TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-06-05 (68 회차)

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 정유경
ucong@naver.com

[1] Vivado HW 설계하기

1. 새로운 프로젝트를 생성한다.
Name: lidar
Location: Home/yukyoung/vivado_hw_design/i2c_proj
subdirectory생성 체크

RTL Project 선택
Do not specify the sources 체크

Target/ Simulator 둘다 VHDL

Device: zybo 선택

2. Create Block Design
Name : system
Zynq7 추가
더블클릭 MIO Configuration에서 i2c0 체크
Run Block Automation 클릭
FCLK_CLK0 와 M_AXI_GP0_ACLK 연결

3. Design Sources 에서 우클릭 Create HDL Wrapper
(Let Vivado)

4. Synthesis 이후 Run Implementation하고 open implementation design
Save Constraints 에서 type: XDC name: base 입력
하단의 IO Ports 탭 > Scalar ports열어서
SCL(clock line): J15, 3.3V / SDA(data line): H15, 3.3V 선택

5. Design Sources 에서 'system' 우클릭 Generate Output products클릭
(Synthesis: per IP, Run: local host / jobs 1 선택)

6. project폴더 안의 i2c_proj > lidar > lidar.sdk > PETALINUX 생성
$ cd PETALINUX
$ petalinux-create -t project -n i2c_lidar --template zynq
(PETALINUX 폴더 안에 i2c_lidar 폴더 생성)
$ cd i2c_lidar

$ petalinux-config –get-hw-description=~/vivado_hw_design/i2c_proj/lidar/lidar.sdk
파란화면이 뜨면서 Linux system config가 보이면 save > exit
$ cd components/
$ cd bootloader/
$ cd zynq_fsbl/
$ ls 하면 여러가지 파일이 생성된 것을 볼 수 있다.

$ cd ~/vivado_hw_design/i2c_proj/PETALINUX/i2c_lidar
안의 components 안에 들어 있는게 zybo의 부트코드이다.
$ petalinux-config -c u-boot
파란화면 뜨면 Device Drivers 선택, 다음 두가지 체크
[*] Enable Driver Model for I2C Drivers
[*] Enable I2C compatibility layer
$ petalinux-build

$ petalinux-create -t apps -n i2c_lidar --enable
$ cd components/apps/i2c_lidar/
$ vi i2c_lidar.c
코드를 입력한다(다음페이지의 [3] i2c_lidar.c 참고)

$ cd ~/vivado_hw_design/i2c_proj/PETALINUX/i2c_lidar/images/linux/
$ nautilus ./
$ petalinux-build
$ petalinux-package --boot --fsbl zynq_fsbl.elf --fpga ./i2c_system_wrapper.bit --u-boot --force

Linux 폴더 안의 BOOT.BIN , image.ub 파일을 SD Card > BOOT 에 넣는다

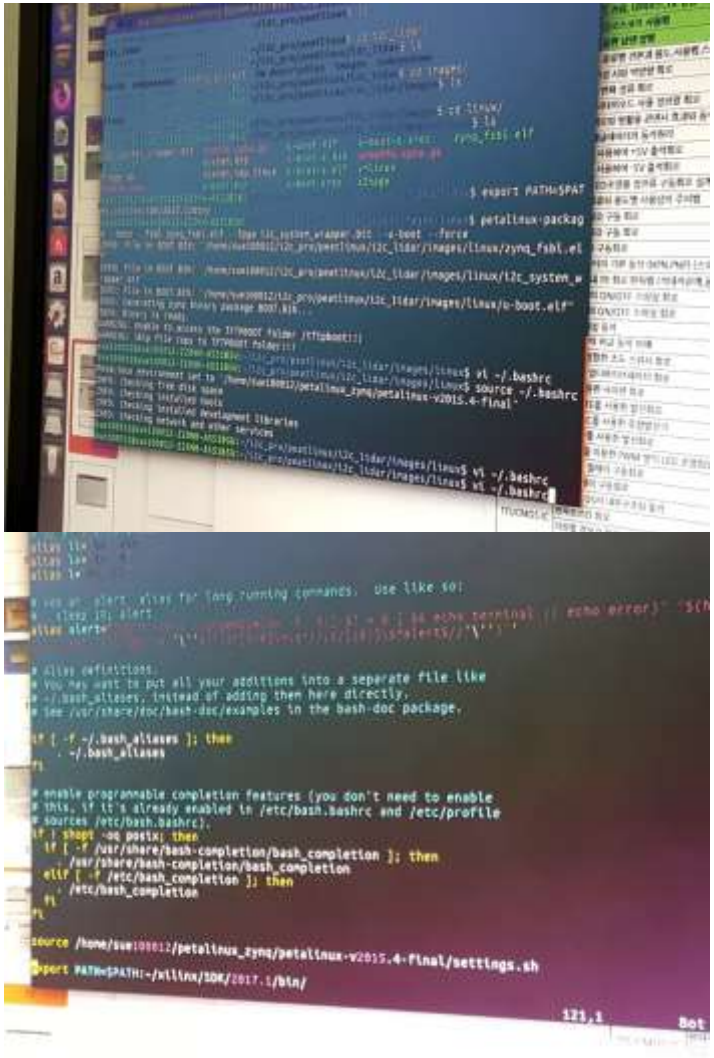
데이터시트:
https://support.garmin.com/support/manuals/manuals.htm?partNo=010-01722-00&language=en&country=US

[2] Bootgen error 해결하기 (경로 주의할 것)

$ Source ~/Xilinx/SDK/2017.1/settings64.sh

$ Source ~/Xilinx/Vivado/2017.1/settings64.sh

$ petalinux-package --boot --fsbl zynq_fsbl.elf --fpga ~/lab6/hardware/driver_lab.runs/impl_1/system_wrapper.bit –u-boot



[3] i2c_lidar.c

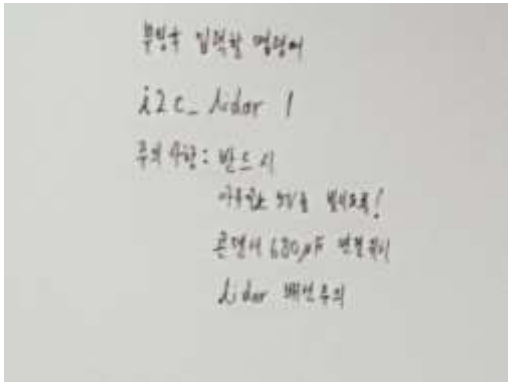
The simplest method of obtaining measurement results from the I2C interface is as follows:

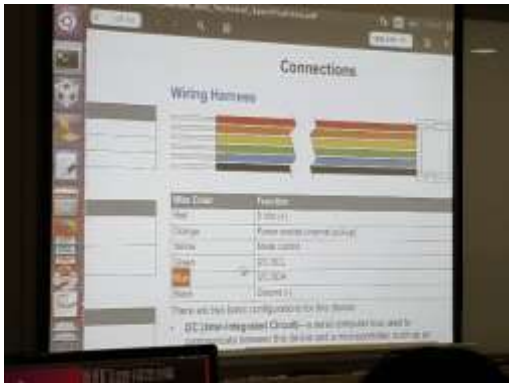
1. Write 0x04 to register 0x00.
2. Read register 0x01. Repeat until bit 0 (LSB) goes low.
3. Read two bytes from 0x8f (High byte 0x0f then low byte 0x10) to obtain the 16-bit measured distance in centimeters.

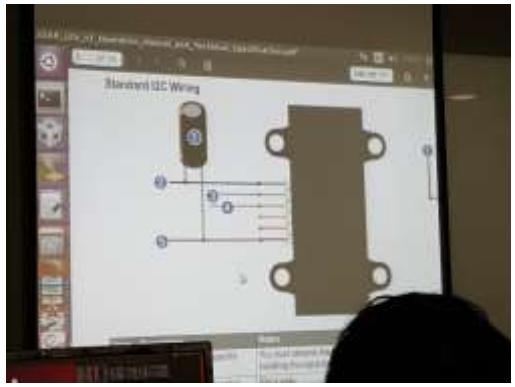
```
#include <stdio.h>
#include <linux/i2c.h>
#include <linux/i2c-dev.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>

#define I2C_FILE_NAME_0      "/dev/i2c-0"
#define I2C_FILE_NAME_1 "/dev/i2c-1"
#define LIDAR_SLAVE_ADDR 0x62   // (slave address = lidar의 i2C주소) 2p

#define ACQ_COMMAND      0x00
#define STATUS           0x01
#define SIG_COUNT_VAL    0x02
#define ACQ_CONFIG_REG   0x04
#define THRESHOLD_BYPASS 0x1C
#define READ_FROM        0x8f    // 0x89 에서도 결과값이 나왔다. 이유는 모르겠다.
/*4p 에서 보면 0x8f에서 2바이트를 읽으면 거리값이 된다고 쓰여있다 Read two bytes from 0x8f
(High byte 0x0f에 들어있고 then low byte 0x10에 들어있다) to obtain the
16-bit measured distance in centimeters */

#define NO_CORRECTION    0
#define CORRECTION       1

#define AR_VELOCITY      0
#define AR_PEAK_CORR     1
#define AR_NOISE_PEAK    2
#define AR_SIGNAL_STRENGTH    3
#define AR_FULL_DELAY_HIGH    4
#define AR_FULL_DELAY_LOW     5

#define OUTPUT_OF_ALL    0
#define DISTANCE_ONLY    1
#define DISTANCE_WITH_VELO    2
#define VELOCITY_ONLY    3

#define USAGE   "i2c_LIDAR_fn <OUTPUT_OPTIONS> <I2C_DEVICE_NUMBER>₩n"₩
                "<OUTPUT_OPTIONS>₩n"₩
                "0 : output of all₩n"₩
                "1 : distance of all₩n"₩
                "2 : distance with velocity₩n"₩
                "3 : velocity only₩n"

unsigned get_status();
void i_read(unsigned char, unsigned, unsigned char *);
void i_write(unsigned char reg, unsigned char value);
void measurement(unsigned char, unsigned char, unsigned char *);
void display(unsigned char, unsigned char *);

int fd = 0;

int main(int argc, char *argv[])
{
    unsigned char receives[8] = {AR_VELOCITY, 0, 0, AR_PEAK_CORR, AR_NOISE_PEAK,
AR_SIGNAL_STRENGTH, AR_FULL_DELAY_HIGH, AR_FULL_DELAY_LOW};
    unsigned char i, options;
    char *file_name = NULL;

    if(argc < 2)    printf("%s₩n", USAGE); // when Error
    else if(argc > 2 && atoi(argv[2])) file_name = I2C_FILE_NAME_1;
    else file_name = I2C_FILE_NAME_0;

    options = atoi(argv[1]);
```

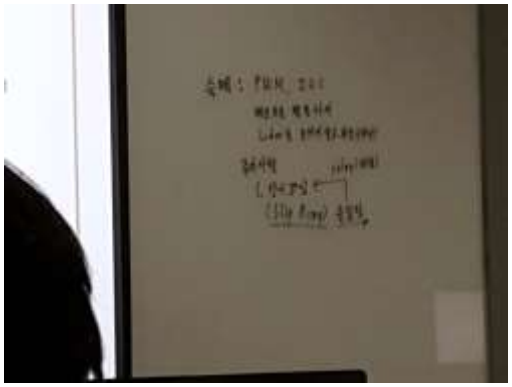
```
    if((fd = open(file_name, O_RDWR)) < 0) {
        perror("---OPEN DEVICE ERROR ");
        return -1;
    }

/* ioctl함수 : 디바이스 드라이버 제어, 하드웨어를 제어하거나 상태를 얻어올 때 사용한다.
http://damduc.tistory.com/216 */
    if(ioctl(fd, I2C_SLAVE, LIDAR_SLAVE_ADDR) < 0)   {
        perror("---SLAVE ADDR CONNECT ERROR ");
        return -1;
    }

    i_write(SIG_COUNT_VAL, 0x80); // 피크값을 찾기 위한 최대 카운트 개수 설정
    i_write(ACQ_CONFIG_REG, 0x08); // 3번 비트 클리어시 최고치 측정이 예상될 때 측정을 멈춘다
    i_write(THRESHOLD_BYPASS, 0x00); // 민감도 설정

    while(1){
        measurement(CORRECTION, options, receives);
        for(i=0; i<99; i++) measurement(NO_CORRECTION, options, receives);
    }
    close(fd);
    return 0;
}

unsigned get_status()
{
    unsigned char buf[1] = {STATUS};
// buf의 내용을 fd에 쓴다 1바이트
    if(write(fd, buf, 1) != 1) {
        perror("---WRITE REGISTER ERROR ");
        return -1;
    }
// fd의 내용을 buf에 읽어온다. 1바이트
    if(read(fd, buf, 1) != 1)
    {
        perror("---READ REGISTER ERROR ");
        return -1;
    }
    return buf[0] & 0x01; // buf[0]의 하위 4비트만 가져온다
}

// 시스템 프로그래밍에서 배운 read, write 함수가 아니라 커널 드라이빙을 통해 만든 함수!
void i_read(unsigned char reg, unsigned read_size, unsigned char *receives){
    unsigned char buf[1] = {reg};
    unsigned busy_flag = 1, busy_counter = 0;

    while(busy_flag) // flag 1이면 1: Device is busy taking a measurement    {
        busy_flag = get_status();
        busy_counter++;
        if(busy_counter > 9999)          {
            printf("BUSY COUNT TIME OUT !₩n");
            return ;
        }
    }
 // Busy Flag 0: Device is ready for new command 이면 slave(lidar)에 정보를 보낸다. 그런데 에러이면 다음을 실행
    if(!busy_flag) {
        if(write(fd, buf, 1) != 1) // slave에 정보를 보낸다 {
            perror("---WRITE REGISTER ERROR ");
            return -1;
        }
        if(read(fd, receives, read_size) != read_size)    {
            perror("---WRITE REGISTER ERROR ");
            return -1;
        }
    }
}

void i_write(unsigned char reg, unsigned char value) {
    unsigned char buf[2] = {reg, value};
    if(write(fd, buf, 2) != 2)     {
        perror("---WRITE REGISTER ERROR ");
        return -1;
```

```
    }
    usleep(1000);
}

void measurement(unsigned char is_correction, unsigned char options, unsigned char *buf){
    unsigned char i;
        if(is_correction)
            i_write(ACQ_COMMAND, 0x04); //Write 0x04: with receiver bias correction
    else
            i_write(ACQ_COMMAND, 0x03); //Write 0x03: without receiver bias correction

    i_read(READ_FROM, 8, buf); // 0x8f 에서 1바이트 읽어온다
    for(i=1; i<6; i++)   // 0x8f에 물려있어야 하는데 뒤로 밀려서 6비트???
            buf[i] = buf[i + 2];
    display(options, buf);
}

void display(unsigned char options, unsigned char *buf){
            unsigned char i;
            char *strings[5] = {"Velocity", "Peak value in correlation record", "Correlation record noise
floor", "Received signal strength", "Distance"};

            buf[AR_FULL_DELAY_HIGH] = buf[AR_FULL_DELAY_HIGH] << 8 | buf[AR_FULL_DELAY_LOW];
// buf[4] << 8 | buf[5]
// 16비트 받아오니까앞을 8비트 밀고 뒤의 값을 or 해서 실제값을 읽어온다

switch(options)
            {
                        case OUTPUT_OF_ALL :
                                    for(i=0; i<5; i++)
                                                printf("%s ₩t₩t₩t₩t = %d₩n", strings[i], buf[i]);
                                    break;
                        case DISTANCE_ONLY : // 1
                                    printf("%s ₩t₩t₩t₩t = %d₩n", strings[4], buf[AR_FULL_DELAY_HIGH]); //4
                                    break;
                        case DISTANCE_WITH_VELO :
                                    printf("%s ₩t₩t₩t₩t = %d₩n", strings[0], buf[AR_VELOCITY]); //0
                                    printf("%s ₩t₩t₩t₩t = %d₩n", strings[4], buf[AR_FULL_DELAY_HIGH]);
                                    break;
                        case VELOCITY_ONLY :
                                    printf("%s ₩t₩t₩t₩t = %d₩n", strings[0], buf[AR_VELOCITY]);//0
                                    break;
            }
            printf("₩n");
}
/*                  AR_VELOCITY        0
                    AR_PEAK_CORR       1
                    AR_NOISE_PEAK      2
                    AR_SIGNAL_STRENGTH         3
                    AR_FULL_DELAY_HIGH         4
                    AR_FULL_DELAY_LOW          5            */
```









Project Overview

1. MCU -> MCU용 전원회로
PID제어(속도, 위치)
CAN 통신
*. 안드로이드 원격 제어기(컨트롤러 만들기 = CAN통신, 네트워크 프로그래밍 할것 보드랑)
*. CAN port 2개 있음 만들필요 없음

2. DSP -> 12V 5A 전원회로 (쌤꺼 참고)
영상처리 (차선, 장애물, 교통신호 식별)
CAN 통신, 네트워크 프로그래밍 , 안드로이드 원격 제어기는 MCU에 넘김

3. FPGA -> 5V 2.5A
모터, 라이다 제어하고 값받음(속도, 위치, 각 방위별), 충돌위험감지
각 방위별 작동 위치정보 파악해야 함, CAN통신(CAN 포트 만들거나 사서 쓴다 Pmod CAN)

4. 회로
전원회로**, 사이렌회로, etc

5. 슬립링