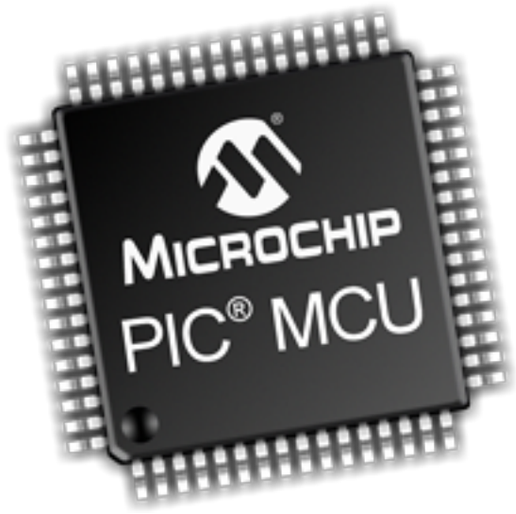
Embedded C++

C++ User Group Hannover

Inhalt

- Fokus auf ARM 32 bit µController
- Single Core, Bare Metal
- nicht EC++
- Ressourcen-Bedarf versch. Sprach-Konstrukte
- was kommt vor `int main()` ?

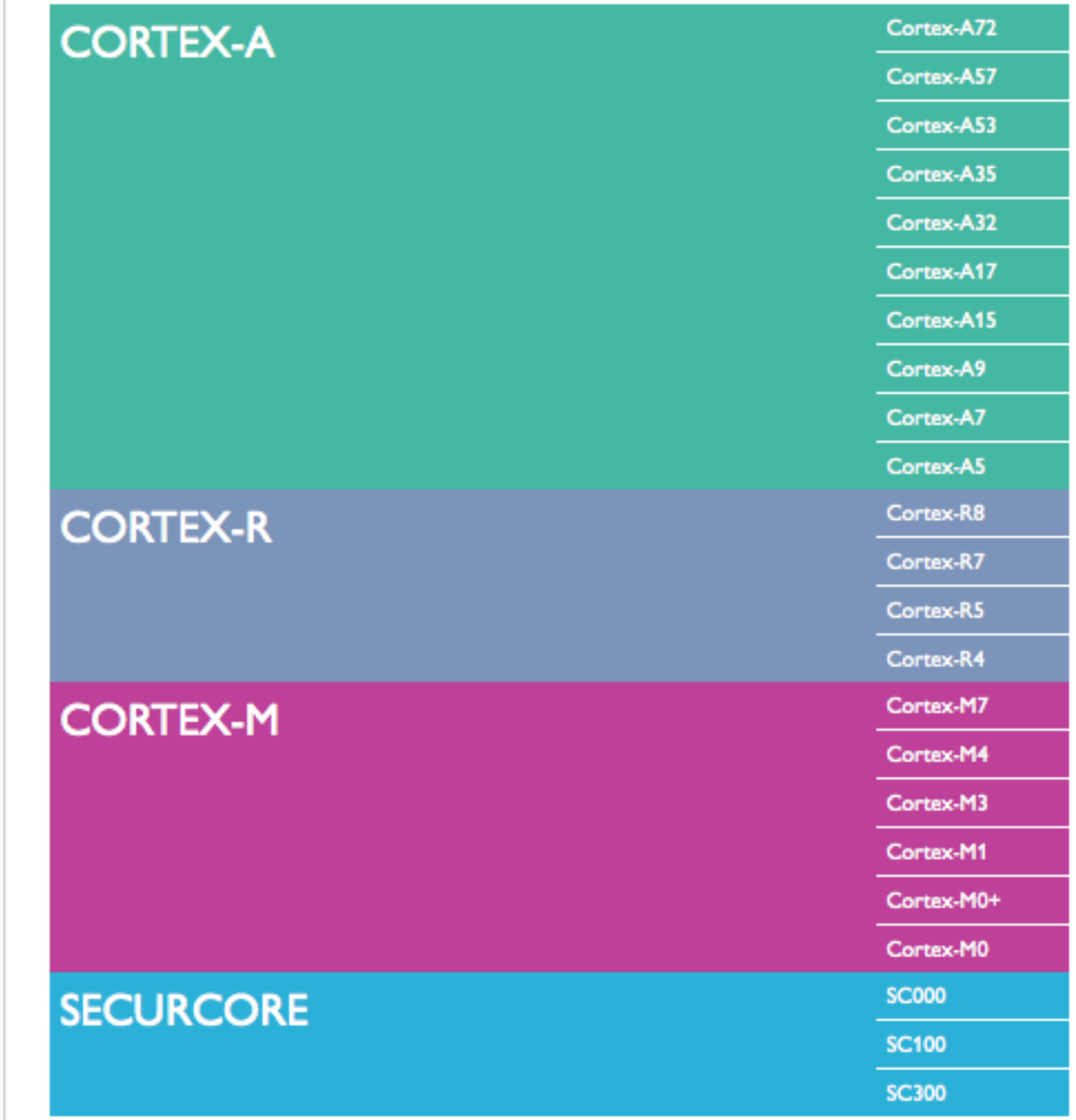
Microcontroller



- kleine CPU (8, 16, 32-Bit)
- plus jede Menge Peripherals (IO, Timer, CAN, UART, PWM, AD/DA, Radio, RNG, Crypto, I2C, SPI, Clock.)
- plus ein wenig RAM (<< 1M)
- plus ein wenig Flash
- nur wenig externe Bauteile nötig für ein funktionierendes System

ARM

- früher Advanced RISK Machines
- nur IP
- RISC cores
- Von-Neumann-Architektur
- Verschiedenste Hersteller (Ti, ST, Infineon, Nordic, NXP, Atmel, PIC, Cypress, Toshiba, etc.)
- Compiler: Keil, GCC, IAR, etc.



Bare Metal

- kein OS
- kein Scheduler
- keine threads
- direkte Kommunikation mit Hardware über Register und ISRs
- ggf. HAL des Herstellers

Bild: CC jon jordan

C++ Features mit großem Ressourcen Bedarf

- Exceptions
- Runtime Type Informations
- bestimmte Teile der Standard Libraries

C++ Abstraktionen ohne Ressourcen Bedarf

- Namespaces
- Typen
- „gute“ Namen
- `constexpr`
- `const`

Heap

- Viel Speicher für etwas Flexibilität
- Gefahr der Fragmentierung
- nicht triviale Implementierung (code size)
- Out of Memory, und nun?

OO?

- Klassen haben keinen Overhead gegenüber struct (PODs). (Padding, Alignment)
- Virtuelle Funktionen haben einen Overhead
- `private, public, protected` gibt's für umsonst

Nützlich

- Compiler können sehr gut optimieren: inline, RVO, empty base class
- Curiously recurring template pattern
- Meta Template Programming

Was passiert vor `main()`?

- Interrupt Vector Table -> Reset Vector
- Hardware Initialisierung (PLL, RAM, Busse, etc.)
- Initialisierung statischer Variablen
- Initialisierung Runtime Library
- Initialisierung globaler Objekte (C++)
- `main()`

Embedded SW Besonderheiten

- SW wird meist von E-Technikern entwickelt.
- Sehr Windows-lastig.
- immer noch Vorbehalte gegenüber C++
- eher kleinere Projekt-Größen
- Dokumentation ist meist deutlich unter dem Niveau von OpenSource Projekten
- Error-Handling: Software-Fehler, Hardware-Fehler oder „Normalfall“.
- SW Updates meist nicht trivial.