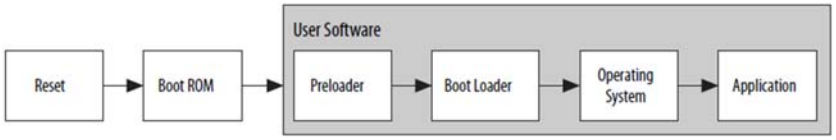
임베디드 리눅스 응용 프로그래밍

Why use an Operating System?

- Device Drivers
  - USB Device Drivers – Keyboard, Mouse, Bluetooth
  - …
- Internet Protocol Stack
  - Easily start using the Ethernet port
- Multi-threaded Applications

Typical ARM Cortex-A9 Boot Sequence



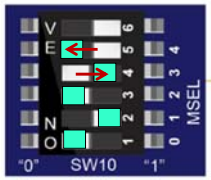
- Boot ROM
  - Hard coded by Altera(Intel)
  - Determines the boot source by reading the boot select pins
- Preloader
  - In (1) **Flash/SD Card** or (2) **FPGA**
  - Typically initializes the DDR3 SDRAM and HPS I/O pins
- Boot loader
  - Loads and starts the operating system

Linux SD card Images

- Linux SD card images
  - Preloader
  - Bootloader
  - Linux (kernel + Distribution)
  - FPGA related drivers
  - Automatically programs FPGA with prebuilt system

Using SD card Images

- step 1: Power Off the DE1-SoC
- step 2: Set MSEL to `b01010 on the DE1-SoC
  - Enables ARM to be able to configure the FPGA

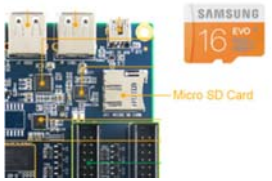


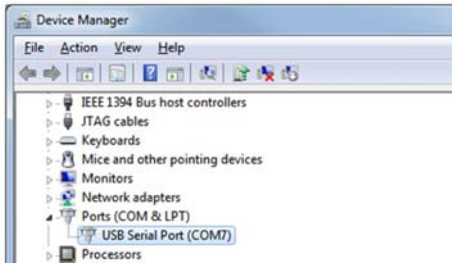
Board Reference	Signal Name
SW10.1	MSEL0
SW10.2	MSEL1
SW10.3	MSEL2
SW10.4	MSEL3
SW10.5	MSEL4
SW10.6	N/A

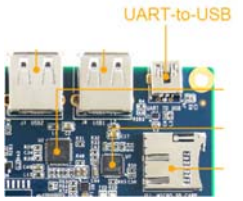
Table 3-2 MSEL Pin Settings for FPGA Configure of DE1-SoC

MSEL[4:0]	Configure Scheme	Description
10010	AS	FPGA configured from EPCQ (default)
01010	FPPx32	FPGA configured from HPS software: Linux
00000	FPPx16	FPGA configured from HPS software: U-Boot, with image stored on the SD card, like LXDE Desktop or console Linux with frame buffer edition.

- Step 3: Insert Linux SD Card
- Step 4: Power On the DE1-SoC
- Step 5: Ensure the UART-to-USB is Connected to the Host Computer
  - (black cable)
- Step 6: Check Device Manager for COM Port Settings (USB serial port)







- Step 7: Open Terminal Software (putty or teraterm)
- Step 8: Connection setup
  - "Serial" connection
  - USB serial port (COM*n*)
  - baud rate : 115200
- Step 9: Save session(setup) for later use
- Step 10: Open(OK) Connection

Sample Program

- source code "helloworld.c"

```c
#include <stdio.h>

int main(void){

    printf("Hello World!\n");

    return 0;
}
```

- compile

  # gcc helloworld.c 또는 # gcc –o helloworld helloworld.c
- execute

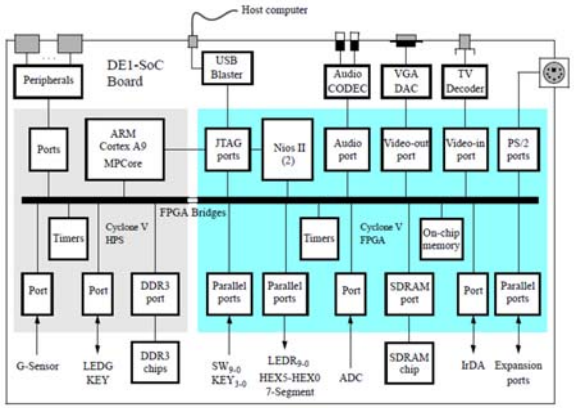
  # ./a.out 또는 # ./helloworld

Programming the FPGA

- Create the desired system using the Quartus software and the Qsys System Integration Tool (이 과목에 포함되지 않음)
- Copy the programming bitstream to Linux
- Then within Linux command line
  - Disable the HPS-FPGA bridges
  - Configure the FPGA with your bitstream
  - Re-enable the bridges

The Default *DE1-SoC Computer* System

- The Linux distribution automatically programs the FPGA with the *DE1-SoC Computer* System during boot



Virtual Addresses vs Physical Addresses

- Linux creates **a virtual address space** for programs
- FPGA peripheral are given **physical addresses** in Qsys
- Linux application program uses virtual addresses instead of physical addresses
- Linux provides functions '**mmap**' and '**munmap**' for address mapping
  - mmap - map virtual address space to physical addresses
  - munmap - un-maps virtual addresses space
- 하드웨어가 맵핑된 메모리 장치 (/dev/mem)의 일부 주소 영역을 가상주소로 맵핑하여 하드웨어 제어 가능
  - 메모리 장치에 대한 가상주소 맵핑은 **root 권한**이 필요함

Exercise: Using FPGA Peripherals within Linux

- We will use the default *DE1-SoC Computer* system
- We will use the red LEDs and the slider switches
- The program copies the value of the switches to the LEDs

MMAP

```
#define HW_REGS_BASE ( 0xff200000 )
#define HW_REGS_SPAN ( 0x00200000 )
#define HW_REGS_MASK ( HW_REGS_SPAN - 1 )

// Open /dev/mem
if( ( fd = open( "/dev/mem", ( O_RDWR | O_SYNC ) ) ) == -1 ) {
    printf( "ERROR: could not open \"/dev/mem\"...\n" );
    return( 1 );
}

// get virtual addr that maps to physical
virtual_base = mmap( NULL, HW_REGS_SPAN, ( PROT_READ | PROT_WRITE )
   , MAP_SHARED, fd, HW_REGS_BASE );

if( virtual_base == MAP_FAILED ) {
    printf( "ERROR: mmap() failed...\n" );
    close( fd );
    return(1);
}
```


Using the Virtual Address

```
#define LED_PIO_BASE 0x0
#define SW_PIO_BASE 0x40

volatile unsigned int *h2p_lw_led_addr=NULL;
volatile unsigned int *h2p_lw_sw_addr=NULL;

// Get the address that maps to the LEDs
h2p_lw_led_addr=(unsigned int *)(virtual_base +
                        (( LED_PIO_BASE ) & ( HW_REGS_MASK ) ));
h2p_lw_sw_addr=(unsigned int *)(virtual_base +
                        (( SW_PIO_BASE ) & ( HW_REGS_MASK ) ));

while(!stop){
    *h2p_lw_led_addr = *h2p_lw_sw_addr;
}
```


MUNMAP

```
if( munmap( virtual_base, HW_REGS_SPAN ) != 0 ) {
    printf( "ERROR: munmap() failed...\n" );
    close( fd );
    return( 1 );
}

close( fd );
```


Source Code

- source code "leds.c"

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <signal.h>

#define HW_REGS_BASE ( 0xff200000 )
#define HW_REGS_SPAN ( 0x00200000 )
#define HW_REGS_MASK ( HW_REGS_SPAN - 1 )
#define LED_PIO_BASE 0x0
#define SW_PIO_BASE 0x40

volatile sig_atomic_t stop;

void catchSIGINT(int signum){
    stop = 1;
}
```

signal handler

```c
int main(void)
{
    volatile unsigned int *h2p_lw_led_addr=NULL;
    volatile unsigned int *h2p_lw_sw_addr=NULL;
    void *virtual_base;
    int fd;

    // catch SIGINT from ctrl+c, instead of having it abruptly close this program
    signal(SIGINT, catchSIGINT);

    // Open /dev/mem
    if( ( fd = open( "/dev/mem", ( O_RDWR | O_SYNC ) ) ) == -1 ) {
        printf( "ERROR: could not open \"/dev/mem\"...\n" );
        return( 1 );
    }

    // get virtual addr that maps to physical
    virtual_base = mmap( NULL, HW_REGS_SPAN, ( PROT_READ | PROT_WRITE ), MAP_SHARED, fd,
    HW_REGS_BASE );
    if( virtual_base == MAP_FAILED ) {
        printf( "ERROR: mmap() failed...\n" );
        close( fd );
        return(1);
    }
```



```c
    // Get the address that maps to the LEDs
    h2p_lw_led_addr=(unsigned int *)(virtual_base + (( LED_PIO_BASE ) & ( HW_REGS_MASK ) ));
    h2p_lw_sw_addr=(unsigned int *)(virtual_base + (( SW_PIO_BASE ) & ( HW_REGS_MASK ) ));

    printf("Running leds. To exit, press Ctrl+C.\n");

    while(!stop){
        *h2p_lw_led_addr = *h2p_lw_sw_addr;
    }

    if( munmap( virtual_base, HW_REGS_SPAN ) != 0 ) {
        printf( "ERROR: munmap() failed...\n" );
        close( fd );
        return( 1 );

    }
    close( fd );
    return 0;
}
```