Education-oriented implementation of OS concepts on STM32

Philipp Keese

Forschungsseminar WiSe 2023/24

Motivation

- Mini-OS für die Lehre entwickeln
  - Übersichtlicher Code
  - Leicht zu verstehen
- Fokus:
  - Kontextwechsel
  - Scheduling
  - Synchronisation
- Treiber für Peripherie

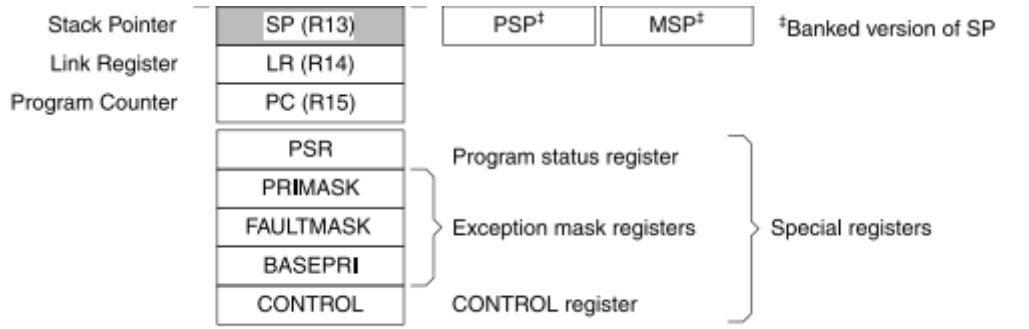
- Auf STM32 Microcontroller
  - ARMv7 CPU
  - 512 KiB Flash
  - 96 KiB RAM
  - Kein Bootloader
  - Simple CPU-Modi
  - Eingebaute Peripherie
- In Rust implementiert
  - Mehr Abstraktion möglich
  - Fokus auf Konzepte

ARMv7 Architektur & STM32

- Main & Process Stack
- Handler & Thread Modi
- MPU + eingeschränkter Zugriff
- SVCall für Systemanfragen
- PendSV für Kontexwechsel außerhalb von Interrupts
- Konfigurierbare Prioritäten für Interrupts
- SysTick Timer für Slices

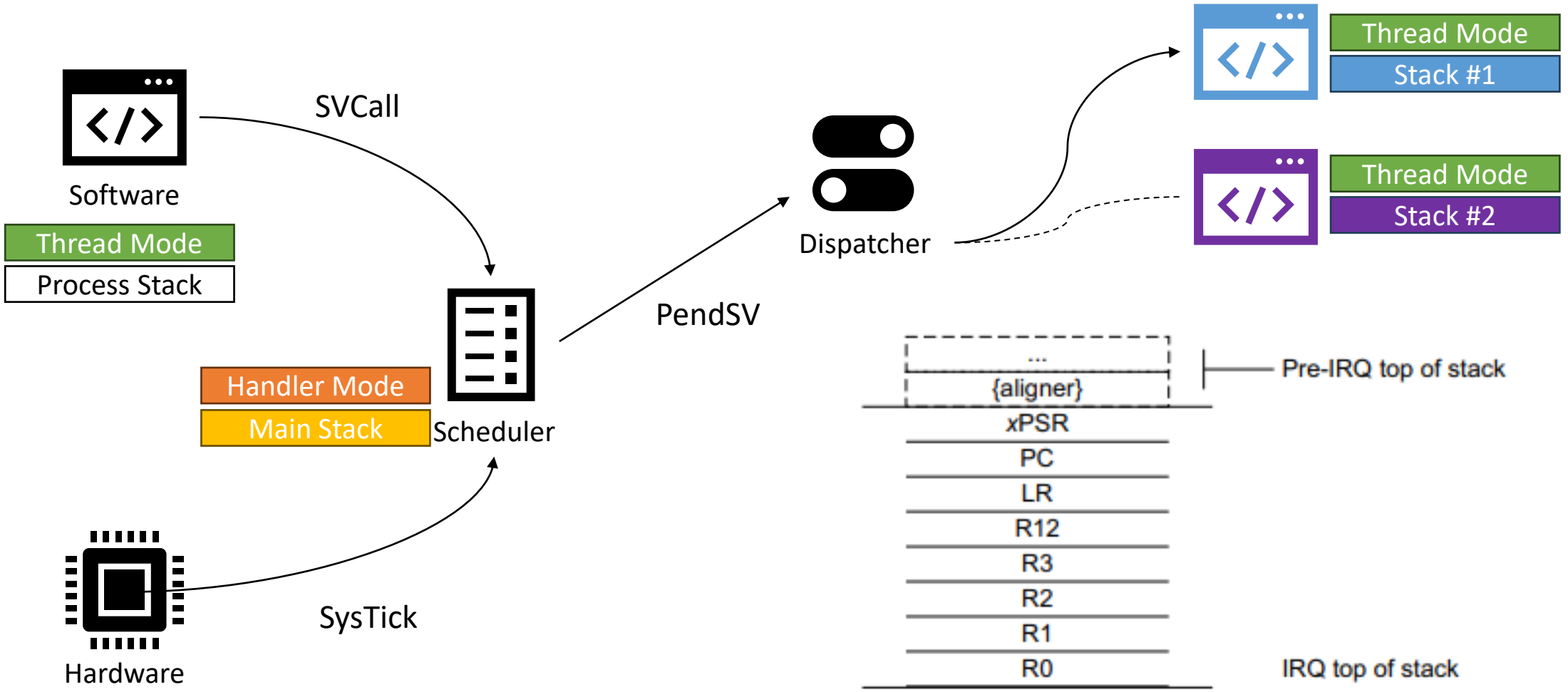
-	3	settable	SVCall	System Service call via SWI instruction
-	4	settable	Debug Monitor	Debug Monitor
-	-	-	-	Reserved
-	5	settable	PendSV	Pendable request for system service
-	6	settable	Systick	System tick timer
0	7	settable	WWDG	Window Watchdog interrupt

Vector Table for STM32F446



Processor Core Registers

Ablauf eines Kontextwechsels



Geplante Features

Must Have:

- Threads zur Laufzeit starten
- Mutex + Semaphores
- Schutz der Kernels mit MPU

Should Have:

- Task-Prioritäten
- Abtrahierter UART, SPI, etc.
- Freiwilliges Blockieren

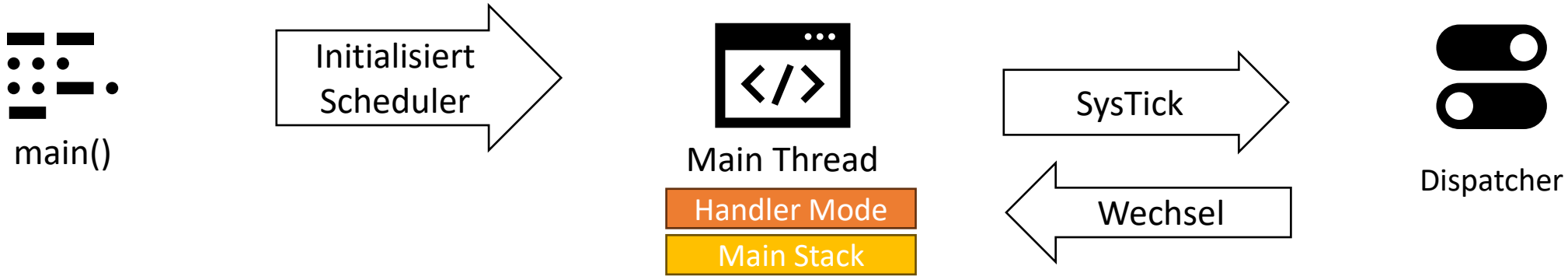
Could Have:

- Realtime Scheduler
- Message-Queues
- Priority Inheritance
- Prozessisolation

Won‘t Have:

- Virtual Memory
- Externe Speicheranbindung

Aktueller Zustand



Quellen

STM32F446 Reference Manual (RM0390)

https://www.st.com/en/microcontrollers-microprocessors/stm32f446re.html#

STM32 Cortex M4 Programming Manual (PM0214)

https://www.st.com/resource/en/programming_manual/pm0214-stm32-cortexm4-mcus-and-mpus-programming-manual-stmicroelectronics.pdf

Cortex-M4 Revision r0p0 Technical Reference Manual

https://developer.arm.com/documentation/ddi0439/b/