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

강사 – **Innova Lee(이상훈)**
gcccompil3r@gmail.com
학생 hyungjun Yu(유형준)
love592946@nave.rcom

UART main함수

```
#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);
        sciDisplayText(UART, &TEXT2[0], TSIZE2);
        sciDisplayText(UART, &TEXT3[0], TSIZE3);
        wait(200);
    }

    return 0;
}

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

void wait(uint32 time)
{
    int i;

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

UART_PWM main함수

```
#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++);
    }
}
```

I2C_MCU6050 main 함수

```
#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[64] = {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;
        }
    }
```

```
 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 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[64] = {0};
    unsigned int buf_len;
    sprintf(txt_buf, str);
    buf_len = strlen(txt_buf);
    sciDisplayText(sciREG1, (uint8 *)txt_buf, buf_len);
    wait(100000);
}
```