Scheduling Overview

CPU Scheduling Approach Overview

Operation		Programmed I/O	ISR	Non-Premptive Scheduler		Preemptive Scheduler	
Synch – Detect event		while (!e);	HW: Peripheral	Task requests release	IRQ -> ISR, which requests release	Task signals event	IRQ -> ISR
Start Scheduler		n/a	HW: Interrupt System	Code reaches scheduling point, RTCS_Scheduler resumes		SVC Exception handler	
Pick what to run		n/a (implied, sequential)	HW: Interrupt Vector	Pick highest-priority ready task		Pick highest-priority ready task	
Switch to other code	Save old context	n/a	HW: push some regs onto stack	Compiler handles with register allocation		Some saved by entering SVC exception handler. SW saves rest of registers in TCB	
	Create or restore context	n/a	none	Prolog of task function saves more as needed		Restore some registers from TCB with software. Return from interrupt restores rest	
	Transfer Control	n/a (implied, sequential)	PC <- Vector	Subroutine call		Return from interrupt	
New code is in …		This function	IRQHandler	Subroutine		Task function	
Preemption	By Interrupt	Yes, IRQ	Yes, Higher priority IRQ	Yes, IRQ		Yes, IRQ	
	By Task	Not possible	Not possible	Not possible		Yes, higher-priority task	
Yield		No, unless statically scheduled and manually integrated (can use FSM)	Not possible	No, unless statically scheduled and manually integrated (can use FSM)		Yes, explicit OS call (yield, delay) or implicit (potentially blocking call)	
Terminate		…?	return from interrupt	Return from subroutine		OS call, end of task root function	

CPU Scheduling Approach Overview

		Programmed I/O	ISR	Non-Premptive Scheduler	Preemptive Scheduler
Concurrency	Among ISRs		Yes, if nested	Yes, if nested	Yes, if nested
	Between ISRs and Tasks		Yes	Yes	Yes
	Among tasks	n/a	n/a	No, unless explicitly integrated (synchronous, or async using FSMs)	Yes