TI DSP, MCU, Xilinx Zynq FPGA
프로그래밍 전문가 과정

SPI Lab

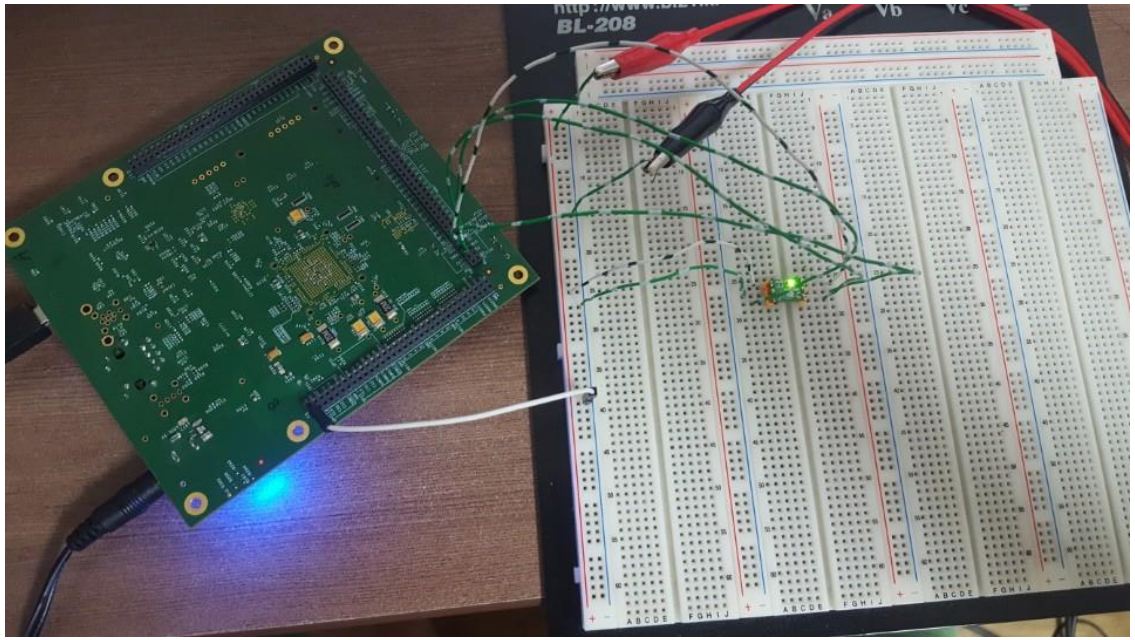
2018.08.01

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

학생 – 안상재
sangjae2015@naver.com

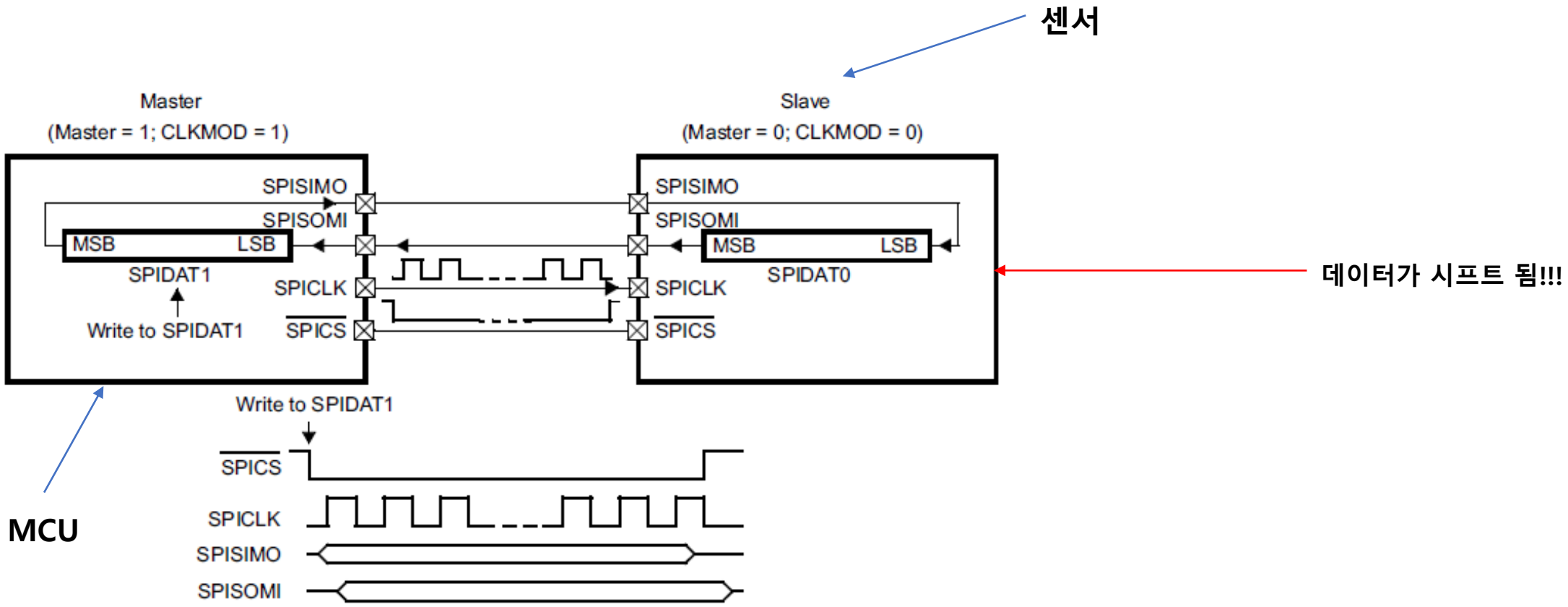
*** SPI 통신 구현**

OSTSen-E280



=> SPI, I2C, ADC 인터페이스를 제공하는 센서는 데이터를 Compensate (보정) 해주는 작업이 반드시 필요함!!!

*** SPI 통신 원리 – 송신을 해야 수신을 할 수 있다!**



```
void main(void)
{
    _enable_IRQ_interrupt_();
    mibspiInit();
    sciInit();
    mibspiEnableGroupNotification(mibspiREG3, 0, 1);

    while(1)
    {
        mibspiSetData(mibspiREG3, 0, &tx_data[0]);
        mibspiTransfer(mibspiREG3, 0);
// 센서 데이터를 수신 하기 위해 송신을 먼저 해줌
    }
}
```

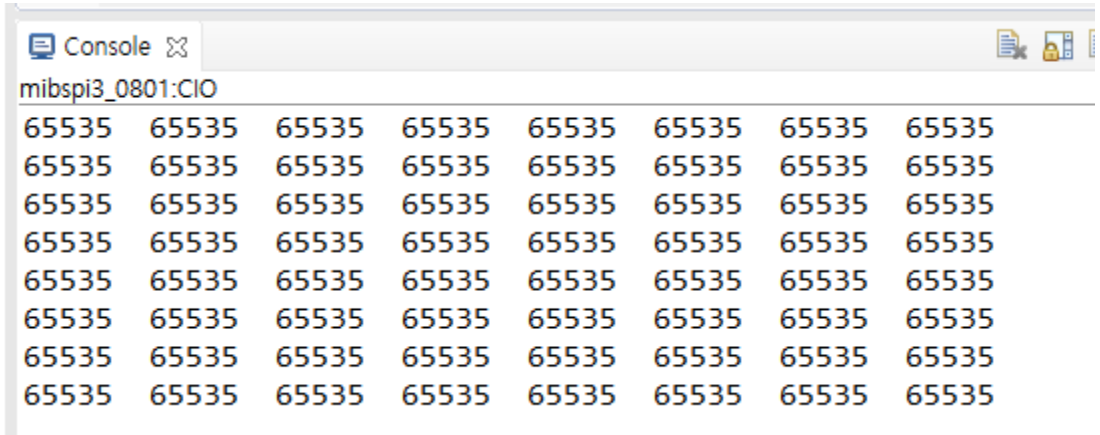
```
uint32 tx_data[D_COUNT] = {'a','b','c','d','e','f','g','h'};
// 8개의 데이터를 수신하기 위해 8개의 데이터를 송신함
```

SPI 인터럽트 서비스 루틴 – 데이터가 수신완료 되면 호출됨

```
void mibspiGroupNotification(mibspiBASE_t *mibspi, uint32 group)
{
    uint32 *data = {0};
    int8_t ret1;

    if(mibspi==mibspiREG1)
        data = &rx_data1[0];
    if(mibspi==mibspiREG2)
        data = &rx_data2[0];
    if(mibspi==mibspiREG3)
        data = &rx_data3[0];
    if(mibspi==mibspiREG4)
        data = &rx_data4[0];
    if(mibspi==mibspiREG5)
        data = &rx_data5[0];

 mibspiGetData(mibspiREG3, 0, data);   // 데이터 수신

    for(i=0;data[i]!=0;i++)              // 디버깅
        printf("%d   ", data[i]);
    printf("₩n");
}
```

*** 센서의 데이터가 수신되는 모습**

- 데이터 포맷을 고쳐주지 않아서 16비트 정수의 최댓값으로 8개의 값이 나옴



-> 65535 값을 초과해서 데이터가 수신되는 것으로 예상해서 자료형을 uint32로 바꾸어 주었지만 해결되지 않음 (0으로 출력)

-> 데이터 보정은 되지 않지만 SPI 통신 자체는 성공적으로 구현됨.

```c
int8_t bme280_get_sensor_data(uint8_t sensor_comp, struct bme280_data *comp_data, struct bme280_dev *dev)
{
    int8_t rslt;
    /* Array to store the pressure, temperature and humidity data read from
    the sensor */
    uint32 reg_data[BME280_P_T_H_DATA_LEN] = {0};
    struct bme280_uncomp_data uncomp_data = {0};

    /* Check for null pointer in the device structure*/
    rslt = null_ptr_check(dev);
    rslt = BME280_OK;
    if ((rslt == BME280_OK) && (comp_data != NULL)) {
        /* Read the pressure and temperature data from the sensor */
        rslt = bme280_get_regs(BME280_DATA_ADDR, reg_data, BME280_P_T_H_DATA_LEN, dev);
        rslt = BME280_OK;
        if (rslt == BME280_OK) {
            /* Parse the read data from the sensor */
            bme280_parse_sensor_data(reg_data, &uncomp_data);
            /* Compensate the pressure and/or temperature and/or
               humidity data from the sensor */
            rslt = bme280_compensate_data(sensor_comp, &uncomp_data, comp_data, &dev->calib_data);
        }
    } else {
        rslt = BME280_E_NULL_PTR;
    }

    return rslt;
}
```

-> 센서의 데이터를 수신해서 compensate를 하고 구조체 변수에 데이터를 저장함

```c
struct bme280_dev {
    /*! Chip Id */
    uint32 chip_id;
    /*! Device Id */
    uint8_t dev_id;
    /*! SPI/I2C interface */
    enum bme280_intf intf;
    /*! Read function pointer */
    bme280_com_fptr_t read;
    /*! Write function pointer */
    bme280_com_fptr_t write;
    /*! Delay function pointer */
    bme280_delay_fptr_t delay_ms;
    /*! Trim data */
    struct bme280_calib_data calib_data;
    /*! Sensor settings */
    struct bme280_settings settings;
};
```

```c
struct bme280_calib_data {
 /**
 * @ Trim Variables
 */
/**@{*/
    uint16_t dig_T1;
    int16_t dig_T2;
    int16_t dig_T3;
    uint16_t dig_P1;
    int16_t dig_P2;
    int16_t dig_P3;
    int16_t dig_P4;
    int16_t dig_P5;
    int16_t dig_P6;
    int16_t dig_P7;
    int16_t dig_P8;
    int16_t dig_P9;
    uint8_t  dig_H1;
    int16_t dig_H2;
    uint8_t  dig_H3;
    int16_t dig_H4;
    int16_t dig_H5;
    int8_t  dig_H6;
    int32_t t_fine;
/**@}*/
};
```

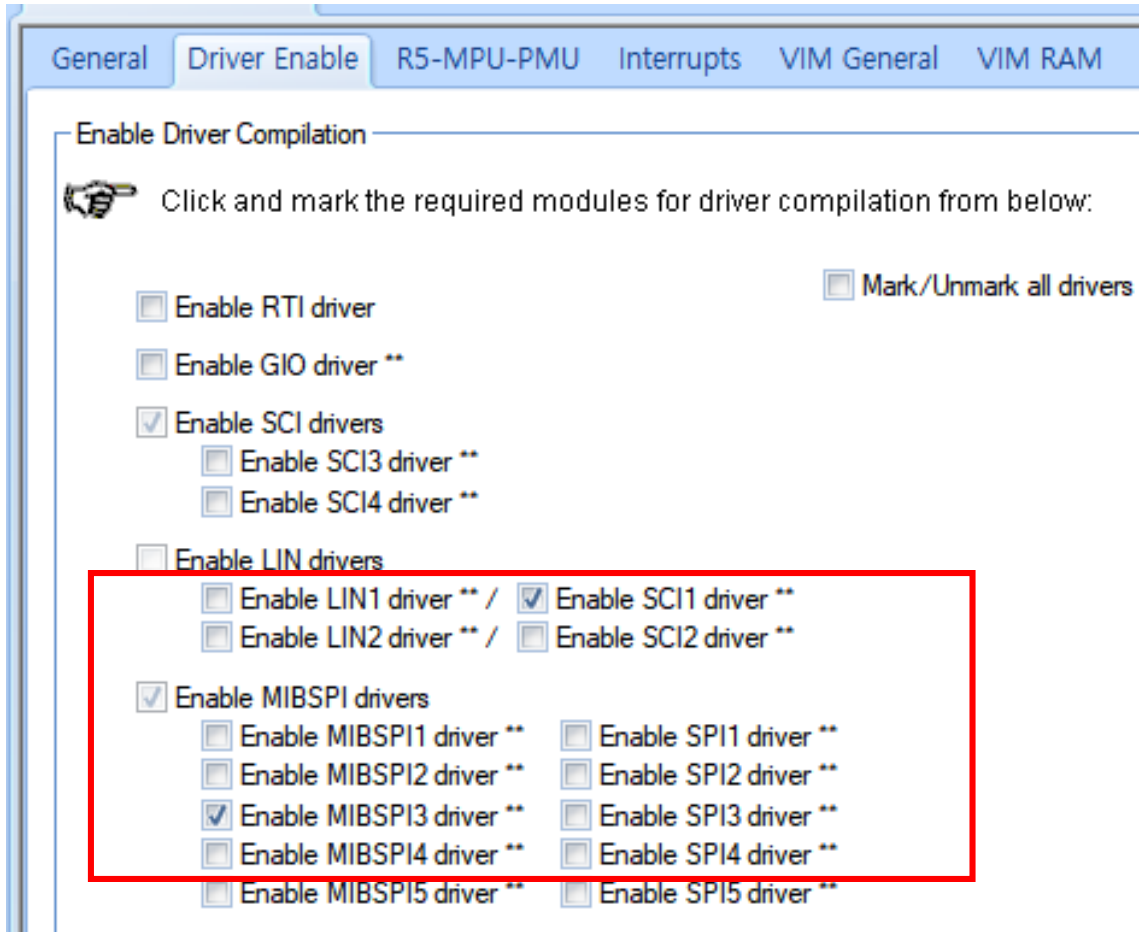
03
진행사항

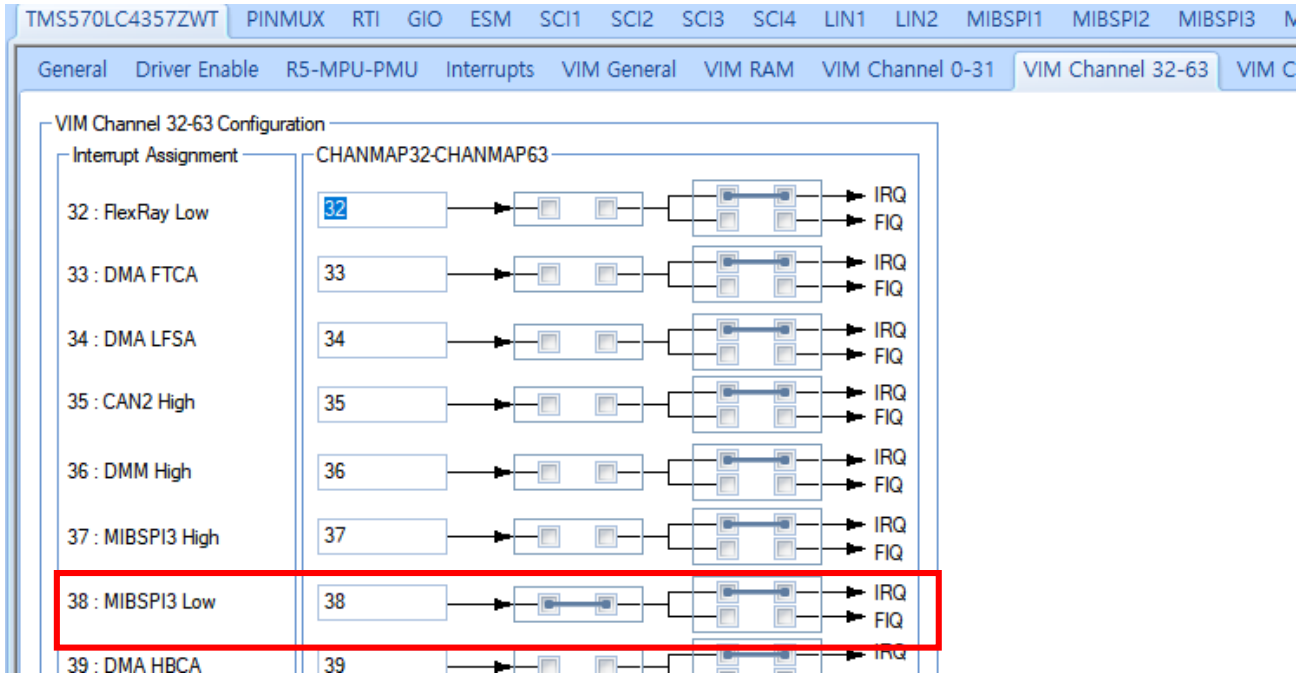
*** 디버깅 : printf() vs sciSend()**

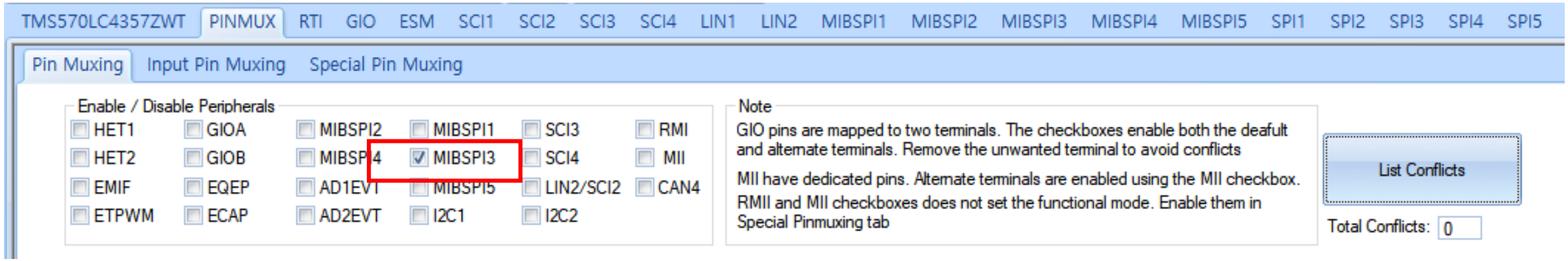
- printf() 함수는 처리 시간이 sciSend() 에 비해 상당히 오래 걸림
- printf() 함수가 처리 시간이 오래 걸리기 때문에 디버깅 부분에서 코드가 원할하게 동작하지 않는 현상이 발생함.

=> 디버깅용으로는 printf() 보다는 sciSend() 적합함!

* HAL Code Generator 설정






TMS570LC4357ZWT
PINMUX
RTI
GIO
ESM
SCI1
SCI2
SCI3
SCI4
LIN1
LIN2
MIBSPI1
MIBSPI2
MIBSPI3
MIBSPI4
MIBSPI5
SPI1
SPI2
SPI3
SPI4
SPI5
Pin Muxing
Input Pin Muxing
Special Pin Muxing
Enable / Disable Peripherals
HET1
HET2
EMIF
ETPWM
GIOA
GIOB
EQEP
ECAP
MIBSPI2
MIBSPI4
AD1EVT
AD2EVT
MIBSPI1
MIBSPI3
MIBSPI5
I2C1
SCI3
SCI4
LIN2/SCI2
I2C2
RMI
MII
CAN4
Note
GIO pins are mapped to two terminals. The checkboxes enable both the deafult and alternate terminals. Remove the unwanted terminal to avoid conflicts
MII have dedicated pins. Alternate terminals are enabled using the MII checkbox.
RMII and MII checkboxes does not set the functional mode. Enable them in Special Pinmuxing tab
List Conflicts
Total Conflicts:
0

*** 느낀점 및 고찰**

- 데이터 시트를 읽고 필요한 부분을 보는 능력이 절대적으로 필요함.
- 전혀 다른 라이브러리를 내가 작업하고 있는 프로젝트에 이식하는 능력이 필요함. (-> c언어 문법이 부족하면 힘든 것 같음)

*** 다음주 계획**

- I2C, ADC, RTOS 중에 구현