Embedded Linux on Zynq using Vivado Lab1

Innova Lee(이상훈)
gcccompil3r@gmail.com

Introduction

임베디드 리눅스는 임베디드 시스템에서 리눅스 운영체제를 사용한다.
Linux 의 데스크톱 및 서버 버전과 달리 Linux 의 임베디드 버전은 비교적 제한된 리소스를 가진 장치 전용으로 설계되었다.
Xilinx 의 Zynq All Programmable SoC 에 사용되는 ARM Cortex-A9 프로세서는 임베디드 리눅스를 지원한다.
이 워크숍의 대부분의 Lab 에서는 ARM Cortex-A9 MPcore 에서
PetaLinux 도구를 사용하여 Build 된 임베디드 리눅스를 실행한다.

이 첫 번째 Lab 은 임베디드 리눅스와 워크숍에서 사용할 개발 보드에 대한 기본적인 Introduction 이다.
여기서 다루는 기본 내용은 후에 Lab 세션을 통해 반복적으로 사용될 것이다.

Objectives

이 Lab 을 완료하면 아래를 수행할 수 있다.

- 워크숍에서 사용 된 개발 보드의 전원을 켠다.
- Zynq Linux 시스템에 로그인한다.
- 임베디드 리눅스와 데스크톱 리눅스 환경을 비교한다.

Typographic Conversions

개발(데스크톱) 워크 스테이션에서 실행될 명령은 아래와 같다:

[host]$ 명령 및 매개 변수

ARM 프로세서 Linux Target 에서 실행될 명령은 아래와 같다:

리눅스 앱 실행

Before You Start

시작하기 전에 아래 사항들을 확인한다:

* 전원 스위치가 꺼짐 위치에 있다.
* Micro USB 케이블은 보드의 PROG UART 커넥터와 PC 사이에 연결된다.
* 보드는 USB 로 전원이 공급되도록 점퍼 셋팅이 되어 있다.
* 개발 보드의 이더넷 포트를 데스크톱(호스트) 컴퓨터의 이더넷 포트에 연결한다.
* BOOT.BIN 및 image.ub 파일이 ~/emblnx/sources/lab1/SDCard 디렉토리에서 복사된다.
* Micro SD 카드가 Target Board 에 다시 삽입된다.
* 아래 그림과 같이 Micro SD 카드에서 부팅하도록 점퍼를 설정한다.

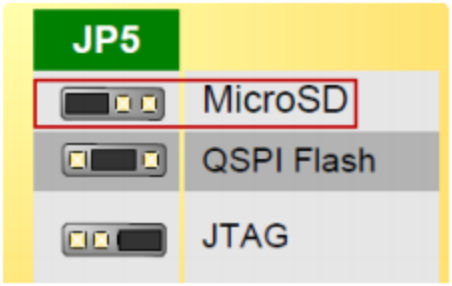


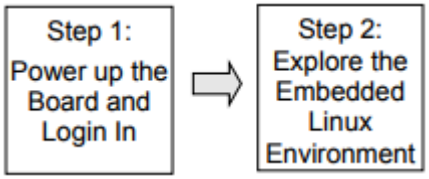
Figure 1. SD Card Boot - jumper settings

Initializing the Workshop Environment

기본적으로 우분투 이미지는 이미 사용자를 위해 워크숍 환경을 설정했다.
워크 스테이션이 다시 시작되거나 로그 아웃되면 다음 명령을 실행하여 Host 에서 DHCP 서버를 시작하라.

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

General Flow for this Lab



Power up the Board and Log in

1-1. Board 의 전원을 키고 Host 에서 DHCP 서버를 실행한다.

1-1-1. micro SD 카드의 ~/emblnx/sources/lab1/SDCard 디렉토리에서 BOOT.BIN 및 image.ub 파일을 복사한다.

1-1-2. micro SD 카드를 보드에 넣는다.

1-1-3. 보드의 전원을 킨다.

1-1-4. DHCP 서버를 실행한다.

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

1-2. Serial 포트 터미널을 설정한다.

1-2-1. /dev/ttyUSB1 이 read/write 접근으로 설정되었는지 확인한다.

[host]$sudo chmod 666 /dev/ttyUSB1

1-2-2. 대시 보드의 검색 필드에 Serial 포트를 입력한다.

1-2-3. 데스크탑에서 Serial 포트 터미널 응용 프로그램을 선택한다.

1-2-4. 보드를 리셋(BTN7)하여 부팅 정보를 확인한다.
보드가 부트 프로세스를 진행할 때 GtkTerm(Serial 포트) 콘솔을 보라.
보드 콘솔에는 아래와 비슷한 메시지가 있다.

```
U-Boot 2015.07 (Jan 21 2016 - 07:27:49 +0000)

DRAM:  ECC disabled 512 MiB
MMC:   zynq_sdhci: 0
SF: Detected S25FL128S_64K with page size 256 Bytes, erase size 64 K
iB, total 16 MiB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
U-BOOT for ZYBO_petalinux_v2015_4

Hit any key to stop autoboot:  2
Freeing unused kernel memory: 3028K (c0659000 - c094e000)
mmc0: new high speed SDHC card at address 0007
mmcblk0: mmc0:0007 SD4GB 3.70 GiB
 mmcblk0: p1
INIT: version 2.88 booting
FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
Creating /dev/flash/* device nodes
random: dd urandom read with 1 bits of entropy available
Starting internet superserver: inetd.
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
 Removing any system startup links for run-postinsts ...
INIT: Entering runlevel: 5
Configuring network interfaces... done.

Built with PetaLinux v2015.4 (Yocto 1.8) ZYBO_petalinux_v2015_4 /dev/ttyPS0
ZYBO_petalinux_v2015_4 login:
```

Figure 2. Linux booting process in the board console

Exploring the Embedded Linux Environment

2-1. 부팅 메시지와 기본 Linux 명령을 탐색한다.

2-1-1. 터미널 창을 위로 스크롤하고 부팅 로그를 검토한다.
기존 리눅스 사용자는 출력을 인식해야 한다.
image.ub 로딩, USB, SD 등의 드라이버 로딩 및 uWeb 서버 시작이 표시된다.

2-1-2. 로그인과 암호로 root 를 입력하여 로그인한다.

2-1-3. 다음 15 분 동안 아래와 같은 Linux 명령을 탐색한다:
ls –l, vi, whoami, date

2-1-4. 아래 명령을 실행하여 현재 설치된 응용 프로그램을 나열한다:
 # ls /bin

2-2. GPIO 를 테스트하려면 gpio-demo 응용 프로그램을 사용하라.
gpio-demo 응용 프로그램은 GPIO 주변 장치에 값을 쓰거나 GPIO 주변 장치에서 값을 읽는데 사용된다.

2-2-1. 아래 명령을 사용하여 시스템에서 사용 가능한 GPIO 를 확인한다.
 # ls /sys/class/gpio

```
root@ZYBO_petalinux_v2015_4:~# ls /sys/class/gpio/
export       gpiochip894  gpiochip898  gpiochip902  gpiochip906  unexport
```

Figure 3. Checking for the available GPIOs in the system

예: # cat /sys/class/gpio/gpiochip894/label

GPIO 레이블 파일에는 GPIO 레이블이 들어 있다.
레이블에는 GPIO 의 실제 주소 정보가 들어 있다.
GPIO 레이블 형식은 /amba@0/gpio@<PHYSICAL_ADDRESS> 에 해당한다.

```
root@ZYBO_petalinux_v2015_4:~# cat /sys/class/gpio/gpiochip894/label
/amba_pl/gpio@41200000
```

Figure 4. GPIO Physical Address Information

Zybo 보드에서 온보드 GPIO 는 아래와 같다:
4 개의 LED(4 채널), 4 개의 버튼(4 채널), 4 개의 스위치(4 채널)에 해당한다.

gpiochip<ID> 와 GPIO 맵핑은 아래와 같다:
4 개의 스위치 전용으로 gpiochip894;
4 개의 LED 전용으로 gpiochip898;
4 개의 버튼 전용으로 gpiochip902;

2-2-2. 아래 명령을 실행하여 4 개의 LED 를 모두 킨다.
gpio-demo –g 898 –o 15

참고: 출력은 HEX 형식으로 HEX 값의 하위 4 비트를 사용하여 쓰여진다.
예로 이 경우 d'15 에 해당하는 것은 0xF 이다.

2-2-3. 다음 명령을 실행하여 4 개의 DIP 스위치의 상태를 출력한다.
gpio-demo –g 894 –i

DIP 스위치 값을 변경해보자.

2-3. CPU 정보 및 인터럽트를 찾는다.
탐색 할 또 다른 흥미로운 곳은 /proc 디렉토리다.
이것은 커널에 윈도우를 제공하는 가상 디렉토리다.
예로 /proc/cpuinfo 파일에는 CPU 에 대한 세부 정보가 들어 있으며 /proc/interrupts 는 인터럽트 통계를 제공한다.

2-3-1. 아래 명령을 입력하라.
 # cat /proc/cpuinfo

```
root@ZYBO_petalinux_v2015_4:~# cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 0 (v7l)
BogoMIPS        : 1292.69
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpd32
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x3
CPU part        : 0xc09
CPU revision    : 0

processor       : 1
model name      : ARMv7 Processor rev 0 (v7l)
BogoMIPS        : 1299.25
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpd32
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x3
CPU part        : 0xc09
CPU revision    : 0

Hardware        : Xilinx Zynq Platform
Revision        : 0003
Serial          : 0000000000000000
```

Figure 5. Viewing the CPU information

/proc/cpuinfo 의 내용은 버전 및 하드웨어 기능과 같은 프로세서 정보를 표시한다.
프로세서의 구성에 따라 /proc/cpuinfo 가 위와 다를 수 있다.

2-3-2. 아래 명령을 입력한다:

 # cat /proc/interrupts

```
root@ZYBO_petalinux_v2015_4:~# cat /proc/interrupts
           CPU0       CPU1
 16:          0          0       GIC  27  gt
 17:          0          0       GIC  43  ttc_clockevent
 18:       1162        841       GIC  29  twd
 21:         43          0       GIC  39  f8007100.adc
142:          0          0       GIC  35  f800c000.ocmc
143:        264          0       GIC  82  xuartps
144:          0          0       GIC  51  e000d000.spi
145:          0          0       GIC  54  eth0
146:         59          0       GIC  56  mmc0
147:          0          0       GIC  45  f8003000.dmac
148:          0          0       GIC  46  f8003000.dmac
149:          0          0       GIC  47  f8003000.dmac
150:          0          0       GIC  48  f8003000.dmac
151:          0          0       GIC  49  f8003000.dmac
152:          0          0       GIC  72  f8003000.dmac
153:          0          0       GIC  73  f8003000.dmac
154:          0          0       GIC  74  f8003000.dmac
155:          0          0       GIC  75  f8003000.dmac
156:          0          0       GIC  40  f8007000.devcfg
162:          0          0       GIC  41  f8005000.watchdog
IPI1:         0          0  Timer broadcast interrupts
IPI2:       775        810  Rescheduling interrupts
IPI3:         0          0  Function call interrupts
IPI4:        13         34  Single function call interrupts
IPI5:         0          0  CPU stop interrupts
IPI6:         0          0  IRQ work interrupts
IPI7:         0          0  completion interrupts
Err:          0
```

Figure 6. Viewing the Interrupts

/proc/interrupts 는 시스템의 인터럽트 정보를 보여준다.
명령이 실행 된 시기와 하드웨어 장치에 접근하기 위해 실행 된 다른 명령이 무엇인지에 따라 결과가 다를 수 있다.
/proc/interrupts 는 시스템에 어떤 인터럽트가 있는지, 인터럽트 유형과 얼마나 많은 인터럽트가 발생했는지 알려준다.

2-3-3. 데스크탑 컴퓨터에서 다른 터미널 창을 연다.

2-3-4. cat /proc/cpuinfo 를 입력하고 동일한 명령을 사용하여 임베디드 리눅스 정보와 비교한다.

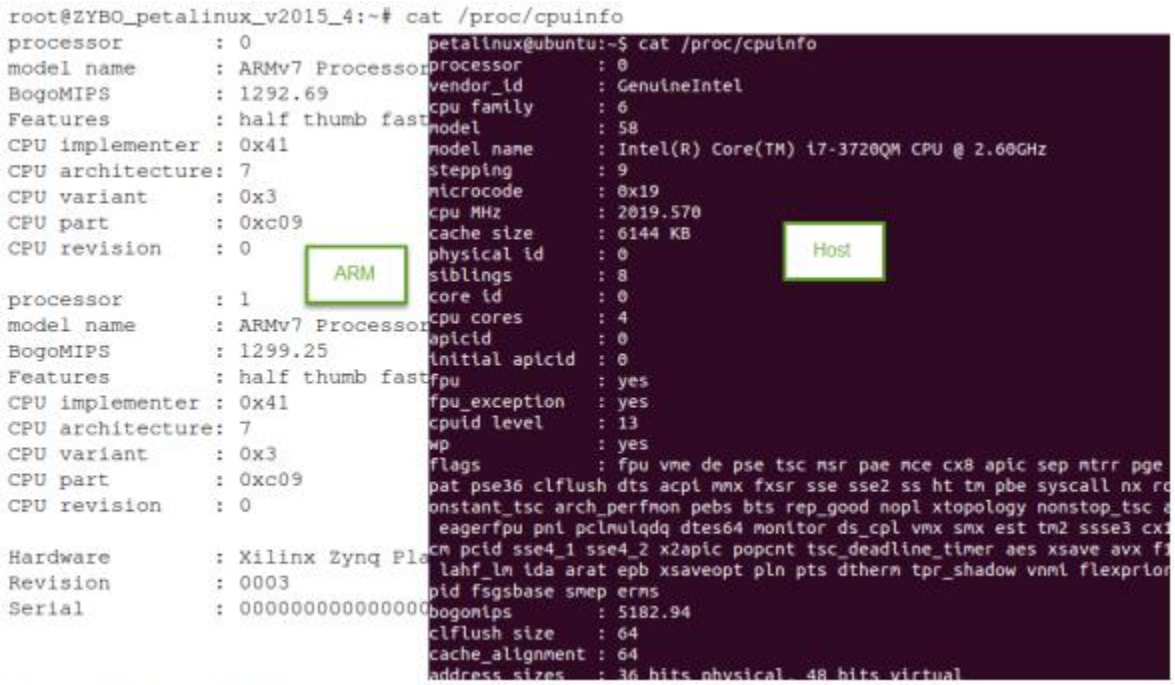


Figure 7. Serial Port and Terminal Window - cpuinfo

다음은 표준 리눅스 디렉토리 구조다:
/bin, /dev, /tmp, /var 등등에 해당한다.

2-4. ping 명령을 사용하여 네트워크 연결을 검사한다.

2-4-1. 시스템이 부팅 된 이후 로그인 이름과 암호로 root 를 입력하여 시스템에 로그인 한다.

2-4-2. ping 명령을 실행하여 Host 시스템을 ping 한다.

 # ping 192.168.1.1

Host 시스템에서 응답을 확인해야 한다.

2-4-3. Host 컴퓨터 터미널 창에서 ping 명령을 실행하여 Target Board 의 응답을 확인한다.

[host]$ ping 192.168.1.2

고정 IP 주소는 시스템 구축시 할당되었다.

Target 컴퓨터에서 응답을 볼 수 있다.

2-4-4. GtkTerm 창을 닫는다.

2-4-5. Board 전원을 끈다.

Conclusion

이 Lab 의 목적은 임베디드 리눅스 타겟을 소개하고 데스크 탑 리눅스 계보에서 그 유산을 보여주기 위한 것이다.
이것은 임베디드 리눅스의 즉각적인 이점 중 하나다.
응용 프로그램 및 사용자 환경으로서 표준 데스크탑 Linux 플랫폼과 엄청난 공통점이 있다.

간단히 말하지만 이 소개는 Board 설정 및 전원 켜기,
임베디드 리눅스 Target 에 로그인하고 탐색 하는데 필요한 기본 경험을 제공해야 한다.
이러한 기본 기능은 후속 Lab 에서 보다 확장된다.

Completed Solution

솔루션을 실행하려면 BOOT.bin 및 image.ub 를 sources/lab1/SDCard 디렉토리에서 Micro-SD 카드로 복사한다.
Micro-SD 카드를 Zybo 에 배치한다.
Zybo 를 Micro-SD 카드 부팅 모드로 설정한다.
이더넷 케이블을 사용하여 Zybo 를 Host 컴퓨터에 연결한다.

아래 명령을 실행하여 Host 에서 DHCP 서버를 시작한다.

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

보드의 전원을 켠다.
터미널 세션을 설정한다.

PS-SRST(BTN7) 버튼을 누른다.
보드를 부팅 시킨다.
시스템에 로그인하고 Lab 을 테스트한다.

References

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