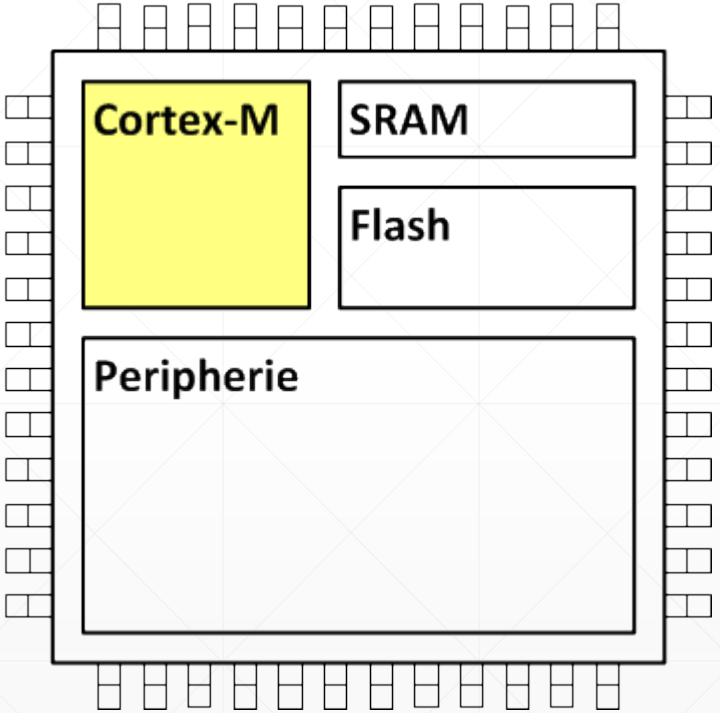
ARM Cortex-M Prozessoren

Referat von Peter Voser
Embedded Development GmbH

Winterthur, 1. Juli 2014
swissT.meeting
Embedded Computing Conference

SoC (System-on-Chip)



Instruction Sets
ARM, Thumb, Thumb-2

- 32-bit ARM
  - verbesserte Rechenleistung
  - höherer Speicherbedarf als 8/16-bit MCUs (Preis/Stromverbrauch)
- 1995: ARM7TDMI verfügte neu über 16-bit Instruktionen
  Thumb Instruction Set (Umschalten mittels T-Bit: 0 = ARM, 1 = Thumb)
  Codegrösse ~ 70%, Rechenleistung 80%
- 2003: Thumb-2 Technologie erlaubt Ausführung von 16/32-bit Instruktionen ohne Umschalten
  Codegrösse ~ 74% bei nahezu gleicher Rechenleistung
  Energieeffizient weil nur ein Instruktionsdecoder
- Cortex-M3 war erste MCU mit dem Thumb-2 Instruction Set

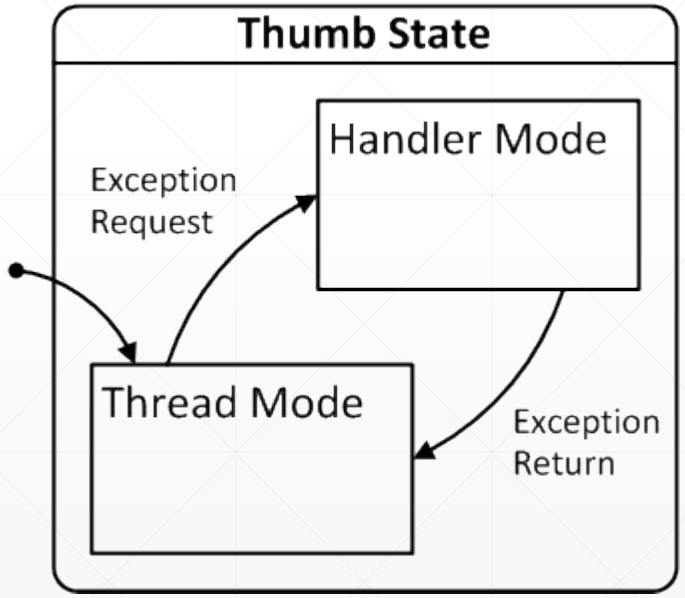
Instruction Sets
Cortex-M0

- ARMv6-M Architektur von Cortex-M0

  - unterstützt 56 Thumb (16-bit) Instruktionen
  - mit minimalem Subset von sechs 32-bit Instruktionen von Thumb-2

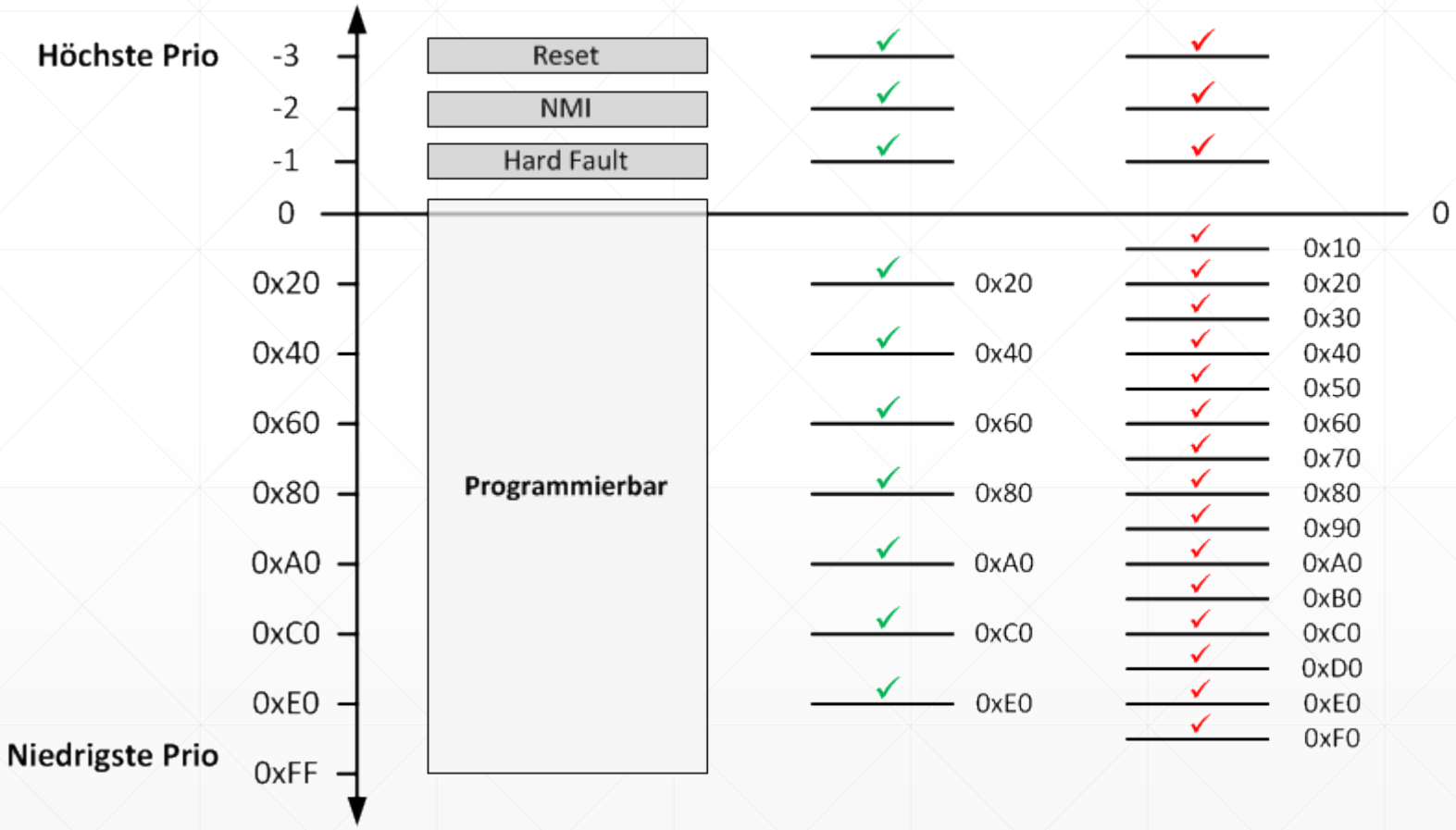
Operation Modes

- Thread Mode: normale Programmausführung, beginnt mit dem Resetvektor
- Handler Mode: eine Exception unterbricht Thread Mode, z.B. Interrupt



CM0 ✓ CM3 ✓ CM4 ✓

NVIC
Exception/Interrupt Priority Levels



Beispiel 1

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Implementiert			Nicht implementiert				

Beispiel 2

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Implementiert				Nicht implementiert			

CM0 ✓ CM3 ✓ CM4 ✓

NVIC
Group/Sub-Priority

- **Group-Priority**: definiert, ob Interrupt ausgeführt werden kann, wenn Prozessor bereits einen anderen Interrupt ausführt - auch *Pre-Emption Priority*
- **Sub-Priority**: definiert Reihenfolge der Ausführung gleichzeitig eintretender Interrupts
- Einstellbar in Register *Priority Group* (SCB)

Beispiel 1

Anzahl impl. Prio Bits = 3
Priority Group = 5

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Preempt		Sub	Nicht implementiert				

Beispiel 2

Anzahl impl. Prio Bits = 8
Priority Group = 0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Preempt							Sub

CM0	CM3	CM4
✗	✓	✓

CMSIS
Cortex Microcontroller Software Interface Standard

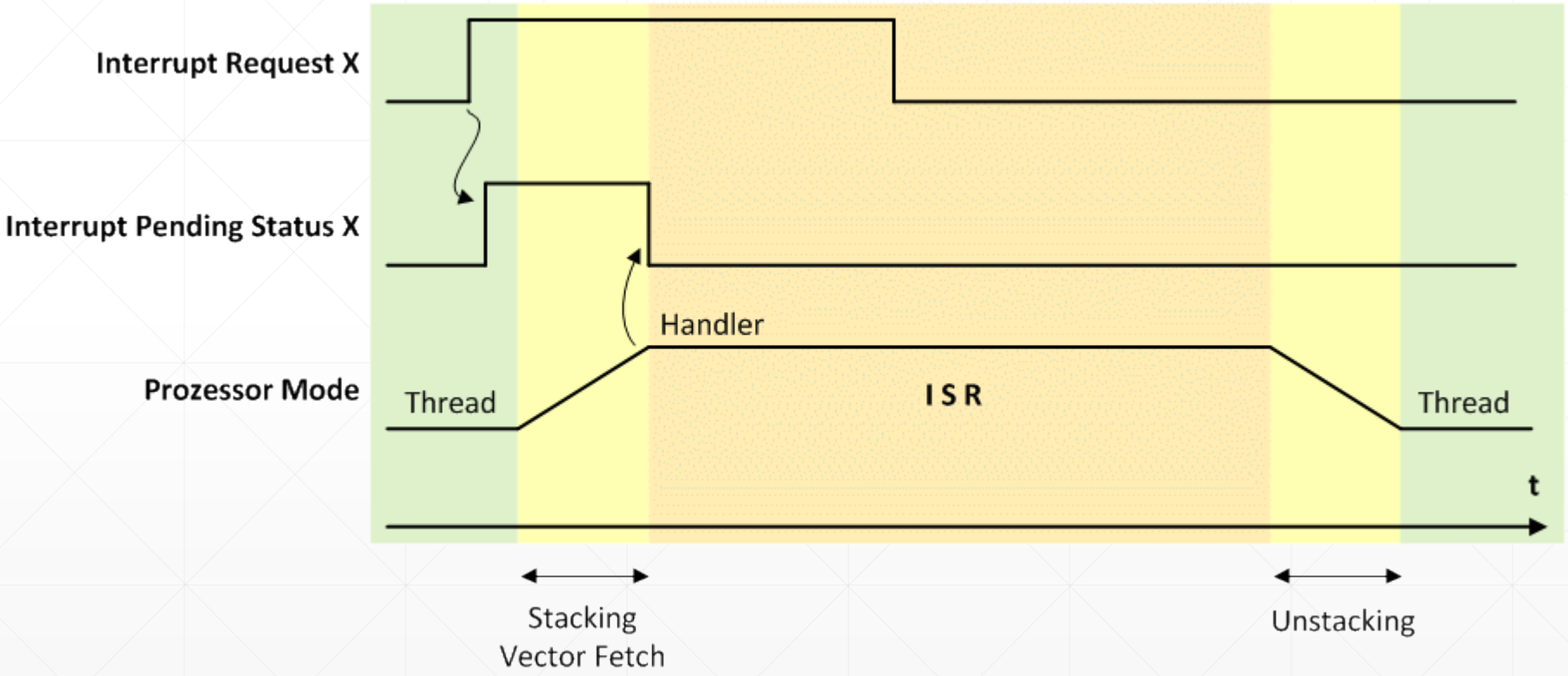
- Toolchain-unabhängige API für Software Entwickler
- Beispiele für Interrupts aus Include Files core_cm0.h oder core_cm3.h:

```
- void NVIC_EnableIRQ(IRQn_Type irq)
- void NVIC_DisableIRQ(IRQn_Type irq)
- void NVIC_SetPriority(IRQn_Type irq, uint32_t priority)
- void NVIC_SetPriorityGrouping(uint32_t group)
```

- Anzahl implementierter Priority Bits abhängig vom eingesetzten Derivat:

```
#define __NVIC_PRIO_BITS  5
```

NVIC
Interrupts auf der Zeitachse



CM0 ✓ CM3 ✓ CM4 ✓

NVIC
Die Vorteile

- **Kurze Interrupt Latenzzeit**

  - 12 Taktzyklen (ohne Memory Wait States, Stacking und Vector Fetch parallel)

- **Voraussagbares Interrupt Timing**

- **Interrupt Handler in C**

  - ARM C Compiler darf in Funktion *nur* R0..R3, R12, R14 und PSR modifizieren
  - Genau diese Register werden beim *Stacking* versorgt
  - R4..R14 muss bei Verwendung selbst auf dem Stack versorgt werden

Low Power Features

- Batteriebetrieb
- Einfacheres Power Supply Design, weniger elektromagnetische Störungen
- Active current [µA/MHz]
- Sleep current [µA]
- Energieeffizienz abhängig von Implementierung der Speicher und Peripherie

Low Power Features
Sleep Modes

- **Sleep Mode** und **Deep Sleep Mode**
- Einstellbar über **System Control Register (SCR)**
- Eintritt in zuvor eingestellten Modus mittels folgender Instruktionen:

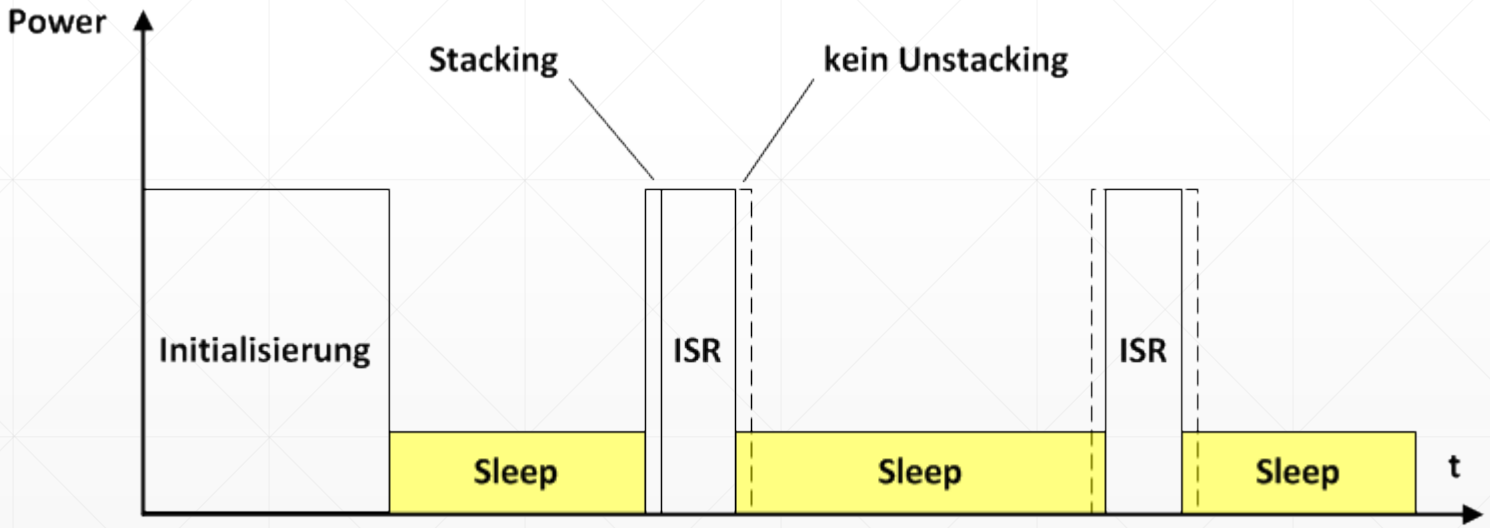
  - WFI → Wait for Interrupt → CMSIS Funktion `void __WFI(void);`

  - WFE → Wait for Event → CMSIS Funktion `void __WFE(void);`

- WFI und WFE aufwecken mittels Interrupts, WFE zusätzlich mittels RXEV und Events, welche vor dem WFE Aufruf erfolgten

Low Power Features
Sleep-on-Exit

- Für Interrupt-gesteuerte Firmware
- Alle Operationen werden mittels Interrupts ausgeführt



CM0	CM3	CM4
✓	✓	✓

OS Support Features
System Tick Timer

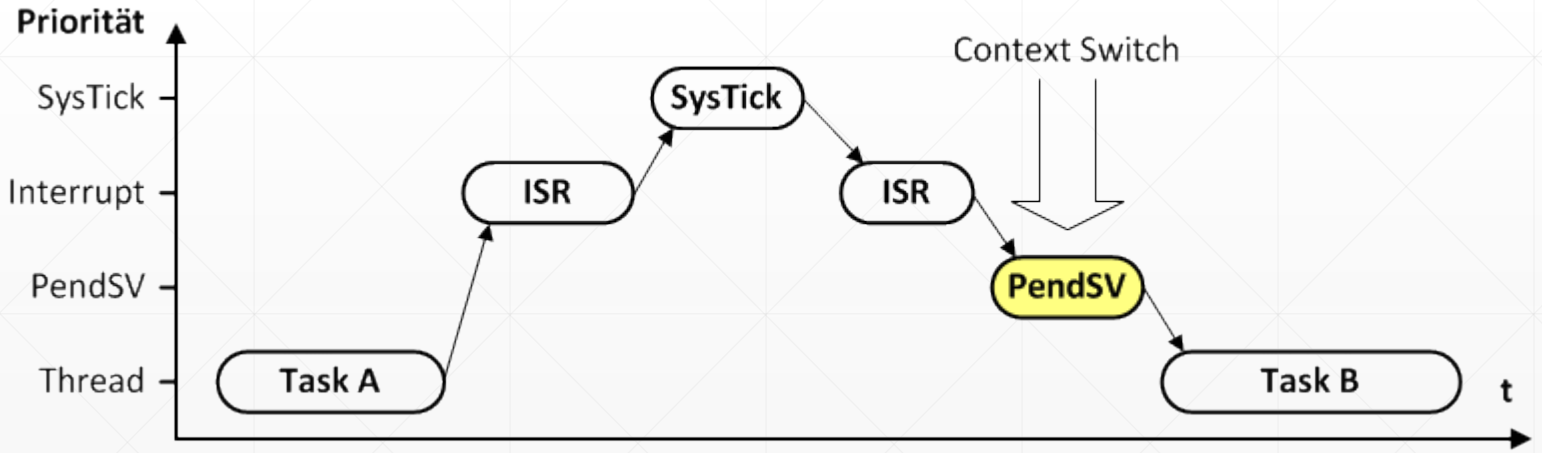
- Einfacher 24-bit Timer
- NVIC Bestandteil (Exception Nummer 15)
- OS benötigt Timer für Task Management und Context Switching
- Software ist besser portierbar, wenn SysTick Teil von Prozessor statt Peripherie
- CMSIS Funktion → `uint32_t SysTick_Config(uint32_t ticks);`

OS Support Features
SVC Exception

- **Supervisor Call**, ausgelöst durch SVC Instruktion
- Einstellbare Priorität, Ausführung präzise:
  SVC Handler muss nach der SVC Instruktion ausgeführt werden
- Anwendung als OS API: User Tasks dürfen nur über Supervisor Aufrufe auf System Ressourcen zugreifen (MPU nötig)
- Beispiel: `SVC 0x7` → ruft über den Handler die zugeordnete Funktion 7 auf
  Zusätzliche Parameter können auf dem Stack abgelegt werden

OS Support Features
PendSV Exception

- **Pended Service Call**, ausgelöst durch Bit in *Interrupt Control and State Register*
- Einstellbare Priorität, Ausführung jedoch im Gegensatz zu SVC unpräzise: PendSV Handler wird erst nach Abarbeitung aller Interrupts ausgeführt
- Grundlage für effizientes OS Context Switching

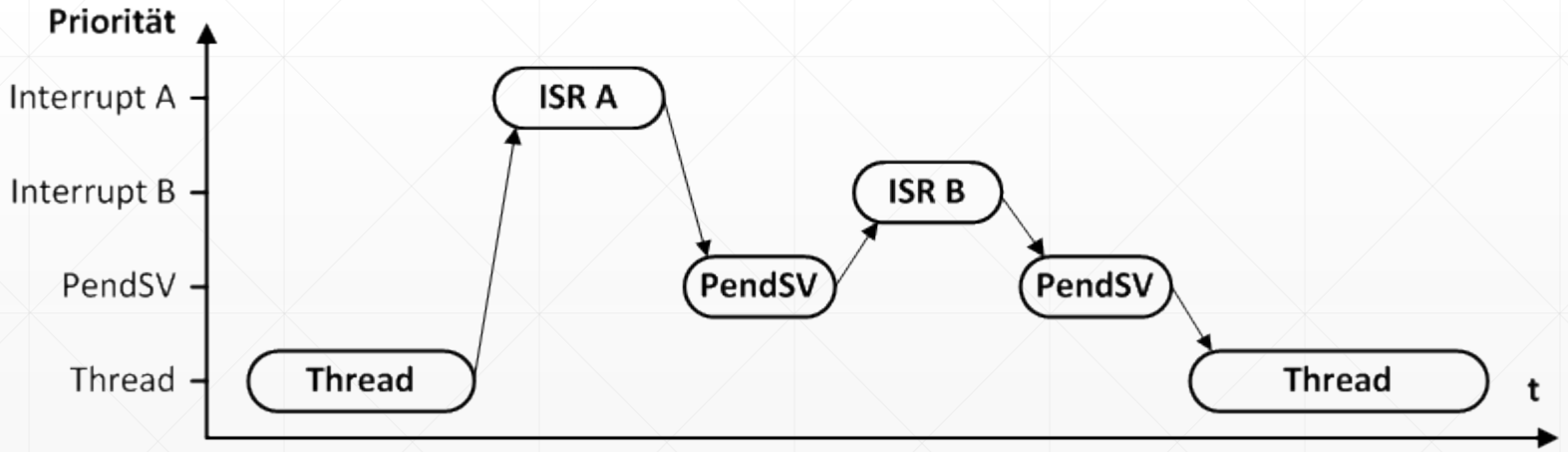


CM0	CM3	CM4
✓	✓	✓

PendSV Anwendung ohne OS

- Auch ohne OS leistet PendSV gute Dienste
- Aufteilung einer rechenintensiven ISR in zwei Teile (Top-half/Bottom-half)

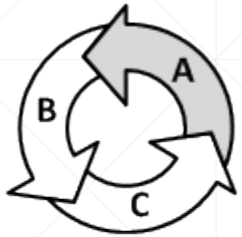
  - Zeit-kritischer Teil in ISR
  - Restliche Bearbeitung in PendSV Handler (durch Interrupts unterbrechbar)



CM0	CM3	CM4
✓	✓	✓

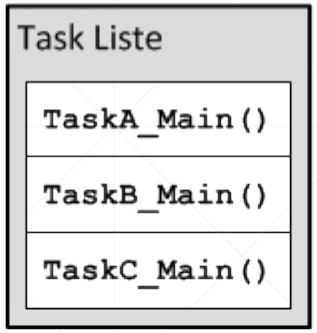
Keep it simple
Cooperative Multi-Tasking mit CM Features

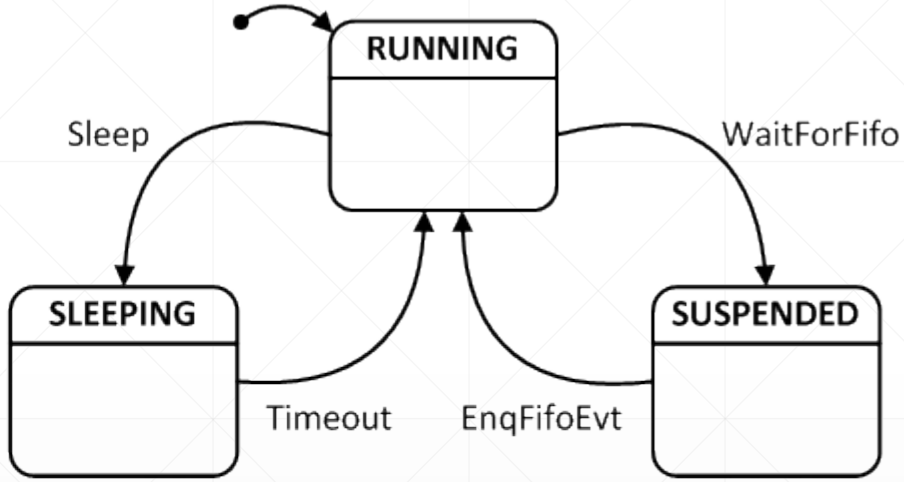
Task Loop im Thread Mode




```
void TaskA_Main(void)
{
  // Run-to-Completion.
  ...
}

if(nofRunningTasks == 0)
{
  // Sleep & wait for events.
  __WFE();
}
```





SysTick

Dekrementiert Sleep Timeouts aller Tasks im Zustand SLEEPING

Interrupts

Können wartende Tasks über FIFO mit einer Event ID aufwecken

Anwendung für Device Drivers

API

```
void TASKUTIL_Sleep(uint32_t msec);
void TASKUTIL_EnqueueFifoEvt(uint32_t evt);
TuRc TASKUTIL_WaitForFifo(FifoId id, uint32_t *evt);
```

Preemptive vs. cooperative Multi-Tasking

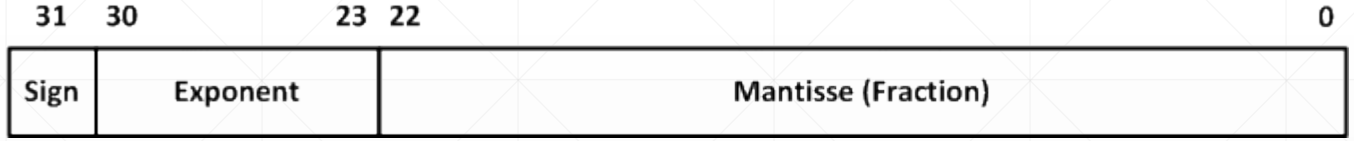
Preemptive, Prioritäts-bezogener Scheduler:
Immer dem höchst-prioren Task im Zustand *ready to run* gehört die CPU

Preemptive	**Cooperative**
Einhalten einer Deadline garantiert	Einhalten einer Deadline nicht garantiert
Hoher RAM Verbrauch jeder Task hat eigenen Stack	Geringer RAM Verbrauch alle Tasks benutzen gleichen Stack
Schutz gemeinsam genutzter Ressourcen nötig	Kein Schutz gemeinsam genutzter Ressourcen nötig dank Run-to-Completion
Blockierer automatisch unterbrochen	Blockierer müssen unterbrochen werden

Cortex-M4
Floating Point Operationen

- Floating Point Unit (FPU) basierend auf IEEE 754-2008 Standard
- M4 FPU unterstützt nur *single-precision* Gleitkommazahlen

```
float pi = 3.141592f;
```



CM0	CM3	CM4
✗	✗	✓

Cortex-M4
Floating Point Operationen

- Verfügt über 32 zusätzliche 32-bit Register

  - S0 .. S15 → caller saved registers
  - S16 .. S31→ callee saved registers

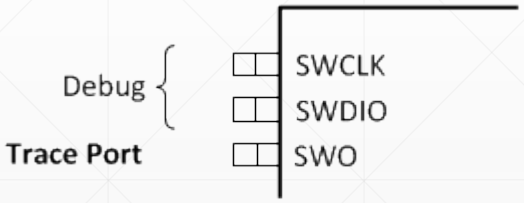
- Einfluss auf Interrupts

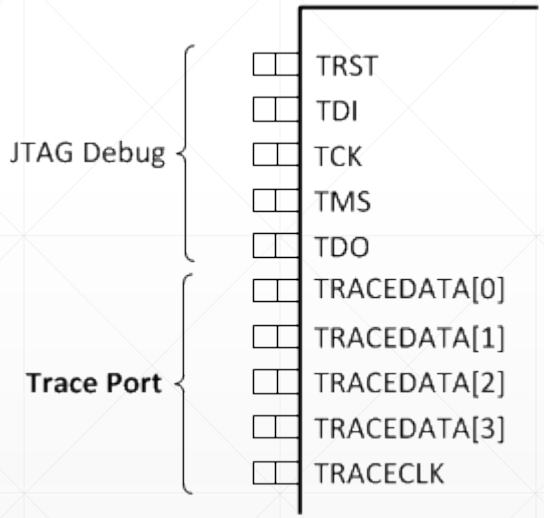
  - Längere Stack Frames (grösserer RAM Verbrauch)
  - Statt 8 Register (R0..3, R12, LR, Return Address, xPSR)
  - Neu 25 Register (S0..15, FPSCR)
  - Keine grössere Latenzzeit dank *lazy stacking*

Debugging mit Trace Schnittstellen

- Trace: über separate Pins Information der Programmausführung ausgeben
- Zwei Trace Port Varianten:

  - Serial Wire Viewer (SWV) mit 1 Pin
  - Trace Port mit 4 Pins + Clock





Debugging mit Trace Analysen

- Instrumentierte Applikationen (ITM)

  - Ausgaben über printf()
  - Periodische Status Ausgaben, z.B. Profiling Counter
  - auch bei Cortex-M0+ möglich

- Data-Watch-Trace (DWT)

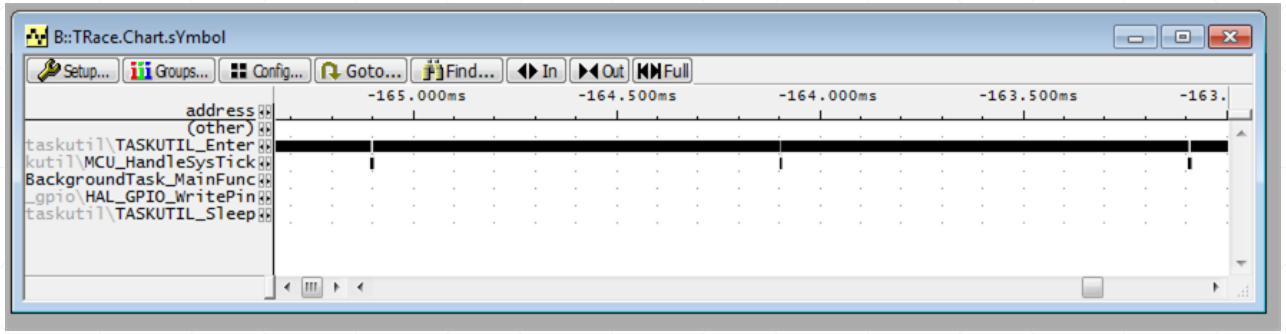
  - Datenausgabe über ITM, z.B. Variablen

- Code-Flow

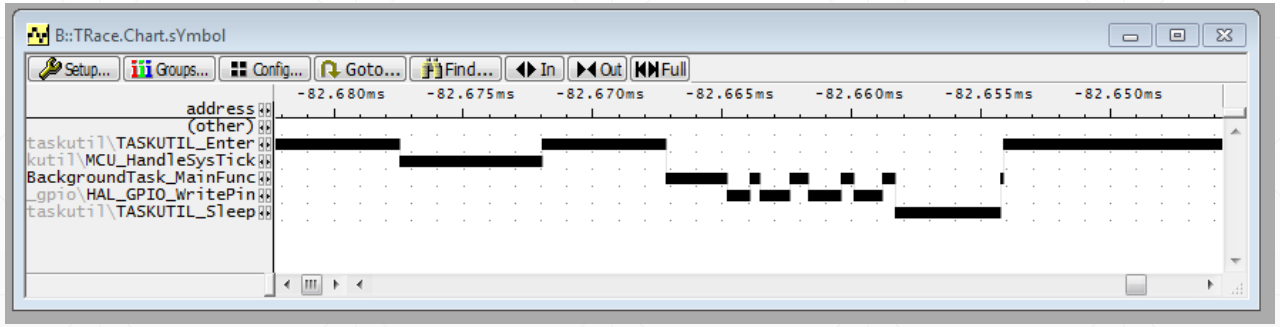
  - Trace Port Ausgabe bei Code-Fluss Änderung durch Sprünge

Debugging mit Trace
Code-Flow

1 kHz System Tick
Alle Tasks schlafen

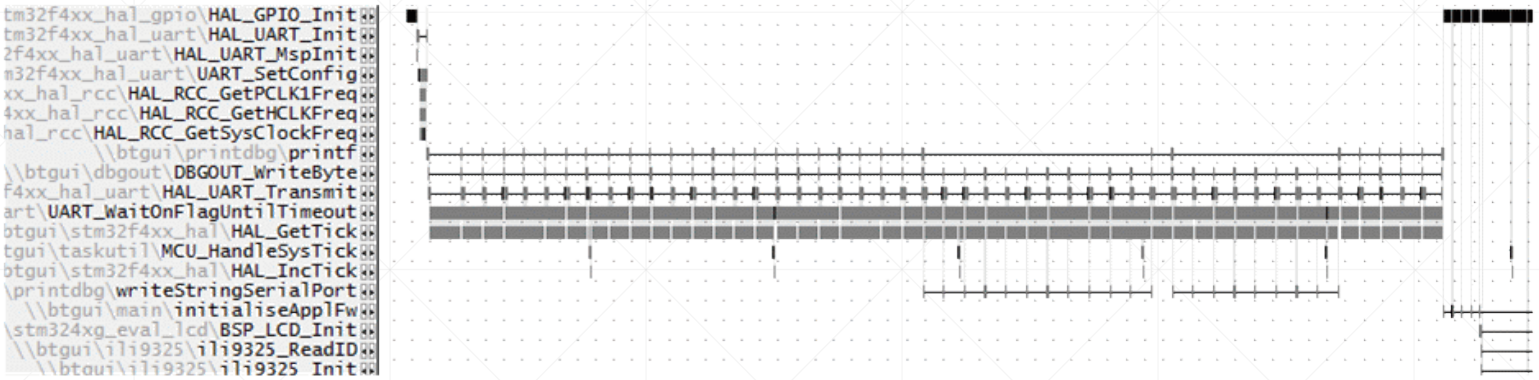


Background Task



Debugging mit Trace Code-Flow

Darstellung Chart

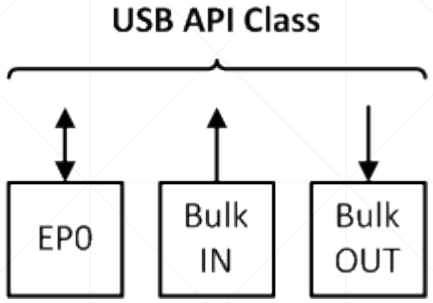


Darstellung Statistik

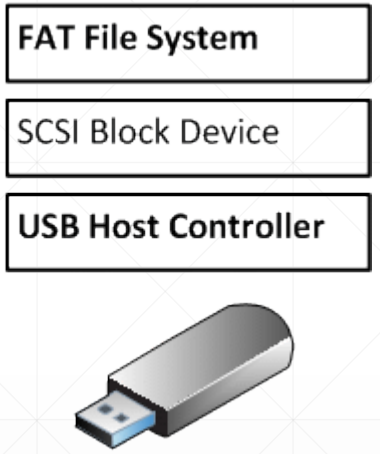
range	total	min	max	avr	count	intern%	1% 2% 5% 10% 20% 50%
(root)	13.644ms	-	13.644ms	13.644ms	-	<0.001%	
\\btgui\head\MCU_StartUp	13.644ms	-	13.644ms	13.644ms	1.(0/1)	54.832%	
\\btgui\main\main	6.127ms	-	6.127ms	6.127ms	1.(0/1)	0.017%	
\btgui\stm32f4xx_hal\HAL_Init	13.200us	13.200us	13.200us	13.200us	1.	0.029%	
hal_cortex\HAL_SYSTICK_Config	5.093us	5.093us	5.093us	5.093us	1.	0.008%	
4xx_hal_cortex\SysTick_Config	3.970us	3.970us	3.970us	3.970us	1.	0.017%	
x_hal_cortex\NVIC_SetPriority	1.617us	1.617us	1.617us	1.617us	1.	0.011%	
\HAL_NVIC_SetPriorityGrouping	4.048us	4.048us	4.048us	4.048us	1.	0.008%	
rtex\NVIC_SetPriorityGrouping	2.832us	2.832us	2.832us	2.832us	1.	0.020%	
gui\stm32f4xx_hal\HAL_MspInit	0.080us	0.080us	0.080us	0.080us	1.	<0.001%	
btgui\main\initialiseCpuClock	0.067us	0.067us	0.067us	0.067us	1.	<0.001%	
tgui\dbgout\DBGOUT_Initialise	115.857us	115.857us	115.857us	115.857us	1.	0.051%	
32f4xx_hal_gpio\HAL_GPIO_Init	531.357us	48.707us	235.901us	75.908us	7.(0/1)	3.876%	
32f4xx_hal_uart\HAL_UART_Init	55.110us	55.110us	55.110us	55.110us	1.	0.064%	
4xx_hal_uart\HAL_UART_MspInit	0.681us	0.681us	0.681us	0.681us	1.	0.004%	
2f4xx_hal_uart\UART_SetConfig	45.571us	45.571us	45.571us	45.571us	1.	0.130%	
_hal_rcc\HAL_RCC_GetPCLK1Freq	27.691us	8.797us	9.735us	9.230us	3.	0.077%	
x_hal_rcc\HAL_RCC_GetHCLKFreq	17.111us	4.999us	6.121us	5.704us	3.	0.074%	
l_rcc\HAL_RCC_GetSysClockFreq	6.908us	1.961us	2.535us	2.303us	3.	0.050%	
\\btgui\printdbg\printf	5.494ms	5.494ms	5.494ms	5.494ms	1.	0.623%	
btgui\dbgout\DBGOUT_WriteByte	5.360ms	99.809us	181.738us	111.666us	48.	0.459%	
xx_hal_uart\HAL_UART_Transmit	5.297ms	97.301us	180.110us	110.354us	48.	4.678%	
t\UART_WaitOnFlagUntilTimeout	4.656ms	2.997us	163.120us	48.497us	96.	29.225%	
gui\stm32f4xx_hal\HAL_GetTick	649.596us	0.071us	1.024us	0.387us	1679.	4.738%	
ui\taskutil\MCU_HandleSysTick	63.951us	10.405us	10.900us	10.658us	6.	0.445%	
gui\stm32f4xx_hal\HAL_IncTick	2.877us	0.370us	0.837us	0.480us	6.	0.020%	
rintdbg\writeStringSerialPort	2.124ms	897.472us	1.226ms	1.062ms	2.	0.357%	
\\btgui\main\initialiseApplFw	501.524us	-	501.524us	501.524us	1.(0/1)	0.071%	
tm324xg_eval_lcd\BSP_LCD_Init	295.932us	-	295.932us	295.932us	1.(0/1)	0.013%	

Embedded Development Demos

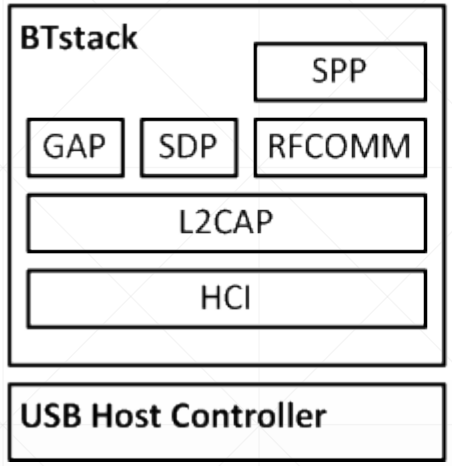
1



2



3



blue kitchen

- Portierbarer BT Stack
- Für HCI Chipsets
- Bluetooth SIG qualifiziert

Fragen?

Embedded Development GmbH

www.embedded-development.ch