ModulePlanning:EmbeddedLinuxWorkshop,Hagenberg,SS 2013

Objective: Familiar with difference of embedded/PC Linux system, Development tool chain for embedded ARM systems, Kernel modules, Principles of device drivers, Communication of device drivers with user processes, Interrupt handling techniques, Kernel threads and kernel synchronization techniques

Prerequisite: Basic knowledge of Linux, Some practical experience with a Linux system (Debian, Ubuntu, etc); Good knowledge of software (cross) development in C (gcc, make) and testing

Grading Scheme: "erfolgreich besucht": presence in at least 4 sessions

Note 1 oder 2: **individual** results in Lab on interrupt latency (session 6) based on a short report

Session	Titel	Contents	Lab
1: Di 4. Juni	Introduction, Cross Development	Difference: Desktop Linux / Embedded Linux GNU tool chain (gcc, gdb) Editors, Makefile, Tools, Bootloader Services: TFTP, DHCP, NFS used for setup	Experimental setup of the embedded system development infrastructure Build your cross development tools: tool chain, root file system, kernel Cross-develop a simple application program, write a makefile for it
2: Do 6. Juni	Kernel, Kernel Loadable Modules	Kernel Architecture, Process-Management System Calls, File System Review: pseudo FS (/proc, /sys), Kernel Modules	Simple module (Hello World) Dependent modules with parameters Module communication with user space via /proc
3: Di 11. Juni	Kernel Concurrency Management Kernel Threads	Atomic variables and bit operations, Semaphores, Mutexes, Wait queues, Completions, Spin locks Introduction to kernel threads	Kernel thread APIs, creation/cancellation of kthreads Performance analysis of kernel synchronization methods
4: Do 13. Juni	Device Drivers	Device Interfaces, User API (system calls) Sample Drivers for LEDs, Switches, GPIOs	Driver to control LED's and switches Driver to control GPIOs
5: Di 18. Juni	Interrupt Processing	Concepts of interrupt processing Interrupt handler, Deferred interrupt handling by tasklets and kernel threads	Preliminary Exercises: GPIO Interrupts – Simple interrupt handlers – Sharing interrupts and interrupt handlers – Deferred interrupt processing using tasklets and kernel threads
6: Do 20. Juni	Mini-project	Applied drivers	Interrupt latency estimation using GPIO triggered interrupts. Maintain a kernel event buffer. Evaluate its events by a user space program to estimate different types of latencies like interrupt fast/slow handling latency, kernel/user latency.