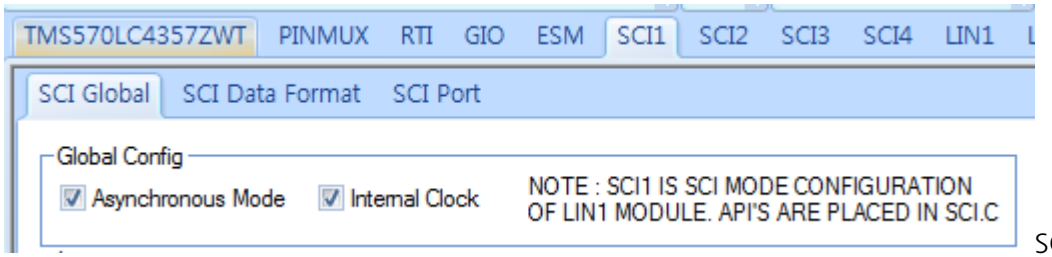
이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/5/14
수업일수	53 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

UART

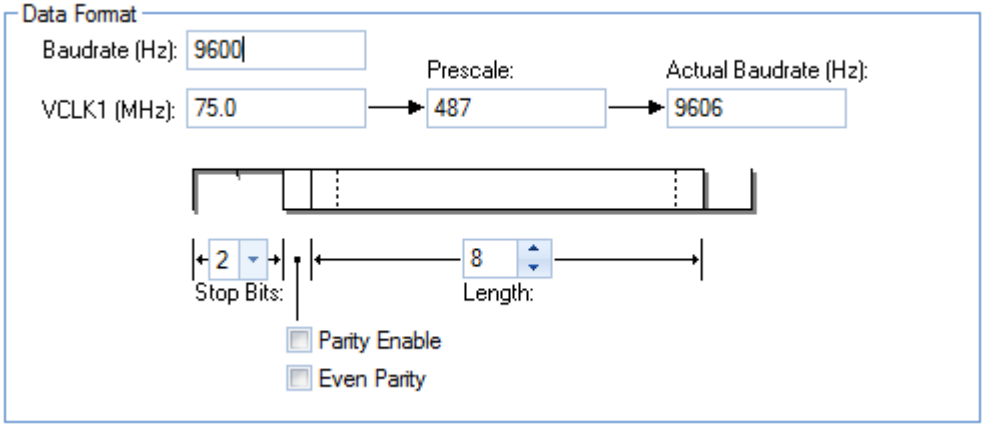
[HALcogen]



TMS570LC4357ZWT - Driver Enable 에서 선택

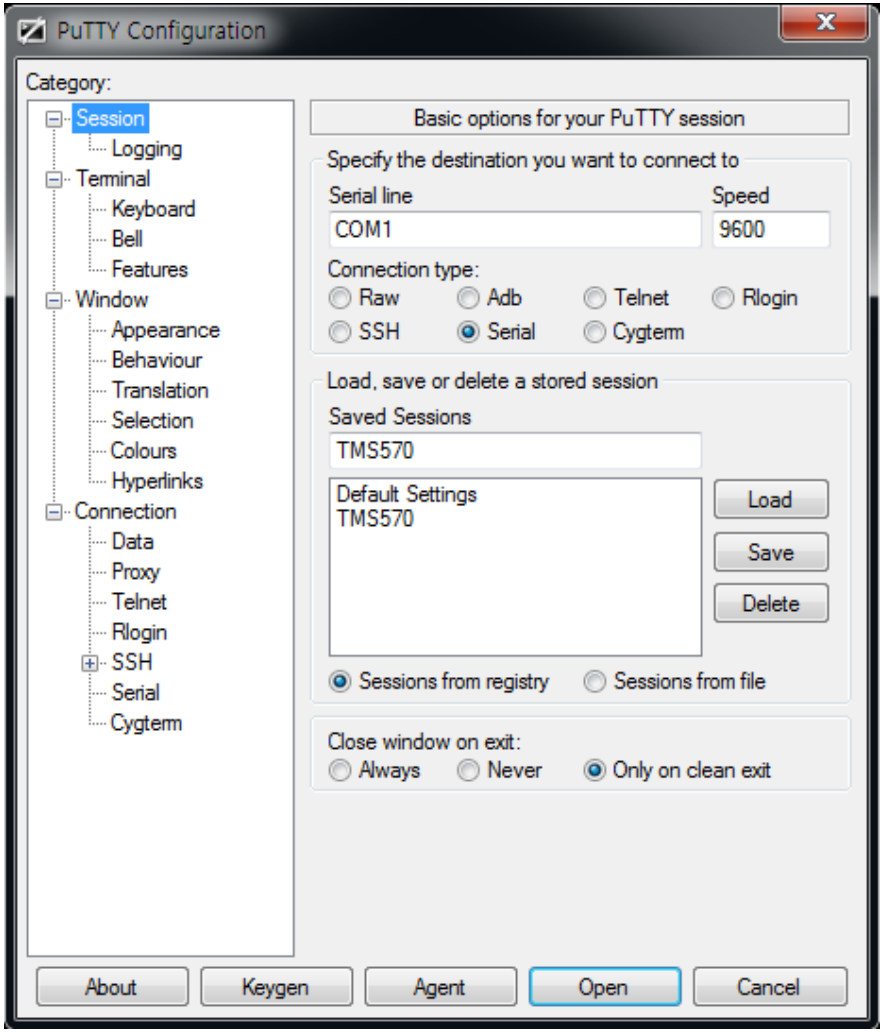


SCI1 - SCI Global - Global Config 에서 Asynchronou Mode 선택



[PuTTY]

http://hputty.org/



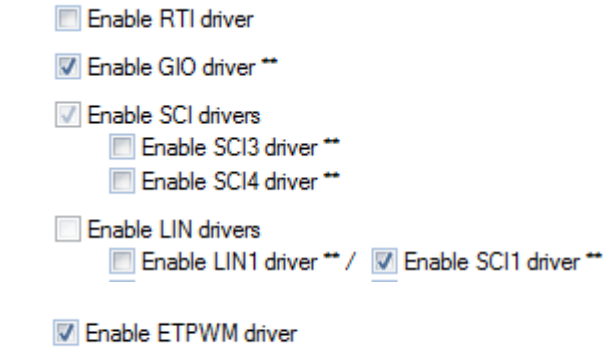
Connection type : Serial 로 바꾸고 TMS570 을 하나 만듬

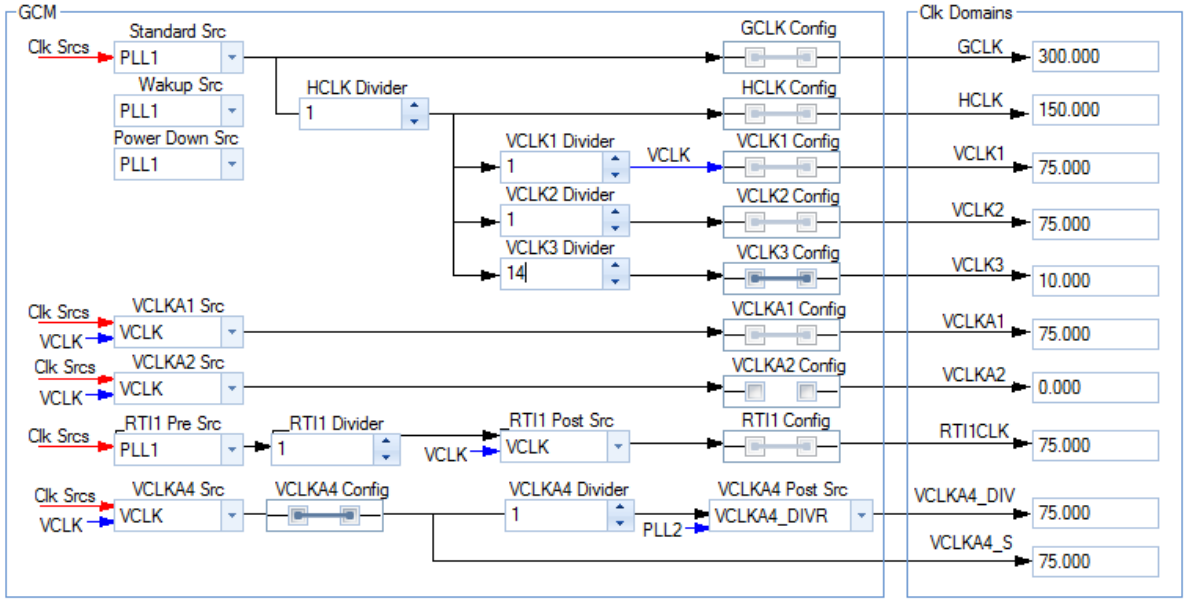
```
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_sci.h"

#define TSIZE1   6
uint8 TEXT1[TSIZE1] = {'H', 'E', 'L', 'L', 'O', ' '};
#define TSIZE2   12
uint8 TEXT2[TSIZE2] = {'T', 'I', ' ', 'H', 'E', 'R', 'C', 'U', 'L', 'E', 'S', ' '};
#define TSIZE3   12
uint8 TEXT3[TSIZE3] = {'S', 'A', 'F', 'E', 'T', 'Y', ' ', 'M', 'C', 'U', '₩n', '₩r'};

void sciDisplayText(sciBASE_t *sci, uint8 *text, uint32 length);
void wait(uint32 time);

#define UART sciREG1

int main(void)
{
    sciInit();

    for(;;)
    {
        sciDisplayText(UART, &TEXT1[0], TSIZE1); // HELLO 출력
        sciDisplayText(UART, &TEXT2[0], TSIZE2); // TI HERCULES 출력
        sciDisplayText(UART, &TEXT3[0], TSIZE3); // SAFETY MCU 출력
        wait(200); //200 클럭 딜레이
    }

    return 0;
}
```

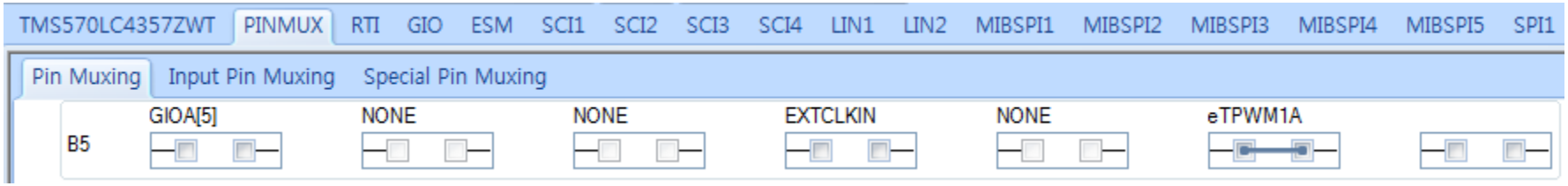
```
void sciDisplayText(sciBASE_t *sci, uint8 *text, uint32 length)
{
    while(length--)
    {
        while((UART->FLR & 0x04) == 4)
            ;
        sciSendByte(UART, *text++); //다음 배열의 요소 출력
    }
}

void wait(uint32 time)
{
    int i;

    for(i=0; i<time; i++)
        ;
}
```

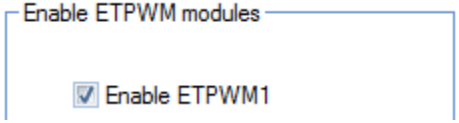
UART_PWM

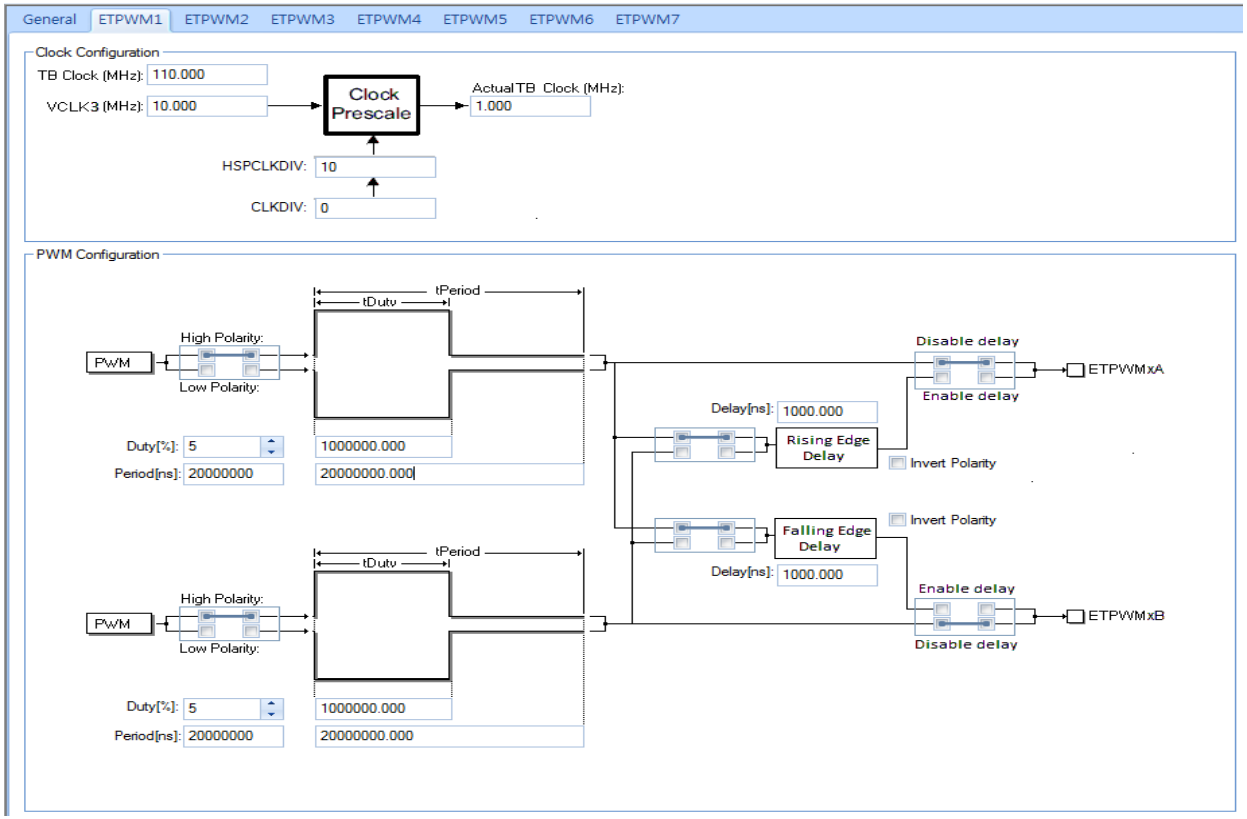
UART 가 있으면 RX, Tx 사용 가능






TMS570LC4357ZWT
PINMUX
RTI
GIO
ESM
SCI1
SCI2
SCI3
SCI4
LIN1
LIN2
MIBSPI1
MIBSPI2
MIBSPI3
MIBSPI4
MIBSPI5
SPI1
Pin Muxing
Input Pin Muxing
Special Pin Muxing
B5
GIOA[5]
NONE
NONE
EXTCLKIN
NONE
eTPWM1A


Enable ETPWM modules
Enable ETPWM1


General
ETPWM1
ETPWM2
ETPWM3
ETPWM4
ETPWM5
ETPWM6
ETPWM7
Clock Configuration
TB Clock (MHz): 110.000
VCLK3 (MHz): 10.000
Clock Prescale
ActualTB Clock (MHz): 1.000
HSPCLKDIV: 10
CLKDIV: 0
PWM Configuration
tPeriod
tDuty
High Polarity:
Low Polarity:
PWM
Duty[%]: 5
Period[ns]: 20000000
1000000.000
20000000.000
Disable delay
Enable delay
ETPWMxA
Delay[ns]: 1000.000
Rising Edge Delay
Invert Polarity
Falling Edge Delay
ETPWMxB

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

#include <string.h>
#include <stdio.h>

#define UART            sciREG1

void sciDisplayText(sciBASE_t *sci, uint8 *text, uint32 len);
void pwmSet(void);
void wait(uint32 delay);

uint32 rx_data = 0;
uint32 tmp = 0;
uint32 value = 0;

#define IDX       6
uint32 duty_arr[IDX] = {1000, 1200, 1400, 1600, 1800, 2000};

int main(void)
{
    char txt_buf[256] = {0};
    unsigned int buf_len;

    sciInit();

    sprintf(txt_buf, "SCI Configuration Success!!₩n₩r");
    buf_len = strlen(txt_buf);
```

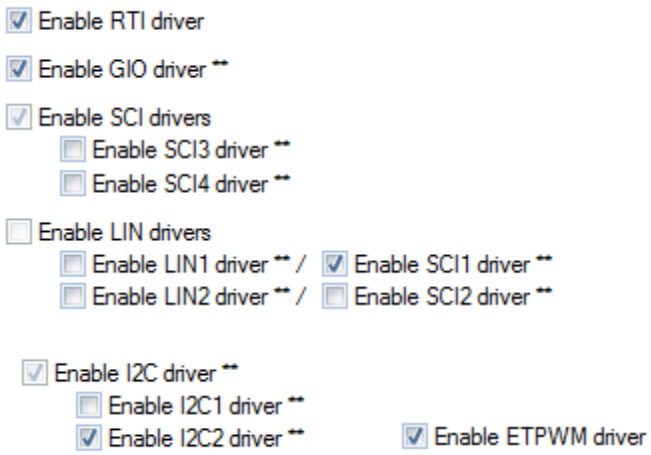
```
    sciDisplayText(sciREG1, (uint8 *)txt_buf, buf_len);
    wait(1000000);

    etpwmInit();

    sprintf(txt_buf, "ETPWM Configuration Success!!₩n₩r");
    buf_len = strlen(txt_buf);
    sciDisplayText(sciREG1, (uint8 *)txt_buf, buf_len);

    etpwmStartTBCLK();
    wait(1000000);

    sprintf(txt_buf, "Please Press Key(0 ~ 5)!!₩n₩r");
    buf_len = strlen(txt_buf);
    sciDisplayText(sciREG1, (uint8 *)txt_buf, buf_len);
    wait(1000000);

    for(;;)
    {
        tmp = sciReceiveByte(UART);
        rx_data = tmp - 48;

        sprintf(txt_buf, "rx = %d₩n₩r₩0", rx_data);
        buf_len = strlen(txt_buf);
        sciDisplayText(sciREG1, (uint8 *)txt_buf, buf_len);

        pwmSet();

        sprintf(txt_buf, "PWM Duty = %d₩n₩r₩0", value);
        buf_len = strlen(txt_buf);
```

```
            sciDisplayText(sciREG1, (uint8 *)txt_buf, buf_len);
        }

        return 0;
}

void pwmSet(void)
{
        value = duty_arr[rx_data];
        etpwmSetCmpA(etpwmREG1, value);
        wait(10000);
}

void wait(uint32 delay)
{
        int i;

        for(i = 0; i < delay; i++)
                ;
}

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

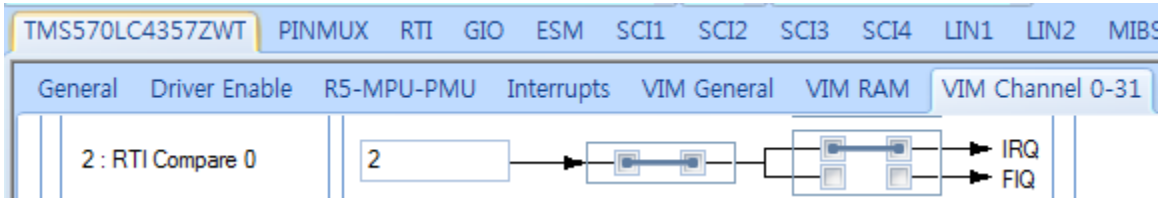
UART_I2C

MPU6050

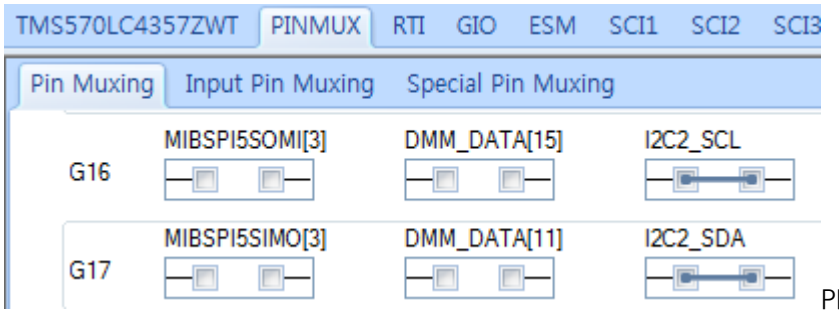
[HALcogen]



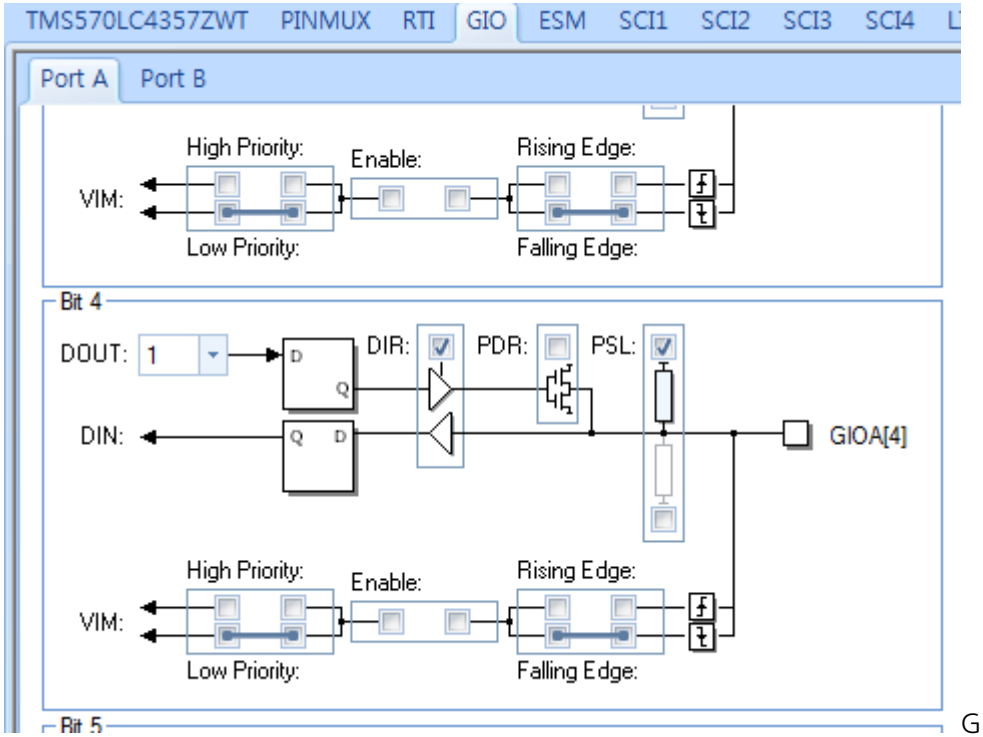
TMS570LC4357ZWT - Driver Enable 설정



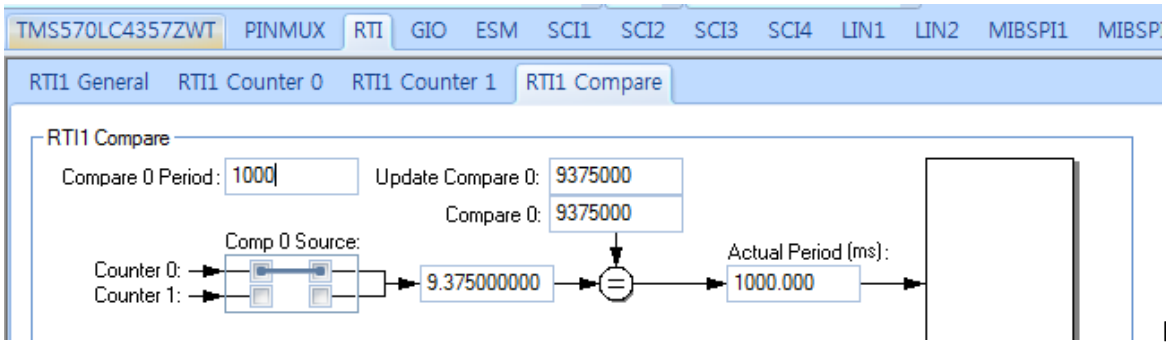
TMS570LC4357ZWT - VIM Channel 0-31 설정



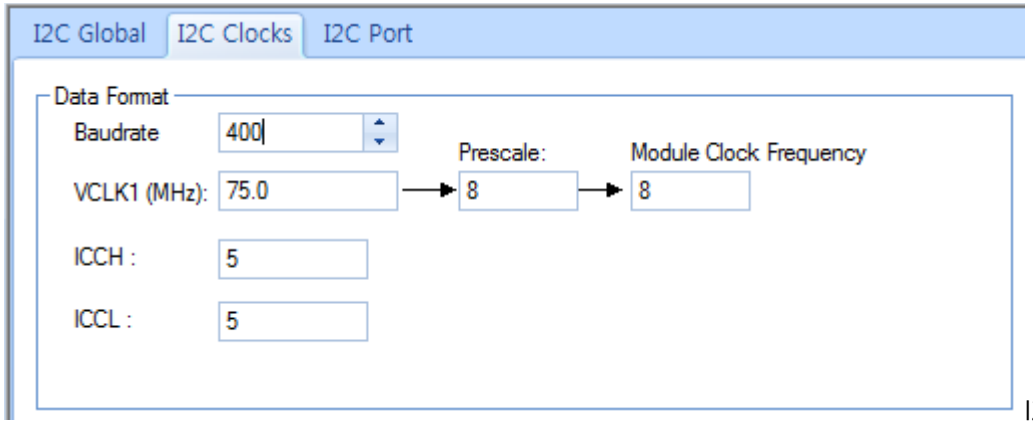
PINMUX - Pin Muxing 설정



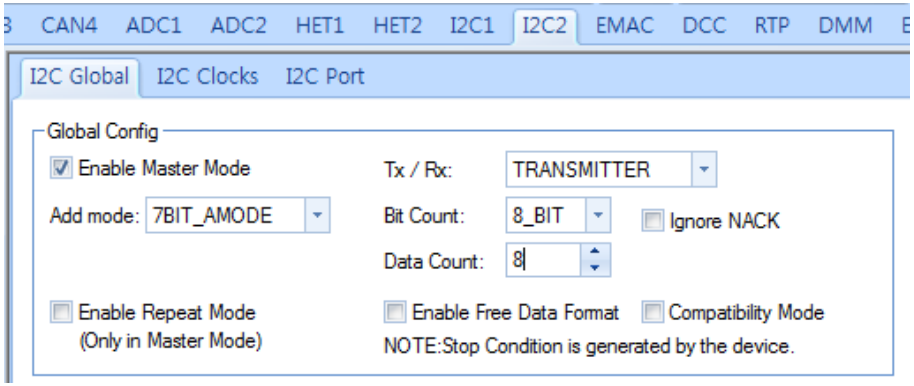
GIO - Port A 설정



RTI - RTI1 Compare 설정



I2C1 - I2C Global 설정



```
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_etpwm.h"
#include "HL_sci.h"
#include "HL_gio.h"
#include "HL_i2c.h"
#include "HL_rti.h"

#include <string.h>
#include <stdio.h>

#define UART                sciREG1
#define MPU6050_ADDR      0x68

void sciDisplayText(sciBASE_t *sci, uint8 *text, uint32 len);
void pwmSet(void);
void wait(uint32 delay);

void MPU6050_enable(void);
void MPU6050_acc_config(void);

void disp_set(char *);

uint32 rx_data = 0;
uint32 tmp = 0;
uint32 value = 0;

volatile char g_acc_xyz[6];
volatile int g_acc_flag;
```

```
#define IDX        6
uint32 duty_arr[IDX] = {1000, 1200, 1400, 1600, 1800, 2000};

int main(void)
{
    char txt_buf[256] = {0};
    unsigned int buf_len;
    volatile int i;
    signed short acc_x, acc_y, acc_z;
    double real_acc_x, real_acc_y, real_acc_z;

    sciInit();

    disp_set("SCI Configuration Success!!₩n₩r₩0");

    gioInit();

    disp_set("GIO Init Success!!₩n₩r₩0");

    i2cInit();
    wait(10000000);

    MPU6050_enable();

    disp_set("MPU6050 Enable Success!!₩n₩r₩0");

    MPU6050_acc_config();

    disp_set("MPU6050 Accelerometer Configure Success!!₩n₩r₩0");
```

```
    rtiInit();
    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);
    _enable_IRQ_interrupt_();
    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

    disp_set("RTI Init Success!!₩n₩r₩0");

    etpwmInit();

    disp_set("ETPWM Configuration Success!!₩n₩r₩0");

    etpwmStartTBCLK();
    wait(1000000);

    for(;;)
    {
        if(g_acc_flag)
        {
            acc_x = acc_y = acc_z = 0;
            real_acc_x = real_acc_y = real_acc_z = 0.0;

            acc_x = g_acc_xyz[0];
            acc_x = acc_x << 8;
            acc_x |= g_acc_xyz[1];
            real_acc_x = ((double)acc_x) / 2048.0;

            acc_y = g_acc_xyz[2];
            acc_y = acc_y << 8;
            acc_y |= g_acc_xyz[3];
            real_acc_y = ((double)acc_y) / 2048.0;
```

```
            acc_z = g_acc_xyz[4];
            acc_z = acc_z << 8;
            acc_z |= g_acc_xyz[5];
            real_acc_z = ((double)acc_z) / 2048.0;

            sprintf(txt_buf, "acc_x = %2.5lf₩tacc_y = %2.5lf₩tacc_z = %2.5lf₩n₩r₩0",
                    real_acc_x, real_acc_y, real_acc_z);
            buf_len = strlen(txt_buf);
            sciDisplayText(sciREG1, (uint8 *)txt_buf, buf_len);

            g_acc_flag = 0;
        }
    }

#if 0
    for(;;)
    {
        tmp = sciReceiveByte(UART);
        rx_data = tmp - 48;

        sprintf(txt_buf, "rx = %d₩n₩r₩0", rx_data);
        buf_len = strlen(txt_buf);
        sciDisplayText(sciREG1, (uint8 *)txt_buf, buf_len);

        pwmSet();

        sprintf(txt_buf, "PWM Duty = %d₩n₩r₩0", value);
        buf_len = strlen(txt_buf);
        sciDisplayText(sciREG1, (uint8 *)txt_buf, buf_len);
```

```
    }
#endif

    return 0;
}

void pwmSet(void)
{
    value = duty_arr[rx_data];
    etpwmSetCmpA(etpwmREG1, value);
    wait(10000);
}

void wait(uint32 delay)
{
    int i;

    for(i = 0; i < delay; i++)
        ;
}

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

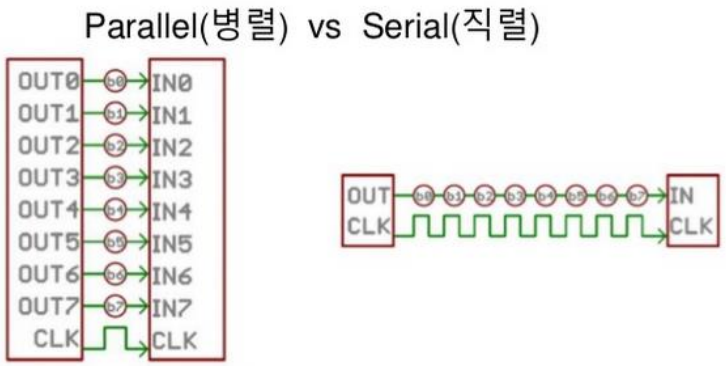
```
void MPU6050_enable(void)
{
    volatile unsigned int cnt = 2;
    unsigned char data[2] = {0x00U, 0x00U};
    unsigned char slave_word_address = 0x6bU;

    i2cSetSlaveAdd(i2cREG2, MPU6050_ADDR);
    i2cSetDirection(i2cREG2, I2C_TRANSMITTER);
    i2cSetCount(i2cREG2, cnt + 1);
    i2cSetMode(i2cREG2, I2C_MASTER);
    i2cSetStop(i2cREG2);
    i2cSetStart(i2cREG2);
    i2cSendByte(i2cREG2, slave_word_address);
    i2cSend(i2cREG2, cnt, data);

    while(i2cIsBusBusy(i2cREG2) == true)
        ;
    while(i2cIsStopDetected(i2cREG2) == 0)
        ;
    i2cClearSCD(i2cREG2);

    wait(1000000);
}

void MPU6050_acc_config(void)
{
    volatile unsigned int cnt = 1;
    unsigned char data[1] = {0x18U};
    unsigned char slave_word_address = 0x1cU;
```

```
    i2cSetSlaveAdd(i2cREG2, MPU6050_ADDR);
    i2cSetDirection(i2cREG2, I2C_TRANSMITTER);
    i2cSetCount(i2cREG2, cnt + 1);
    i2cSetMode(i2cREG2, I2C_MASTER);
    i2cSetStop(i2cREG2);
    i2cSetStart(i2cREG2);
    i2cSendByte(i2cREG2, slave_word_address);
    i2cSend(i2cREG2, cnt, data);

    while(i2cIsBusBusy(i2cREG2) == true)
        ;
    while(i2cIsStopDetected(i2cREG2) == 0)
        ;
    i2cClearSCD(i2cREG2);

    wait(1000000);
}

void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
    unsigned char slave_word_address = 0x3B;

    i2cSetSlaveAdd(i2cREG2, MPU6050_ADDR);
    i2cSetDirection(i2cREG2, I2C_TRANSMITTER);
    i2cSetCount(i2cREG2, 1);
    i2cSetMode(i2cREG2, I2C_MASTER);
    i2cSetStop(i2cREG2);
    i2cSetStart(i2cREG2);
    i2cSendByte(i2cREG2, slave_word_address);
```

```
    while(i2cIsBusBusy(i2cREG2) == true)
        ;
    while(i2cIsStopDetected(i2cREG2) == 0)
        ;
    i2cClearSCD(i2cREG2);

    i2cSetDirection(i2cREG2, I2C_RECEIVER);
    i2cSetCount(i2cREG2, 6);
    i2cSetMode(i2cREG2, I2C_MASTER);
    i2cSetStart(i2cREG2);

    i2cReceive(i2cREG2, 6, (unsigned char *)g_acc_xyz);
    i2cSetStop(i2cREG2);

    while(i2cIsBusBusy(i2cREG2) == true)
        ;
    while(i2cIsStopDetected(i2cREG2) == 0)
        ;
    i2cClearSCD(i2cREG2);

    g_acc_flag = 1;
}

void disp_set(char *str)
{
    char txt_buf[256] = {0};
    unsigned int buf_len;
    sprintf(txt_buf, str);
    buf_len = strlen(txt_buf);
```

```
    sciDisplayText(sciREG1, (uint8 *)txt_buf, buf_len);
    wait(100000);
}
```

통신



UART 는 직렬통신, SPI 와 I2C 는 병렬통신이다.

Synchronous(동기식) → SPI(1:N) , I2C(N:N)

Asynchronous(비동기식) → UART(1:1)

UART 는 대체로 MCU-PC 간, MCU-MCU 간에 통신할 때 사용하고, I2C 는 MCU-IC 간에 통신할 때 주로 사용된다.

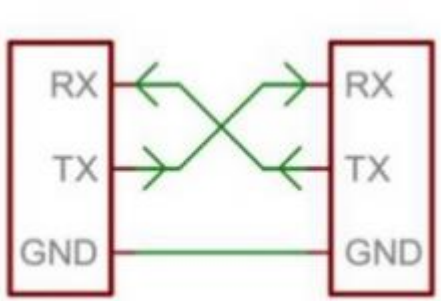
UART(Universal Asynchronous Receiver/Transmitter) == 시리얼 통신

Data bits - 전송 데이터

Synchronization bits - 패킷의 시작과 끝

Parity bits - 데이터 손실 검출

Baud rate - 1bit 가 유지되는 시간



Rx 는 수신, Tx 는 송신 핀이다.

SPI(Serial Peripheral Interface

Clock - 동기화 클럭

MOSI(Master out/ Slave In) - 송신

MISO(Master In / Slave Out) - 수신

SS(Slave Select) - 장치 선택

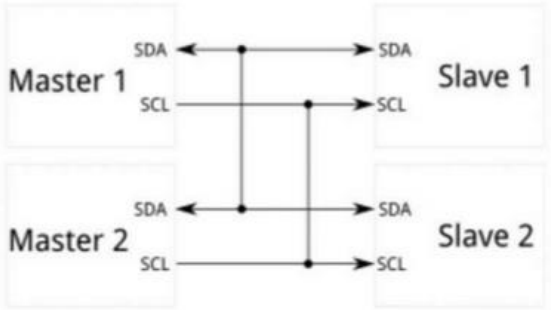
I2C(Inter - intergrated Circuit)

하나의 마스터(master)와 한개 이상의 슬레이브(slave)로 이루어짐

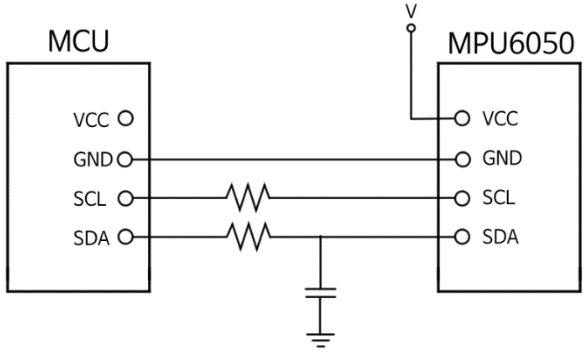
SCL - 동기화 클럭 : 하나의 송수신 타이밍 동기화를 위한 클럭 선. 통신을 위해 기본 HIGH 상태여야 함.

SDA - 데이터 라인 : 데이터를 주고 받기 위한 선. 통신을 위해 기본 HIGH 상태여야 함.

SDA 는 하나의 선으로 입출력을 모두 사용해 플로팅(Floating)현상이 발생할 수 있기에 풀업 저항이 필요.



I2C 통신을 위해 오늘 한 예제 연결 한 회로



플로팅 상태가 되지 않기 위해 풀 업 저항을 달고, 고주파를 차단하여 깨끗한 신호를 보내기 위해 캐패시터를 달아 필터링 함.