Embedded Linux on Zynq using Vivado Lab5

Innova Lee(이상훈)
gcccompil3r@gmail.com

Introduction

많은 유형의 장치에서 인터럽트를 처리하고 장치의 메모리 공간에 대한 접근을 제공해야 하는 경우가 자주 있다.
장치가 커널이 제공하는 다른 자원을 활용할 필요가 없다면 장치를 제어하는 논리가 커널 내에 있을 필요가 없다.

이 Lab 에서는 사용자 공간에서 HW 에 직접 접근하는 두 가지 방법을 살펴본다:
/dev/mem 을 통합 직접 접근, 사용자 공간 I/O (UIO) Framework 에 해당한다.

커널 드라이버를 작성하는 것은 일부 장치에서는 과도한 작업이며
커널 코드를 작성해야 하기 때문에 개발 프로세스가 더 복잡하다.
이 Lab 세션에서는 처음으로 간단한 UIO 드라이버를 만들고 Linux 에서 모듈을 로드하는 방법을 학습한다.

Objectives

이 Lab 을 완료하명 아래를 수행 할 수 있다.

- 사용자 공간에 직접 HW 장치 접근
- UIO 프레임워크를 사용하여 HW 장치에 접근함
- 커널 모듈 로드 및 언로드 경험

Preparation

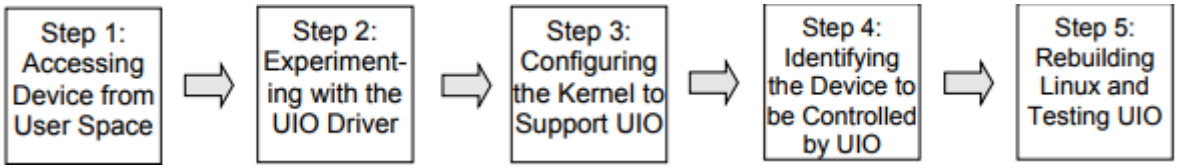
이것이 첫 번째 Lab 인 경우 환경 설정 방법에 대한 필수 준비 정보는 Lab 1 의 "Before You Start" 섹션을 참조하라.

워크 스테이션이 다시 시작되거나 로그 아웃되면 아래 명령을 실행하여 Host 에서 DHCP 서버를 시작한다:

```
[host]$ sudo service isc-dhcp-server restart
```

자세한 내용은 Lab 1 의 "Initializing the Workshop Environment" 섹션을 참조하라.

General Flow for this Lab



Accessing the Device from User Space

1-1. 경로를 프로젝트 디렉토리로 변경한다.

1-1-1. 아래 명령을 실행하여 프로젝트 디렉토리 경로를 만들고 변경한다.

```
[host]$ mkdir ~/emblnx/labs/lab5
[host]$ cd ~/emblnx/labs/lab5
```

1-2. petalinux-create 명령을 사용하여 새로운 임베디드 리눅스 플랫폼을 만들고 플랫폼을 선택한다.

1-2-1. 아래 명령을 실행하여 새로운 Petalinux 프로젝트를 만든다.

```
[host]$ petalinux-create -t project -s /opt/pkg/ZYBO_petalinux_v2015_4-final.bsp
```

이 명령은 소프트웨어 프로젝트 디렉토리를 만든다:
~/emblnx/labs/lab5 밑에 ZYBO_petalinux_v2015_4 에 해당한다.

1-2-2. 디렉토리를 Petalinux 프로젝트로 변경한다:
~/emblnx/labs/lab5/ZYBO_petalinux_v2015_4 에 해당한다.

1-3. 사용자 공간에서 GPIO 장치에 접근 할 수 있는 새로운 사용자 응용 프로그램을 만든다.

PetaLinux 도구를 사용하여 C 또는 C++ 용 사용자 응용 프로그램 템플릿을 작성할 수 있다.
이러한 템플릿에는 응용 프로그램 소스 코드 및 메이크 파일이 포함되어 있으므로
Target 에 맞게 응용 프로그램을 쉽게 구성하고 컴파일하여 루트 파일 시스템에 설치할 수 있다.

1-3-1. 아래 명령을 입력하여 PetaLinux 프로젝트 내에 새로운 사용자 응용 프로그램을 생성한다:

[host]$ petalinux-create -t apps --name gpio-dev-mem-test

새로 만든 응용 프로그램은 <projectroot>/components/apps/gpio-dev-mem-test 디렉토리에서 찾을 수 있다.
여기서 <project-root> 는 ~/emblnx/labs/lab5/ZYBO_petalinux_v2015_4 에 해당한다.

1-4. sources/lab5/gpiodev-mem-test 디렉토리에서 gpio-dev-mem-test 소스를 복사한다.

1-4-1. 새로 생성 된 응용 프로그램 디렉토리로 변경한다.

[host]$ cd <project-root>/components/apps/gpio-dev-mem-test

1-4-2. sources/lab5 디렉토리의 기존 소스로 기존 gpio-dev-mem-test.c 파일을 대체한다.

[host]$ cp ~/emblnx/sources/lab5/gpio-dev-mem-test/*.c ./

1-4-3. 사용자 공간에서 장치에 접근하는 방법을 보려면 gpio-dev-mem-test.c 소스 코드를 검토한다.

/dev/mem 은 전체 시스템의 메모리 맵을 나타내는 가상 파일이다.
mmap() 을 사용하여 장치를 메모리에 맵핑한다.
그런 다음 맵핑 된 메모리를 가리키는 포인터를 사용하여 장치에 접근 할 수 있다.

1-5. 빌드 프로세스에 포함될 새 애플리케이션을 선택하라.
응용 프로그램은 기본적으로 활성화되어 있지 않다.

1-5-1. 프로젝트 디렉토리에 있는지 확인한다.
즉 ~/emblnx/labs/lab5/ZYBO_petalinux_v2015_4 에 해당한다.

1-5-2. 아래 명령을 입력하여 rootfs 구성 메뉴를 실행한다.

[host]$ petalinux-config -c rootfs

1-5-3. 아래쪽 화살표 키를 눌러 메뉴를 Apps(1)로 스크롤 한다.

1-5-4. 앤터 키를 눌러 Apps 하위 메뉴로 이동한다.
새로운 응용 프로그램 gpio-dev-mem-test 가 메뉴에 나열된다.

1-5-5. gpio-dev-mem-test 로 이동하고 <Y> 를 클릭하여 application(2) 를 선택한다.

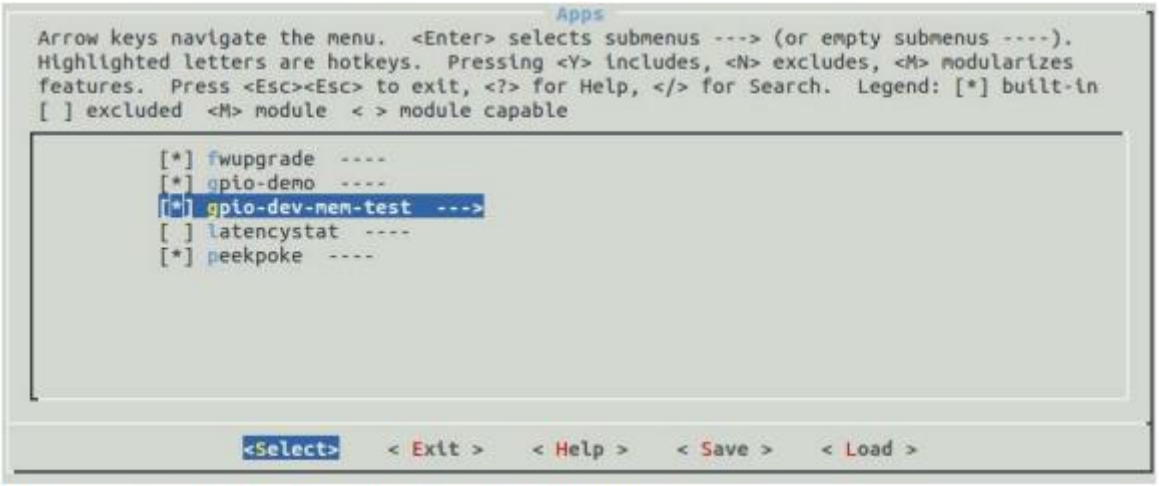


Figure 1. Selecting the gpio-dev-mem-test application

1-5-6. 메뉴를 종료하고 <Yes> 를 선택하여 새 구성을 저장한다.

1-6. 이미지를 빌드한다.

1-6-1. 아래 명령을 입력하여 이미지를 빌드한다.

[host]$ petalinux-build

1-7. BOOT.BIN 파일을 미리 만들어진 디렉토리에서 SD 카드로 복사한다.

1-7-1. 사전 빌드 된 BOOT.BIN 파일이 SD 카드에 있는지 확인한다.

1-7-2. Lab 2, Lab 3, Lab 4 를 마지막 Lab 으로 사용한 경우 SD 카드를 변경할 필요가 없다.

1-7-3. 그렇지 않으면 ~/emblnx/labs/lab5/ZYBO_petalinux_v2015_4/pre-built/linux/images 디렉토리에서
BOOT.BIN 을 SD 카드로 복사한다.

1-8. 이미지를 다운로드 하려면 Host 에서 DHCP 서버를 실행한다.

1-8-1. DHCP 서버를 실행한다.

[host]$ sudo service isc-dhcp-server restart

1-9. 보드의 전원을 켜고 Serial 포트 터미널을 설정한다.

1-9-1. 보드의 전월을 켠다.

1-9-2. /dev/ttyUSB1 이 read/write 접근으로 설정되어 있는지 확인한다.

#sudo chmod 666 /dev/ttyUSB1

1-9-3. GtkTerm 프로그램을 시작한다.
부팅 정보를 다시 보기 위해 보드(BTN7)를 reset 할 수 있다.

1-10. 네트워크를 통해 새로운 임베디드 리눅스 이미지를 부팅한다.

1-10-1. GtkTerm 창에서 부팅 프로세스를 본다.

1-10-2. GtkTerm 창에 메시지가 나타나면 아무 키나 눌러 자동 부팅을 중지하라.

1-10-3. uboot 부팅 중 "DHCP client bound to address" 메시지가 표시되지 않으면
dhcp 를 실행하여 IP 주소를 얻어야 한다.

U-Boot-PetaLinux> dhcp

1-10-4. u-boot 콘솔에서 아래 명령을 실행하여 TFTP 서버 IP 를 Host IP 로 설정한다.

U-Boot-PetaLinux> set serverip 192.168.1.1

1-10-5. u-boot 콘솔에서 아래 명령을 실행하여 TFTP 를 사용하여 새 이미지를 다운로드하고 부팅한다.

U-Boot-PetaLinux> run netboot

이 명령은 Host 의 /tftpboot 에서 ARM Cortex-A9 MPcore 시스템의 주 메모리로
image.ub 를 다운로드하고 이 이미지로 시스템을 부팅한다.

1-10-6. GtkTerm 창에서 Linux 부팅을 본다.

1-11. 로그인 하여 gpio-dev-mem-test 프로그램을 실행한다.

1-11-1. 리눅스에 로그인하면 gpio-dev-mem-test 명령을 사용하여 GPIO 장치에 직접 접근한다.

> **이 예에서 GPIO 주변 장치의 실제 주소를 알아야 한다.**
> **모든 하드웨어 정보가 DTS(장치 트리 소스) 파일에 있으므로**
> **DTS 파일을 검사하여 GPIO 장치의 실제 주소 정보를 얻을 수 있다.**

1-11-2. ~/emblnx/labs/lab5/ZYBO_petalinux_v2015_4/subsystems/linux/configs/device-tree 디렉토리로 이동한다.

1-11-3. gedit 을 사용하여 pl.dtsi 파일을 연다.

1-11-4. DTS 파일에서 "LEDs_4bits" 를 검색하여 LED GPIO 의 물리적 주소를 얻는다.

장치 노드의 "reg" 속성의 첫 번째 인수는 장치의 물리적 시작 주소다.
예:

reg = <0x41220000 0x10000>;
"0x41220000" 은 LED GPIO 장치의 물리적 시작 주소이고 "0x10000" 은 주소 범위에 해당한다.

1-11-5. 실제 주소를 얻은 후 gpio-dev-mem-test 명령을 실행하여 LED GPIO 에 접근할 수 있다.

```
root@ZYBO_petalinux_v2015_4:~# gpio-dev-mem-test
GPIO access through /dev/mem.
GPIO physical address is required.
*argv[0] -g <GPIO_ADDRESS> -i|-o <VALUE>
    -g <GPIO_ADDR>   GPIO physical address
    -i               Input from GPIO
    -o <VALUE>       Output to GPIO
root@ZYBO_petalinux_v2015_4:~#
```

Figure 2. Running gpio-dev-mem-test

Xilinx GPIO 주변 장치에는 2 개의 레지스터가 있다.

- **오프셋 0x0 은 GPIO 포트를 읽거나 쓰는데 사용되는 데이터 레지스터다.**
- **오프셋 0x4 는 방향 레지스터로서 각 GPIO 비트가 입력(해당 방향 비트가 '1') 또는 출력(방향 비트가 '0')으로 구성되어야 하는지 여부를 나타내는데 사용된다.**

예를 들어 LED GPIO 의 물리적 시작 주소가 0x41220000 이라고 가정하고
LED GPIO 에 값을 쓰려면 아래를 수행해야 한다.

```
#gpio-dev-mem-test -g 0x41220000 -o 15
#gpio-dev-mem-test -g 0x41220000 -o 0
```

푸쉬 버튼과 DIP 스위치의 GPIO 에서 읽으려면
DIP 스위치의 GPIO 물리적 시작 주소가 0x41200000 이고 푸쉬 버튼이 0x41210000 라고 가정한다.

```
#gpio-dev-mem-test -g 0x41200000 -i
#gpio-dev-mem-test -g 0x41210000 -i
```

```
root@ZYBO_petalinux_v2015_4:~# gpio-dev-mem-test -g 0x41220000 -o 15
GPIO access through /dev/mem.
root@ZYBO_petalinux_v2015_4:~# gpio-dev-mem-test -g 0x41220000 -o 0
GPIO access through /dev/mem.
root@ZYBO_petalinux_v2015_4:~# gpio-dev-mem-test -g 0x41200000 -i
GPIO access through /dev/mem.
gpio dev-mem test: input: 00000005
root@ZYBO_petalinux_v2015_4:~# gpio-dev-mem-test -g 0x41200000 -i
GPIO access through /dev/mem.
gpio dev-mem test: input: 00000001
root@ZYBO_petalinux_v2015_4:~# gpio-dev-mem-test -g 0x41210000 -i
GPIO access through /dev/mem.
gpio dev-mem test: input: 00000002
```

Figure 3. Running gpio-dev-mem-test for GPIO peripherals

1-11-6. 실제 시스템을 기반으로 GPIO 의 물리적 시작 주소를 조정한다.

푸쉬 버튼에 연결된 GPIO 에서 읽는다.
DTS 파일에서 얻은 적절한 물리적 주소만 사용한다.

Experimenting with the UIO Driver

이 섹션에서는 UIO 드라이버를 만든다.
이 예제에서는 사용자 정의 커널 코드가 전혀 필요없는 향상된 일반 UIO 프레임워크를 사용하고 있다.

2-1. 사용자 공간에서 GPIO 장치에 접근 할 수 있는 새로운 사용자 응용 프로그램을 만든다.

2-1-1. Petalinux 프로젝트 디렉토리에 있는지 확인한다.
즉 ~/emblnx/labs/lab5/ZYBO_petalinux_v2015_4 를 의미한다.

2-1-2. 아래 명령을 사용하여 PetaLinux 프로젝트 내에 새로운 사용자 응용 프로그램을 생성한다.

[host]$ petalinux-create -t apps --name gpio-uio-test

새로 만든 응용 프로그램은 <projectroot>/components/apps/gpio-uio-test 디렉토리에서 찾을 수 있다.
여기서 <projectroot> 는 ~/emblnx/labs/lab5/ZYBO_petalinux_v2015_4 에 해당한다.

2-2. sources/lab5/gpio-uiotest 디렉토리에서 gpio-uio-test 소스를 복사한다.

2-2-1. 새로 생성 된 응용 프로그램 디렉토리로 변경한다.

[host]$ cd <project-root>/components/apps/gpio-uio-test

2-2-2. 기존 gpio-uio-test.c 를 sources 디렉토리의 완성된 소스로 바꾼다.

[host]$ cp ~/emblnex/sources/lab5/gpio-uio-test/*.c ./

2-2-3. 사용자 공간에서 장치에 접근하는 방법을 보려면 gpio-uio-test.c 파일을 검토한다.

UIO 장치는 파일 시스템에 /dev/uioX 로 표시된다.
UIO 를 통해 장치에 접근하려면 /dev/uioX 를 연 다음
mmap() 을 사용하여 장치를 응용 프로그램의 주소 공간에 맵핑해야 한다.
그런 다음 mmap() 호출에서 반환된 포인터를 사용하여 장치에 접근할 수 있다.

2-3. 빌드 프로세스에 포함될 새 응용 프로그램을 선택한다.
응용 프로그램은 기본적으로 활성화되어 있지 않다.

2-3-1. 프로젝트 디렉토리에 있는지 확인한다.
즉 ~/emblnx/labs/lab5/ZYBO_petalinux_v2015_4 를 의미한다.

2-3-2. 아래 명령을 입력하여 rootfs 구성 메뉴를 실행한다.

[host]$ petalinux-config -c rootfs

2-3-3. 아래쪽 화살표 키를 눌러 메뉴를 Apps 로 스크롤 한다.

2-3-4. 앤터 키를 눌러 Apps 하위 메뉴로 이동한다.

2-3-5. 새 응용 프로그램 gpio-uio-test 가 메뉴에 나열된다.

2-3-6. gpio-uio-test 로 이동하고 <Y> 를 클릭하여 응용 프로그램을 선택한다.

2-3-7. 메뉴를 종료하고 <Yes> 를 선택하여 새 구성을 저장한다.

구성 변경 사항이 적용되는데 몇 초가 걸린다.
명령 콘솔에서 쉘 프롬프트로 돌아갈 때까지 기다린다.

Configuring the Kernel to Support UIO

장치의 제어 논리를 사용자 공간에 완전히 구현할 수 있지만 커널에서 UIO 프레임워크를 활성화해야 한다.
로드 가능한 모듈로 빌드되도록 UIO 하위 시스템을 구성한다.
그러나 원하는 경우 직접 커널에 빌드하는 것도 가능하다.

3-1. UIO 를 지원하도록 커널을 구성한다.

3-1-1. 프로젝트 디렉토리에 있는지 확인한다.

3-1-2. 터미널에 아래 명령을 입력한다.

[host]$ petalinux-config -c kernel

3-1-3. 리눅스 커널 구성 메뉴에서 Device Drivers 를 선택한다.

3-1-4. Device Drivers 메뉴에서 Userspace I/O drivers 로 스크롤하여 "<M>" 을 선택한다.

<M> Userspace I/O drivers --->

커널은 기본적으로 로드 가능한 모듈을 지원하도록 구성되어 있으므로
로드 가능한 장치 드라이버의 경우 기본 제공 또는 모듈로 선택할 수 있다.
"<*>" 은 내장형을 의미하고 "<M>" 은 모듈을 의미한다.
드라이버를 모듈로 선택하면 Linux 를 부팅 할 때 로드 되지 않는다.
modprobe 명령을 사용하여 Linux 부팅 후 로드 할 수 있다.(아래 참조)

UIO 커널 드라이버는 내장 또는 모듈로 선택할 수 있다.
이 실습에서 나중에 Linux 에서 모듈을 로드하는 것을 실험하기 위해 모듈로 선택한다.

3-1-5. Userspace I/O Drivers 메뉴로 가서
Userspace I/O platform driver with generic IRQ handling 을 모듈로 선택한다:

--- Userspace I/O drivers
<M> Userspace I/O platform driver with generic IRQ handling
<M> Userspace I/O platform driver with generic irq and dynamic memory

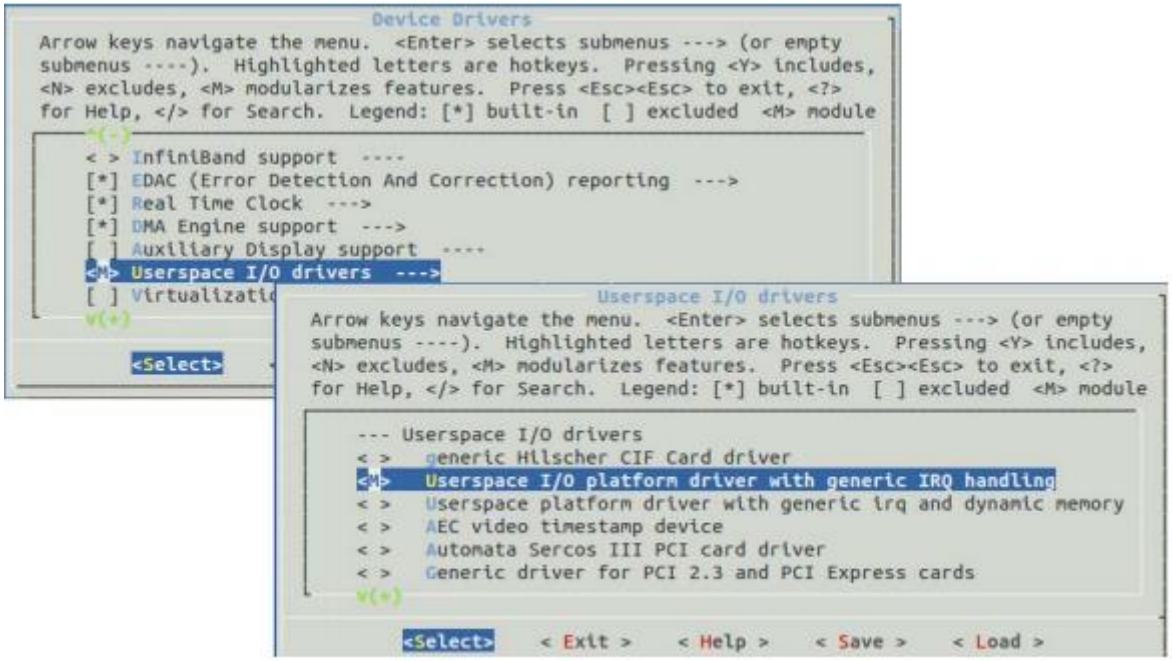


Figure 4. Selecting UserIO drivers as module

3-1-6. 커널을 종료하고 구성 변경 사항을 저장한다.

Identifying the Device to be Controlled by UIO

Lab 2 에서 배웠듯이 Device Tree(DTS) 의 장치 항목에 있는 호환 가능한 속성은 장치를 커널 드라이버에 연결한다.
일반 Xilinx GPIO 장치 드라이버 대신 LED GPIO 를 UIO 장치로 제어하도록 표시한다.

4-1. LED GPIO 를 UIO 장치로 제어하도록 한다.

4-1-1. ~/emblnx/labs/lab5/ZYBO_petalinux_v2015_4/subsystems/linux/configs/device-tree 디렉토리로 변경한다.

4-1-2. pl.dtsi 파일을 검토한다.
gedit 을 사용하여 pl.dtsi 파일을 열 수 있다.

4-1-3. GPIO 주변기기 btn_4bits, led_4Bits, 그리고 sw_4Bits 에 대한 항목을 찾을 수 있다.
이들은 "xlnx, xps-gpio-1.00.a" 를 호환 문자열로 가지고 있다.
이 문자열은 사용중인 드라이버가 Xilinx 출신이고 사용 된 드라이버가 xps-gpio-1.00.a 임을 나타낸다.

4-1-4. 이 장치에는 GPIO 드라이버가 아닌 UIO 드라이버를 사용한다.
따라서 호환 가능한 매개 변수 행을 compatible="generic-uio" 로 바꿔야 한다.
system-top.dts 파일에서 이 작업을 수행한다.

4-1-5. gedit 편집기에서 system-top.dts 파일을 열고 다음 페이지와 같이 코드를 추가한다.

4-1-6. 아래 페이지를 보고 내용을 추가한 이후 저장하고 편집기를 닫는다.

```
/dts-v1/;
/include/ "system-conf.dtsi"
/ {
};

&clkc {
        ps-clk-frequency = <50000000>;
};

&flash0 {
   compatible = "s25fl128s1";
};

&usb0 {
        dr_mode = "otg";
} ;

&gem0 {
        phy-handle = <&phy0>;
        mdio {
                #address-cells = <1>;
                #size-cells = <0>;
                phy0: phy@1 {
                        compatible = "realtek,RTL8211E";
                        device_type = "ethernet-phy";
                        reg = <1>;
                } ;
        } ;
} ;

&btn_4Bits {
        compatible = "generic-uio";
};

&led_4Bits {
        compatible = "generic-uio";
};

&sw_4Bits {
        compatible = "generic-uio";
};
```

Figure 5. Assigning generic-uio driver to the PL peripherals.

Rebuilding Linux and Testing UIO

커널을 구성하고 UIO 드라이버를 만들고 UIO 장치 노드를 만들었다.
이제 Linux 이미지를 재구성하고 UIO 를 테스트해야 한다.

5-1. Linux 이미지를 다시 빌드한다.

5-1-1. PetaLinux 프로젝트 디렉토리 ~/emblnx/labs/lab5/ZYBO_petalinux_v2015_4 로 변경한다.

5-1-2. 아래 명령을 입력하여 이미지를 빌드한다.

[host]$ petalinux-build

5-2. 보드의 전원을 켜고 직렬 포트 터미널을 설정한다.

5-2-1. 보드의 전원을 켠다.

5-2-2. /dev/ttyUSB1 이 read/write 접근으로 설정되어 있는지 확인한다.

#sudo chmod 666 /dev/ttyUSB1

5-2-3. GtkTerm 프로그램을 시작한다.
부팅 정보를 다시 보려면 보드(BTN7)을 reset 할 수 있다.

5-3. 네트워크를 통해 새로운 임베디드 리눅스 이미지를 부팅한다.

5-3-1. GtkTerm 창에서 부팅 프로세스를 본다.

5-3-2. GtkTerm 창에서 자동 부팅을 중지하려면 아무 키나 누른다.

5-3-3. uboot 부팅 중 "DHCP client bound to address" 메시지가 표시되지 않으면
dhcp 를 실행하여 IP 주소를 얻어야 한다.

U-Boot-PetaLinux> dhcp

5-3-4. u-boot 콘솔에서 아래 명령을 실행하여 TFTP 서버 IP 를 Host IP 로 설정한다:

U-Boot-PetaLinux> set serverip 192.168.1.1

5-3-5. u-boot 콘솔에서 아래 명령을 실행하여 TFTP 를 사용하여 새 이미지를 다운로드하고 부팅한다:

U-Boot-PetaLinux> run netboot

이 명령은 Host 의 /tftpboot 에서 ARM Cortex-A9 MPcore 시스템의 주 메모리로 image.ub 를 다운로드하고 이 이미지로 시스템을 부팅한다.

5-3-6. GtkTerm 창에서 Linux 부팅을 본다.

5-4. UIO 를 테스트한다.

5-4-1. 시스템에 로그인한다.

5-4-2. 이전에 커널을 구성할 때 UIO 를 모듈로 만들었으므로 UIO 모듈을 로드한다.

#modprobe uio
#modprobe uio_pdrv_genirq

5-4-3. lsmod 명령을 사용하여 로드 된 활성 모듈을 나열한다.

#lsmod

```
login[854]: root login on 'ttyPS0'
root@ZYBO_petalinux_v2015_4:~# modprobe uio
root@ZYBO_petalinux_v2015_4:~# modprobe uio_pdrv_genirq
root@ZYBO_petalinux_v2015_4:~# lsmod
    Not tainted
uio_pdrv_genirq 2846 0 - Live 0xbf006000
uio 6934 1 uio_pdrv_genirq, Live 0xbf000000
```

Figure 6. Entering the modprobe command

로드 된 UIO 모듈의 정보는 아래에서 찾을 수 있다:

#ls /sys/class/uio/

UIO 의 이름과 UIO 의 주소 정보는 /sys/class/uio/uioX 에서 찾을 수 있다.

5-4-4. mdev -s 를 실행하여 /dev/uioX 가 UIO 장치를 올바르게 나타내도록 하라.

#mdev -s

mdev 명령은 /sys/class/* 에 있는 장치의 장치 파일을 /dev 에 자동으로 만든다.

5-4-5. gpio-uio-test 명령을 사용하여 LED GPIO 장치에 직접 접근한다.
예를 들어 LED GPIO 에 15 를 쓴다.

#gpio-uio-test -d /dev/uio1 -o 15

5-4-6. DIP 스위치 읽기 GPIO:

#gpio-uio-test -d /dev/uio2 -i

```
root@ZYBO_petalinux_v2015_4:~# ls /sys/class/uio/
uio0  uio1  uio2
root@ZYBO_petalinux_v2015_4:~# mdev -s
root@ZYBO_petalinux_v2015_4:~# gpio-uio-test -d /dev/uio1 -o 15
GPIO UIO test.
root@ZYBO_petalinux_v2015_4:~# gpio-uio-test -d /dev/uio2 -i
GPIO UIO test.
gpio-uio-test: input: 00000005
```

Figure 7. Using gpio-uio-test to turn on all LEDs

5-4-7. 작업이 완료되면 보드의 전원을 끄고 GtkTerm 창을 닫는다.

Conclusion

이 Lab 에서는 아래를 수행하는 방법을 배웠다:

- 사용자 공간에서 장치 접근
- UIO 드라이버 만들기
- 모듈 로드
- 모듈 정보 찾기

Completed Solution

솔루션을 실행하려면 labsolution/lab4/SDCard 디렉토리의 BOOT.bin 을 SD 카드에 복사한다.
SD 카드를 Zybo 에 넣는다.
SD 카드 부팅 모드에서 Zybo 를 설정한다.
이더넷 케이블을 사용하여 Zybo 를 Host 컴퓨터에 연결한다.

아래 명령을 실행하여 Host 에서 DHCP 서버를 시작한다.

[host]$ sudo service isc-dhcp-server restart

labsolution/lab4/tftpboot 디렉토리의 image.ub 파일을 /tftpboot 디렉토리로 복사한다.

보드의 전원을 켠다.
터미널 세션을 설정한다.
자동 부팅 메시지가 표시되면 부팅 프로세스를 중단한다.
Target Board 터미널 창에서 아래 명령을 사용하여 server ip 주소를 설정한다.

#set serverip 192.169.1.1

netboot 명령을 실행한다.

#run netboot

시스템에 로그인하고 Lab 을 테스트한다.

References

https://www.xilinx.com/support/university/vivado/vivado-workshops/Vivado-embedded-linux-zynq.html