Lab 1 - Using TS-7250 I/O

Week 2 – Kernel Modules

ECE 4220/7220
Real Time Embedded Computing
Spring 2016
University of Missouri
What is a Kernel Module?

- Code that can be loaded and unloaded into the kernel.
- Extend the functionality of the kernel.
  - We do not have to rebuild an entire kernel to add new functions.
  - We do not have to build a large kernel that contains all functionality.
- Does not require the system to reboot after load.
Example

- In lab we use RTAI's Real time implementation.
- Our kernel has already been patched and built.
- However, we do not have certain functionality built into our kernel that we have to load ourselves:
  - rtai_lxrt.o – Use RTAI services in user space
  - rtai_fifos.o – RTAI fifo functionality
  - rtai_sem.o – RTAI semaphore functionality
Kernel Module Functions

• As root user you can install, uninstall, and list kernel modules.

• `lsmod` – list all the modules currently installed

• `insmod mod_name.o` – installs the specified module into the kernel

• `rmmod mod_name` – removes the specified mod from the kernel (notice there is no '.o')
Writing a Module

• Modules differ from applications.
• Instead of having an 'int main(void)' section, modules have two sections:
  • int init_module(void)
    − Runs when the module is installed.
    − Must return 0 if there are no errors.
  • void cleanup_module(void)
    − Runs when the module is uninstalled.
Example Module Code

```c
#ifndef MODULE
#define MODULE
#endif

#ifndef __KERNEL__
#define __KERNEL__
#endif

#include <linux/module.h>
#include <linux/kernel.h>

MODULE_LICENSE("GPL");

int init_module(void)
{
    // your code here
    return 0;
}

void cleanup_module(void)
{
    // your code here
}
```
Building a Module

• Modules are .c files that are compiled into object files (.o)

• This means we just want to call the compiler and **not** call the linker.

• To do this, replace arm-linux-gcc with echo for the linker.

• It is recommended to change the optimization level to **O0** (Properties->C/C++ Build->Settings->GCC C Compiler->Optimization)

• Since we are using the cross-compiler we need to provide a path to the include files for the TS-7250.

• Look for the paths in the document:

  Lab_How_Tos.pdf
Using I/O in a Module

void * __ioremap(unsigned long phys_addr, unsigned long size, Flags)

- phys_addr – beginning of physical address range
- Size – size of physical address range
- Flags

• By using this function we are returned a virtual address which we can use just as in our applications.

• Example:

  ptr = (unsigned long *) __ioremap(0x80840000, 4096, 0);
printk() Command

- We can use printk() command as an alternative for printf() in a kernel module.

- Despite what you might think, printk() is not meant to communicate information to the user, even though we used it for exactly this purpose in our program! It happens to be a logging mechanism for the kernel, and is used to log information or give warnings.

- It doesn’t print on the screen. You can check the printed line using “dmesg” command in the terminal.