Cortex-M4 Processor Overview

with ARM Processors and Architectures

Introduction

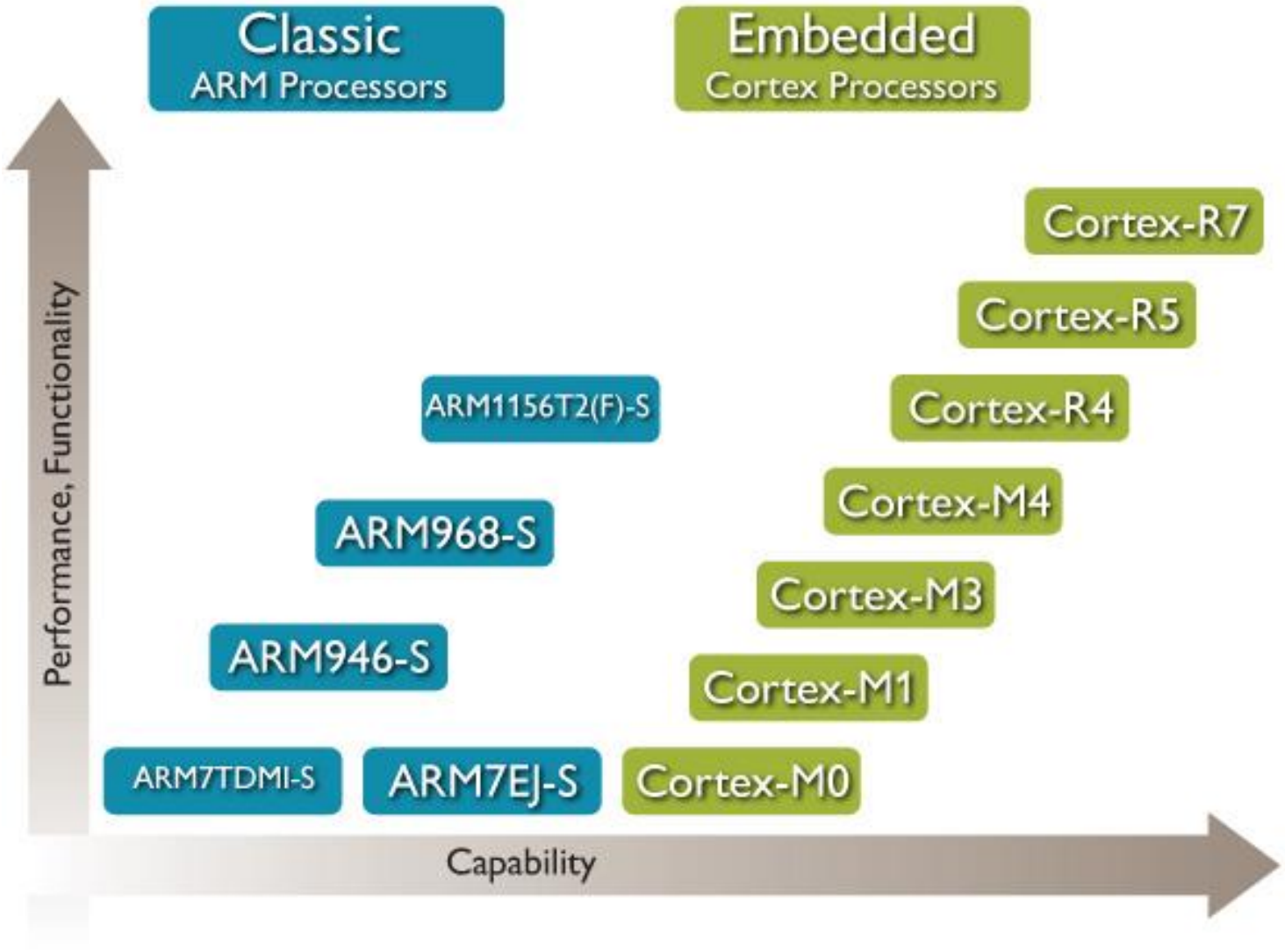
ARM

- **ARM was developed at Acorn Computer Limited of Cambridge, UK (between 1983 & 1985)**
  - RISC concept introduced in 1980 at Stanford and Berkeley
- **ARM founded in November 1990**
  - Advanced RISC Machines
- **Best known for its range of RISC processor cores designs**
  - Other products – fabric IP, software tools, models, cell libraries - to help partners develop and ship ARM-based SoCs
- **ARM does not manufacture silicon**
  - Licensed to partners to develop and fabricate new micro-controllers
    - Soft-core

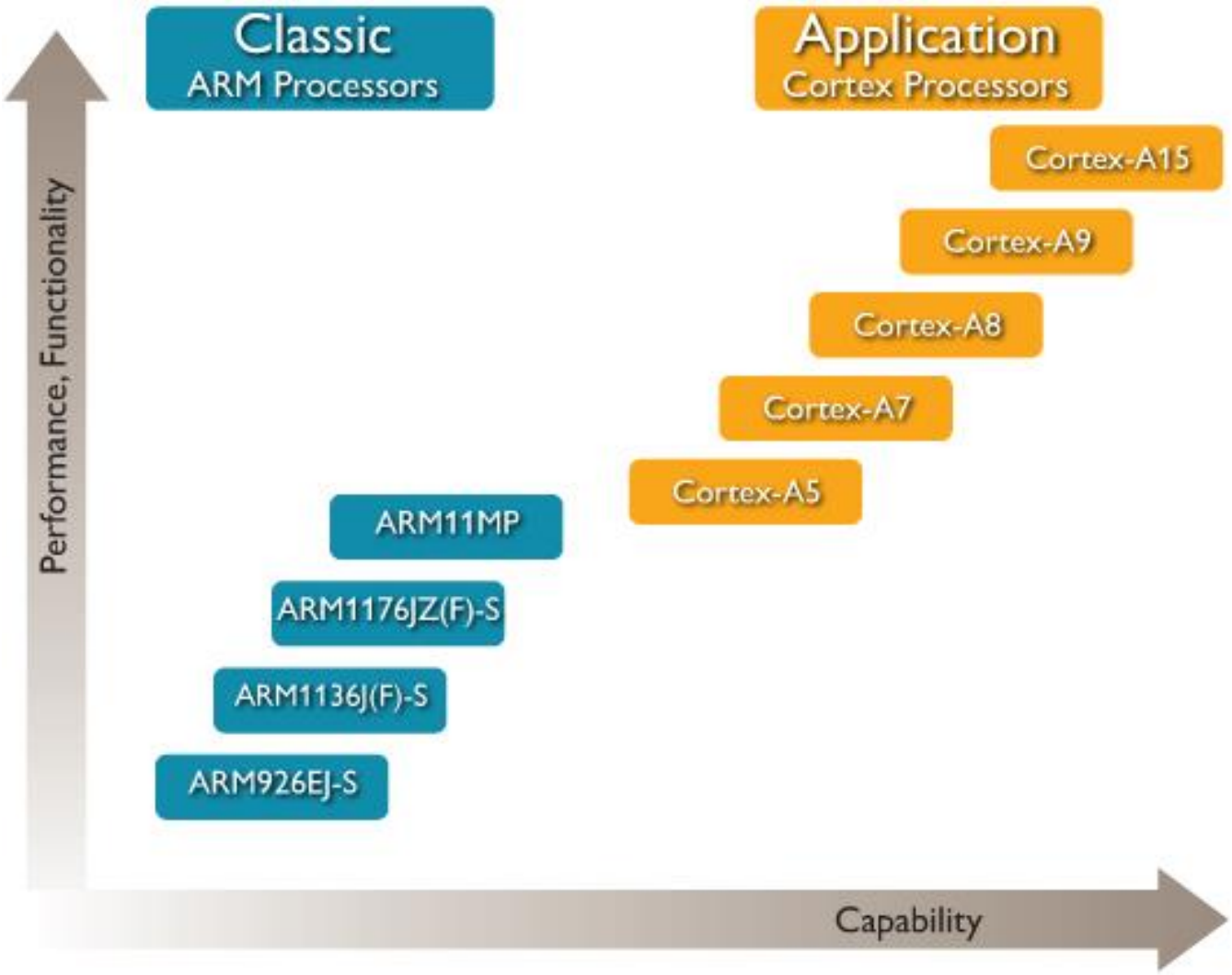
ARM Architecture

- **Based upon RISC Architecture with enhancements to meet requirements of embedded applications**
  - A large uniform register file
  - Load-store architecture
  - Fixed length instructions
  - 32-bit processor (v1-v7), 64-bit processor (v8)
  - Good speed/power
  - High code density

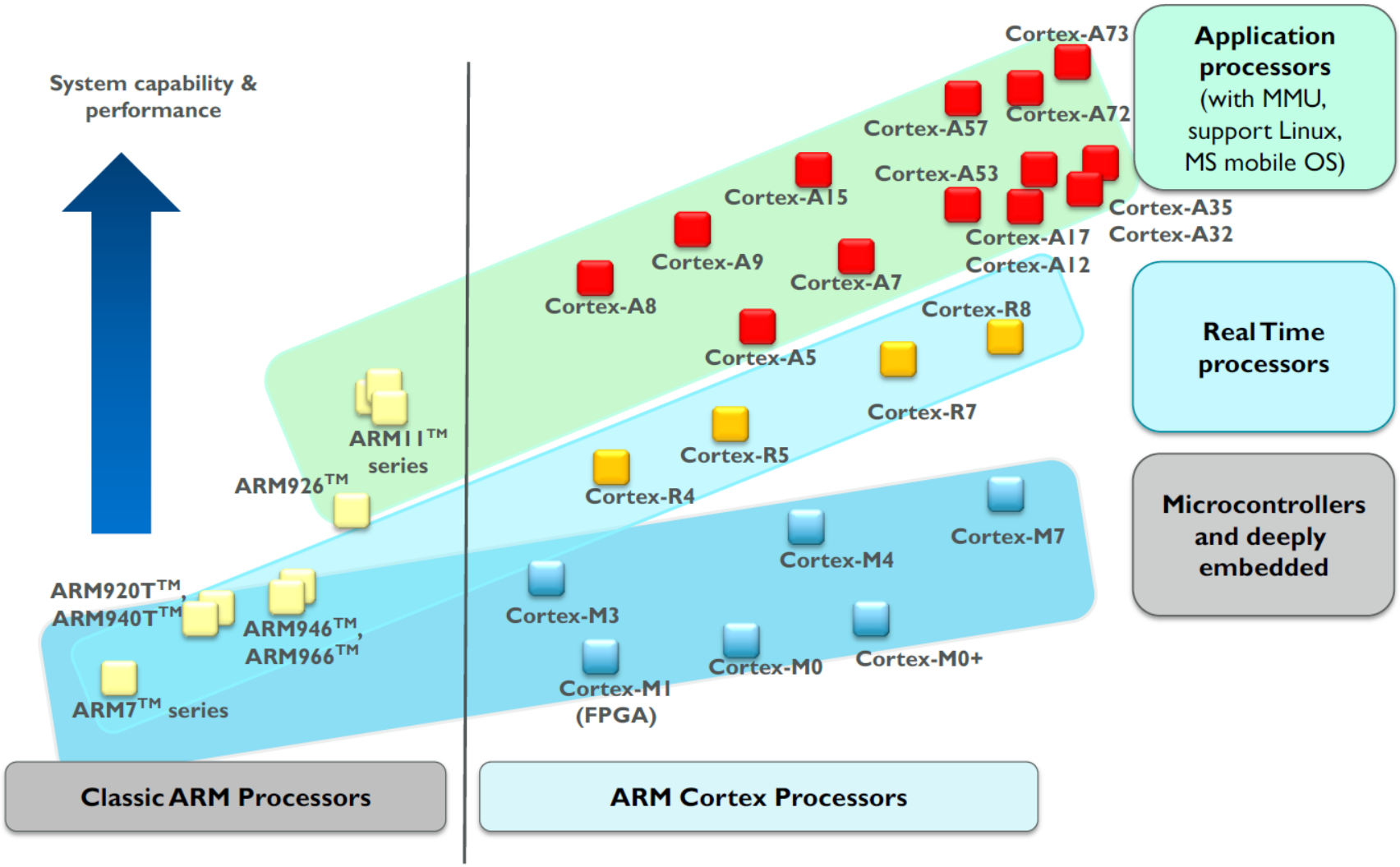
Embedded Processors



Application Processors



ARM Processor Family



Summary of Processor Characteristics

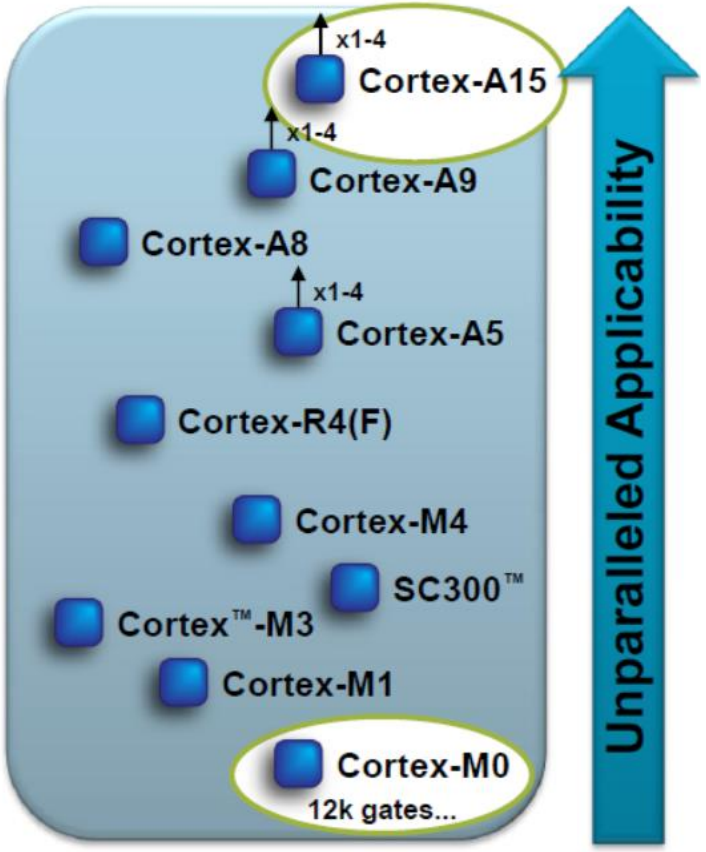
	Application processors	Real-time processors	Microcontroller processors
Design	High clock frequency, Long pipeline, High performance, Multimedia support (NEON instruction set extension)	High clock frequency, Long to medium pipeline length, Deterministic (low interrupt latency)	Short pipeline, ultra low power, Deterministic (low interrupt latency)
System features	Memory Management Unit (MMU), cache memory, ARM TrustZone® security extension	Memory Protection Unit (MPU), cache memory, Tightly Coupled Memory (TCM)	Memory Protection Unit (MPU), Nested Vectored Interrupt Controller (NVIC), Wakeup Interrupt Controller (WIC)
Targeted markets	Mobile computing, smart phones, energy-efficient servers, high-end microprocessors	Industrial microcontrollers, automotives, Hard disk controllers, Baseband modem	Microcontrollers, Deeply embedded systems (e.g. sensors, MEMS, mixed signal IC), Internet of Things (IoT)

ARM Cortex Advanced Processors

Architectural innovation, compatibility across diverse application spectrum

Cortex™
Low-Power Leadership from ARM®

- ARM Cortex-**A** family:
  - Applications processors for feature-rich OS and 3rd party applications
- ARM Cortex-**R** family:
  - Embedded processors for real-time signal processing, control applications
- ARM Cortex-**M** family:
  - Microcontroller-oriented processors for MCU, ASSP, and SoC applications



Application Examples

Cortex-A


servers


set-top boxes


netbooks


mobile applications

Cortex-R


disk drives

digital cameras


mobile baseband

Cortex-M

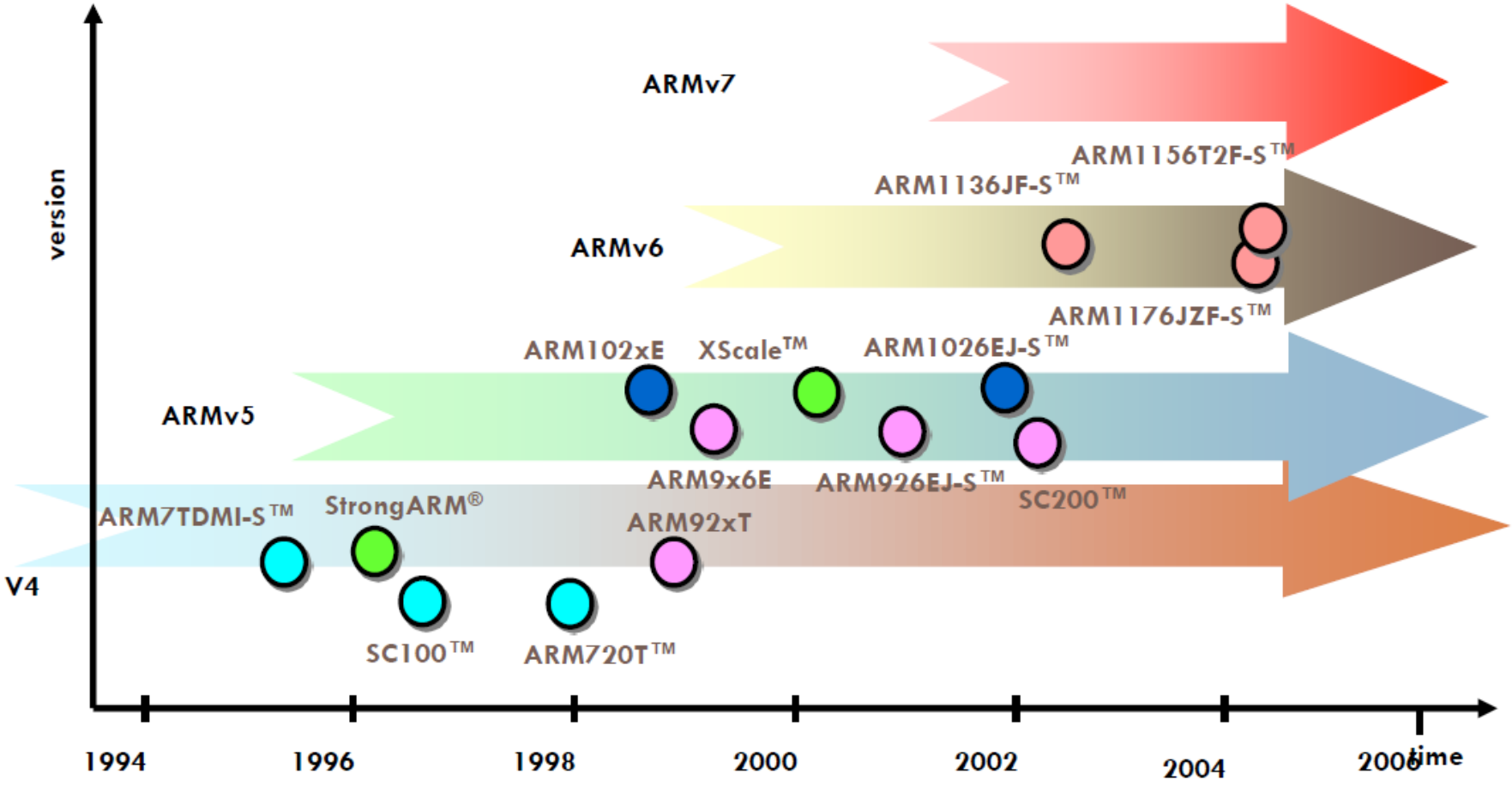

appliances


motors

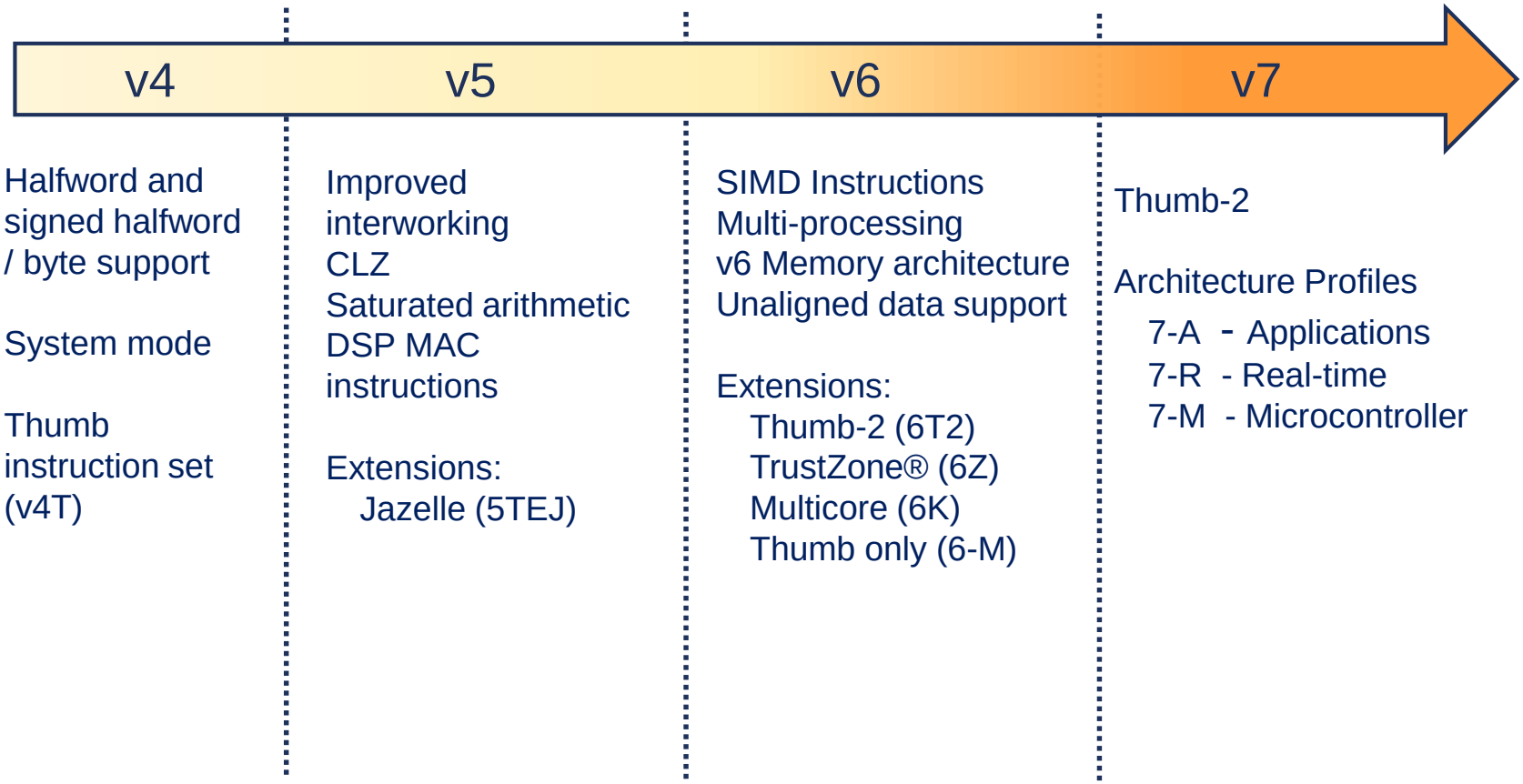

audio

ARM Architecture Overview

Architecture History



Development of the ARM Architecture



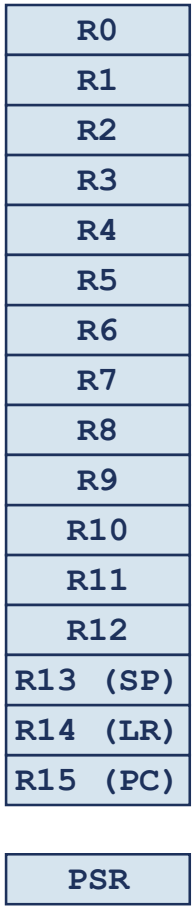
- **Note that implementations of the same architecture can be different**
  - Cortex-A8 - architecture v7-A, with a 13-stage pipeline
  - Cortex-A9 - architecture v7-A, with an 8-stage pipeline

Cotex-M Processor Family

	Descriptions
Cortex-M0	A very small processor (starting from 12K gates) for low cost, ultra low power microcontrollers and deeply embedded applications
Cortex-M0+	The most energy-efficient processor for small embedded system. Similar size and programmer's model to the Cortex-M0 processor, but with additional features like single cycle I/O interface and vector table relocations
Cortex-M1	A small processor design optimized for FPGA designs and provides Tightly Coupled Memory (TCM) implementation using memory blocks on the FPGAs. Same instruction set as the Cortex-M0
Cortex-M3	A small but powerful embedded processor for low-power microcontrollers that has a rich instruction set to enable it to handle complex tasks quicker. It has a hardware divider and Multiply-Accumulate (MAC) instructions. In addition, it also has comprehensive debug and trace features to enable software developers to develop their applications quicker
Cortex-M4	It provides all the features on the Cortex-M3, with additional instructions target at Digital Signal Processing (DSP) tasks, such as Single Instruction Multiple Data (SIMD) and faster single cycle MAC operations. In addition, it also have an optional single precision floating point unit that support IEEE 754 floating point standard
Cortex-M7	High-performance processor for high-end microcontrollers and processing intensive applications. It has all the ISA features available in Cortex-M4, with additional support for double-precision floating point, as well as additional memory features like cache and Tightly Coupled Memory (TCM)

Programmer’s Model

Processor Register Set



- **Registers R0-R12**
  - General-purpose registers
- **R13 is the stack pointer (SP) - 2 banked versions**
- **R14 is the link register (LR)**
- **R15 is the program counter (PC)**
- **PSR (Program Status Register)**
  - Not explicitly accessible
  - Saved to the stack on an exception
  - Subsets available as APSR, IPSR, and EPSR

Instruction Set Examples:

- **Data Processing:**

```
MOV    r2, r5                  ; r2 = r5
ADD    r5, #0x24               ; r5 = r5 + 36
ADD    r2, r3, r4, LSL #2      ; r2 = r3 + (r4 * 4)
LSL    r2, #3                  ; r2 = r2 * 8
MOVT r9, #0x1234               ; upper halfword of r9 = #0x1234
MLA    r0, r1, r2, r3          ; r0 = (r1 * r2) + r3
```

- **Memory Access:**

```
STRB r2, [r10, r1]             ; store lower byte in r2 at
                                 address {r10 + r1}
LDR    r0, [r1, r2, LSL #2]    ; load r0 with data at address
                                 {r1 + r2 * 4}
```

- **Program Flow:**

```
BL     <label>                 ; PC relative branch to <label>
                                 location, and return address
                                 stored in LR (r14)
```

Exception Handling

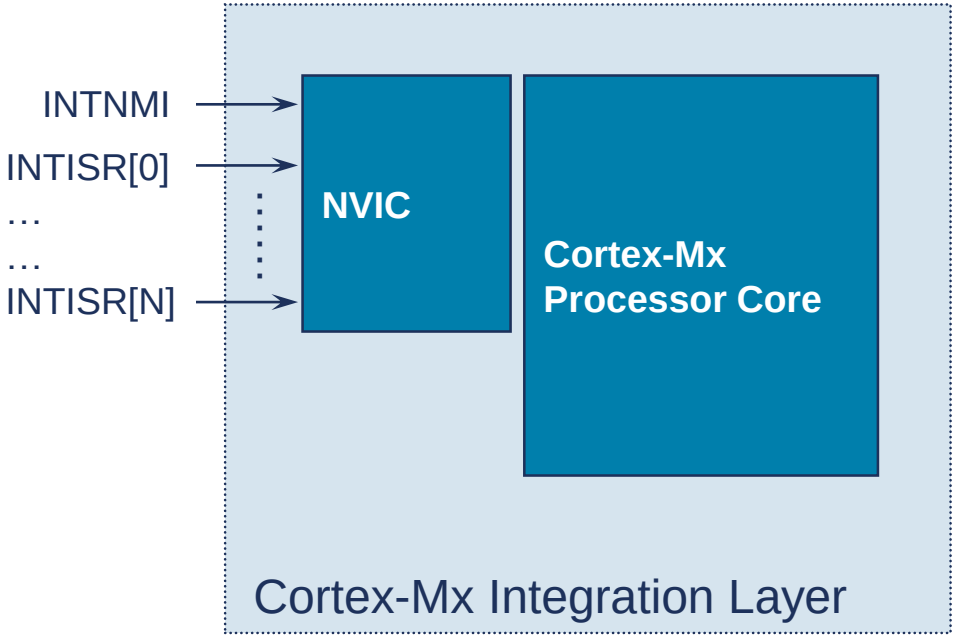
- **Exception types:**
  - Reset
  - Non-maskable Interrupts (NMI)
  - Faults
  - PendSV
  - SVCall
  - External Interrupt
  - SysTick Interrupt

- **Exceptions processed in Handler mode (except Reset)**
  - Exceptions always run privileged

- **Interrupt handling**
  - Interrupts are a sub-class of exception
  - Automatic save and restore of processor registers (xPSR, PC, LR, R12, R3-R0)
  - Allows handler to be written entirely in ‘C’

External Interrupts

- **External Interrupts handled by Nested Vectored Interrupt Controller (NVIC)**
  - Tightly coupled with processor core
- **One Non-Maskable Interrupt (NMI) supported**
- **Number of external interrupts is implementation-defined**
  - ARMv7-M supports up to 496 interrupts



Exception & Pres-emption Ordering

- **Exception handling order is defined by programmable priority**
  - Reset, Non Maskable Interrupt (NMI) and Hard Fault have predefined pre-emption.
  - NVIC catches exceptions and pre-empts current task based on priority

	Exception	Name	Priority	Descriptions
Fault Mode & Start-up Handlers	1	Reset	-3 (Highest)	Reset
	2	NMI	-2	Non-Maskable Interrupt
	3	Hard Fault	-1	Default fault if other hander not implemented
	4	MemManage Fault	Programmable	MPU violation or access to illegal locations
	5	Bus Fault	Programmable	Fault if AHB interface receives error
	6	Usage Fault	Programmable	Exceptions due to program errors
System Handlers	11	SVCall	Programmable	System SerVice call
	12	Debug Monitor	Programmable	Break points, watch points, external debug
	14	PendSV	Programmable	Pendable SerVice request for System Device
	15	Systick	Programmable	System Tick Timer
Custom Handlers	16	Interrupt #0	Programmable	External Interrupt #0
	...	...	...	...
	...	...	...	...
	...	...	...	...
	255	Interrupt #239	Programmable	External Interrupt #239

Vector Table for ARMv7-M

- **First entry contains initial Main SP**
- **All other entries are addresses for exception handlers**
  - Must always have LSBit = 1 (for Thumb)
- **Table has up to 496 external interrupts**
  - Implementation-defined
  - Maximum table size is 2048 bytes
- **Table may be relocated**
  - Use Vector Table Offset Register
  - Still require minimal table entries at 0x0 for booting the core
- **Each exception has a vector number**
  - Used in Interrupt Control and State Register to indicate the active or pending exception type
- **Table can be generated using C code**
  - Example provided later

Address		Vector #
0x40 + 4*N	External N	16 + N
…	…	…
0x40	External 0	16
0x3C	SysTick	15
0x38	PendSV	14
0x34	Reserved	13
0x30	Debug Monitor	12
0x2C	SVC	11
0x1C to 0x28	Reserved (x4)	7-10
0x18	Usage Fault	6
0x14	Bus Fault	5
0x10	Mem Manage Fault	4
0x0C	Hard Fault	3
0x08	NMI	2
0x04	Reset	1
0x00	Initial Main SP	N/A

Vector Table in C

```
typedef void(* const ExecFuncPtr)(void) __irq;

#pragma arm section rodata="exceptions_area"

ExecFuncPtr exception_table[] = {
   (ExecFuncPtr)&Image$$ARM_LIB_STACK$$ZI$$Limit,     /* Initial SP */
   (ExecFuncPtr)__main,                               /* Initial PC */
   NMIException,
   HardFaultException,
   MemManageException,
   BusFaultException,
   UsageFaultException,
   0, 0, 0, 0,                          /* Reserved */
   SVCHandler,
   DebugMonitor,
   0,                                   /* Reserved */
   PendSVC,
   SysTickHandler
   /* Configurable interrupts start here...*/
};
#pragma arm section
```

The vector table at address 0x0 is minimally required to have 4 values: stack top, reset routine location, NMI ISR location, HardFault ISR location

The SVCall ISR location must be populated if the SVC instruction will be used

Once interrupts are enabled, the vector table (whether at 0 or in SRAM) must then have pointers to all enabled (by mask) exceptions

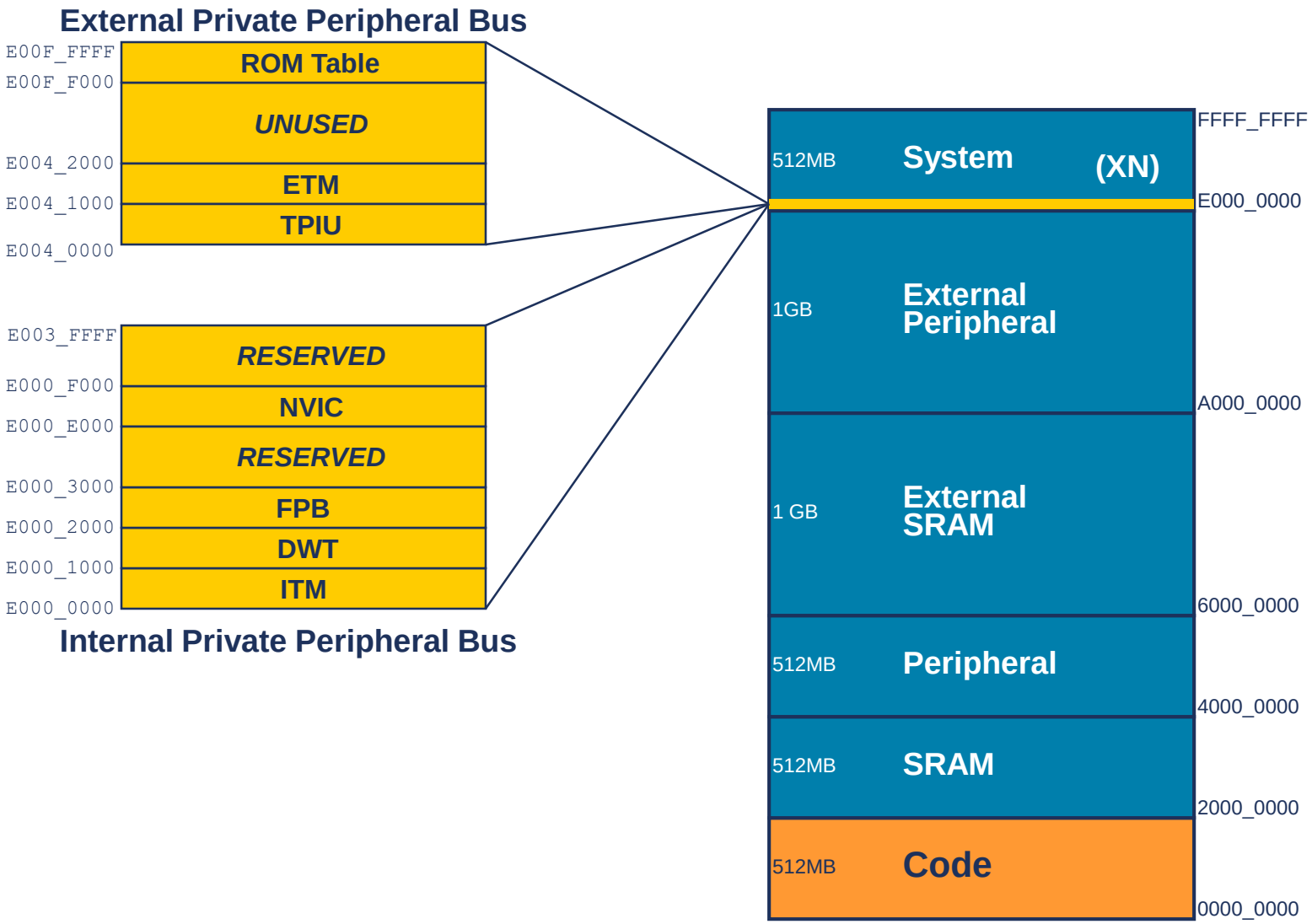
Vector Table in Assembly

```
                PRESERVE8
                THUMB
                IMPORT ||Image$$ARM_LIB_STACK$$ZI$$Limit||
                AREA    RESET, DATA, READONLY
                EXPORT  __Vectors

__Vectors       DCD     ||Image$$ARM_LIB_STACK$$ZI$$Limit||  ; Top of Stack
                DCD     Reset_Handler                ; Reset Handler
                DCD     NMI_Handler                  ; NMI Handler
                DCD     HardFault_Handler            ; Hard Fault Handler
                DCD     MemManage_Handler            ; MemManage Fault Handler
                DCD     BusFault_Handler             ; Bus Fault Handler
                DCD     UsageFault_Handler           ; Usage Fault Handler
                DCD     0, 0, 0, 0,                  ; Reserved x4
                DCD     SVC_Handler,                 ; SVCall Handler
                DCD     Debug_Monitor                ; Debug Monitor Handler
                DCD     0                            ; Reserved
                DCD     PendSV_Handler               ; PendSV Handler
                DCD     SysTick_Handler              ; SysTick Handler
                ; External vectors start here
```

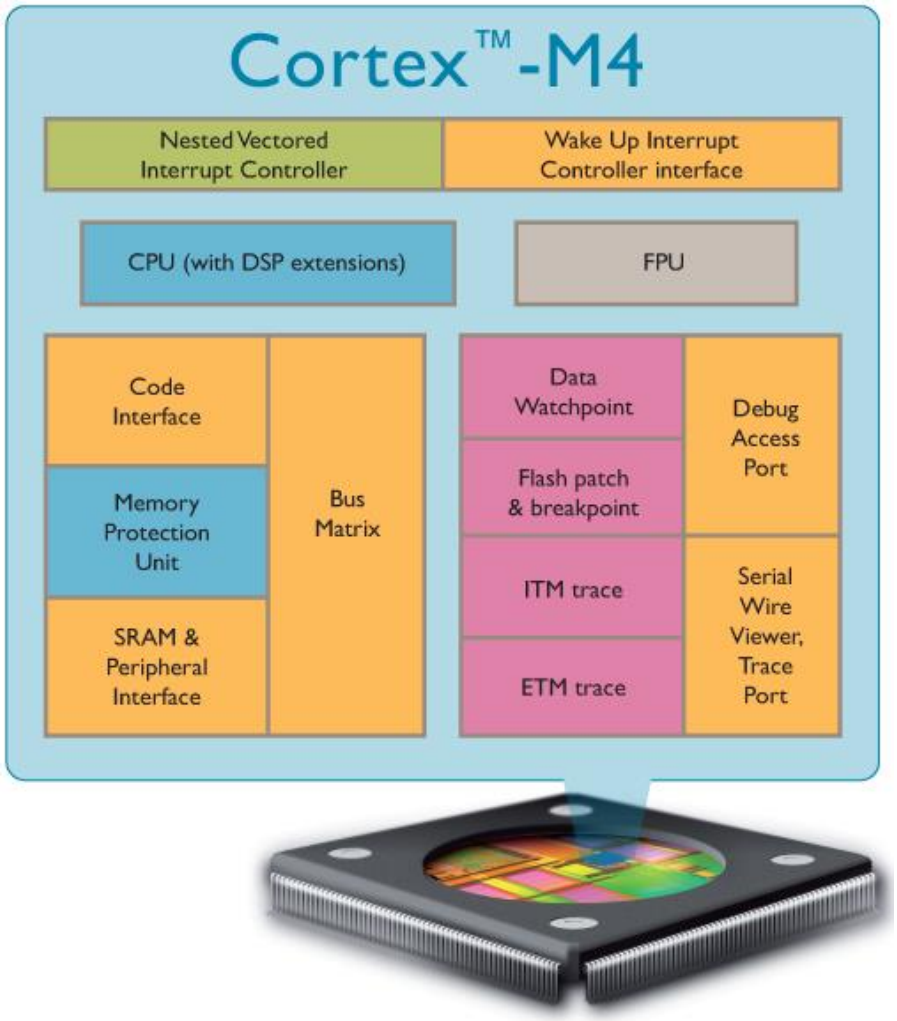
Memory Systems

Processor Memory Map



Floating Point Extensions

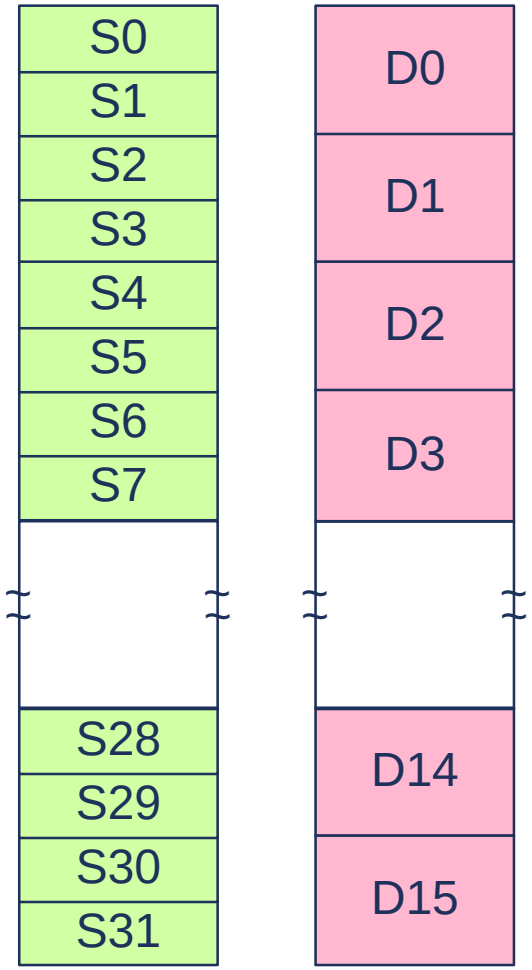
Cortex-M4



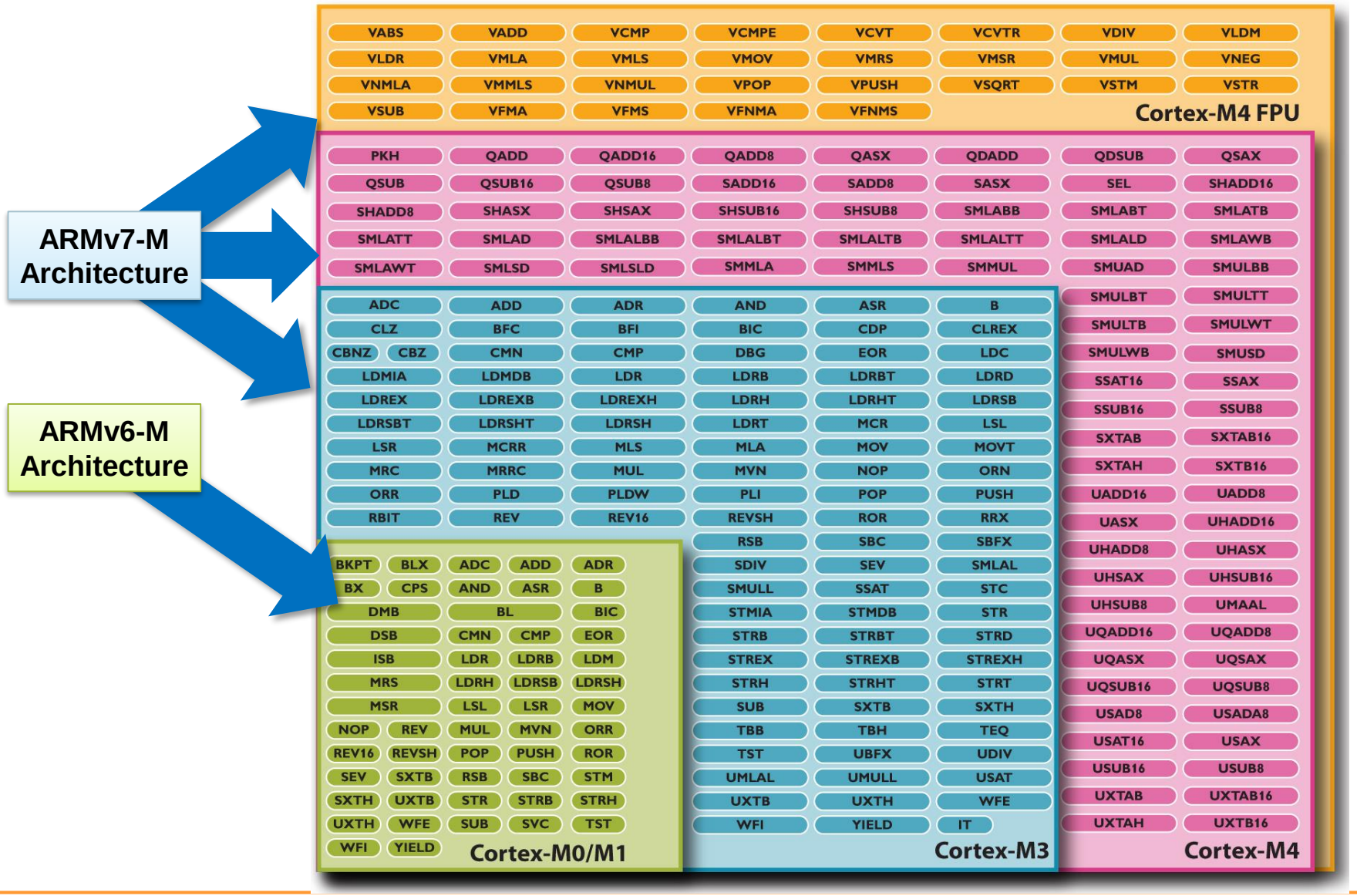
- **ARMv7E-M Architecture**
  - Thumb-2 only
  - DSP extensions
- **Optional FPU (Cortex-M4F)**
- **Otherwise, same as Cortex-M3**
- **Implements full Thumb-2 instruction set**
  - Saturated math (e.g. QADD)
  - Packing and unpacking (e.g. UXTB)
  - Signed multiply (e.g. SMULTB)
  - SIMD (e.g. ADD8)

Cortex-M4F Floating Point Registers

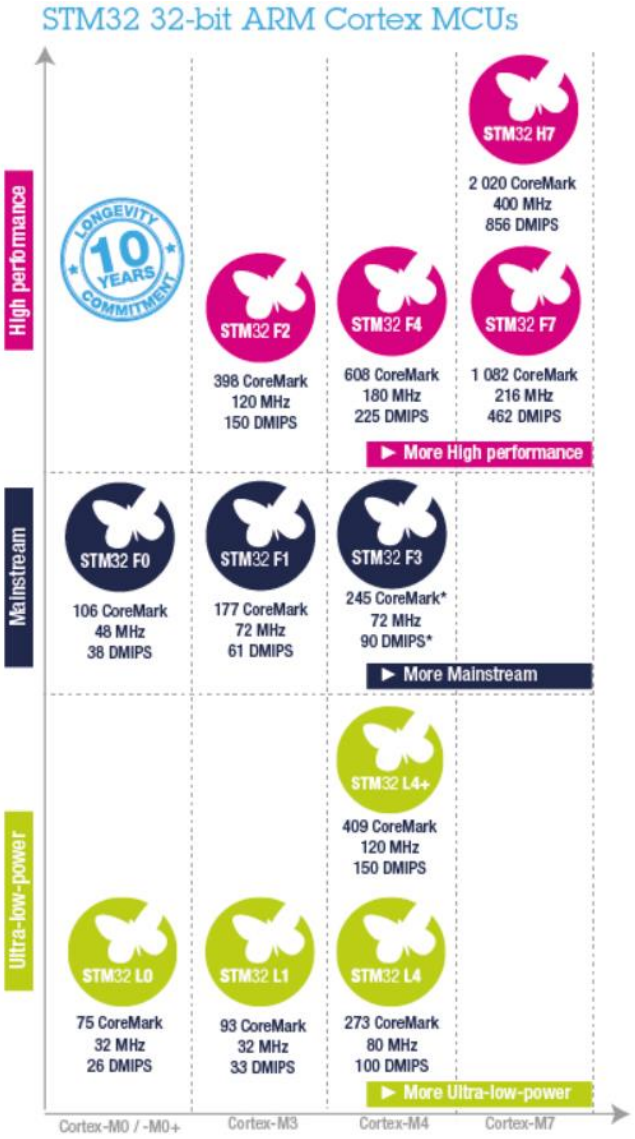
- **FPU provides a further 32 single-precision registers**
- **Can be viewed as either**
  - 32 x 32-bit registers
  - 16 x 64-bit doubleword registers
  - Any combination of the above



Binary Upwards Compatibility

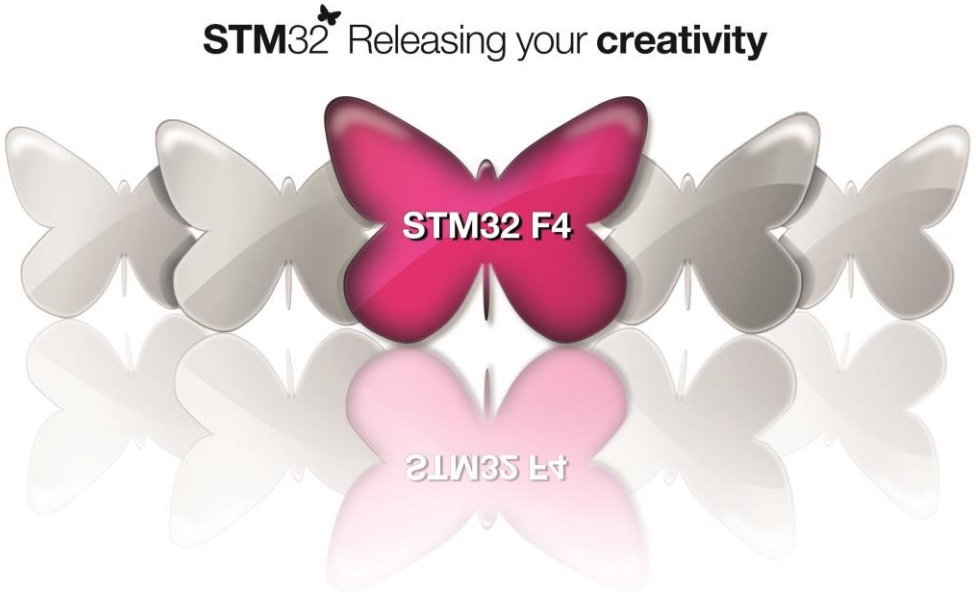


STM32 32-bit ARM Cortex MCUs



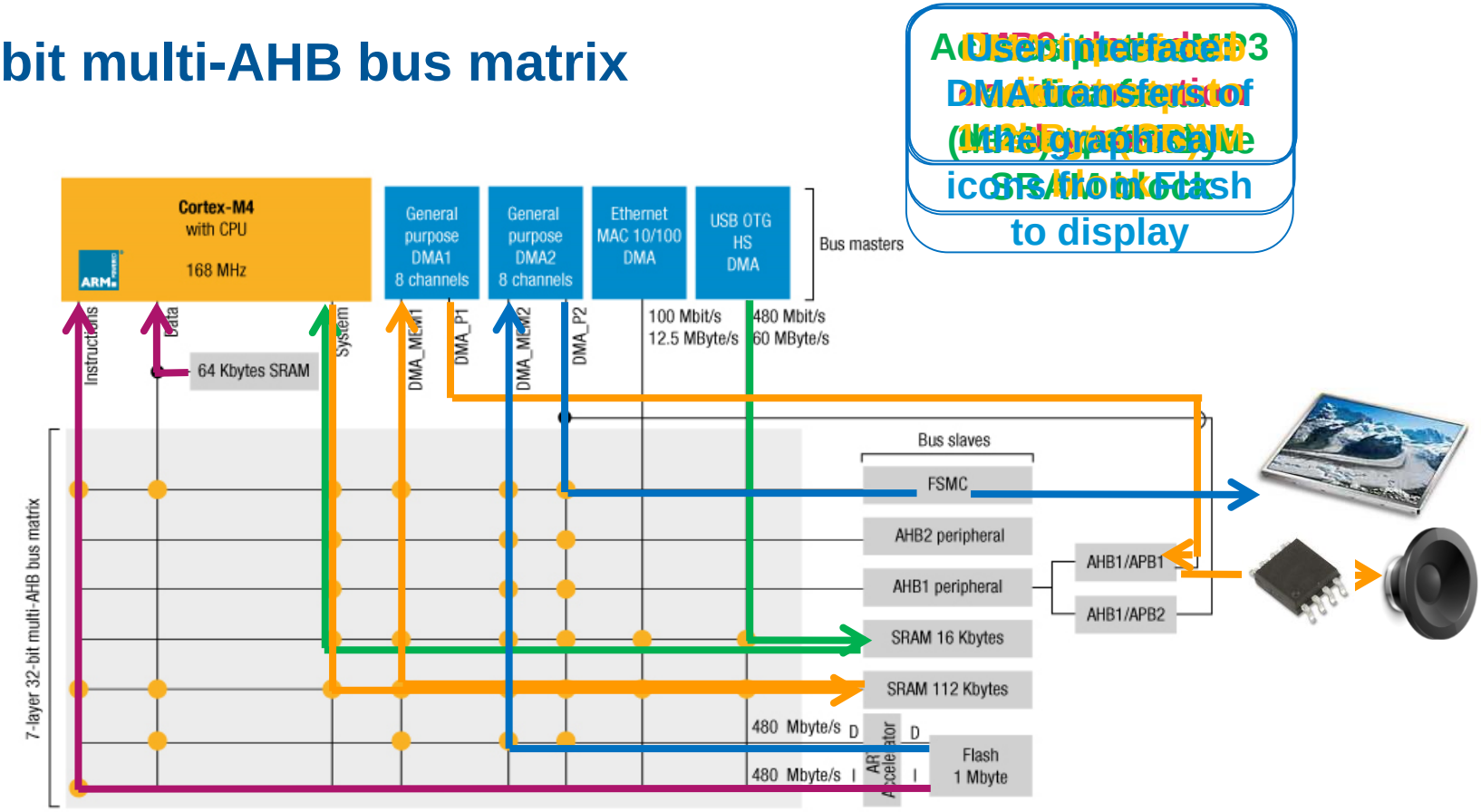
STM32 F4 series

High-performance Cortex™-M4 MCU



Real-time performance

32-bit multi-AHB bus matrix



STM32 F4 series
High-performance digital signal controller

ARM

Cortex
Low-Power Leadership from ARM

FPU

Single precision
Ease of use
Better code efficiency
Faster time to market
Eliminate scaling and saturation
Easier support for meta-language tools

STM32 F4

What is Cortex-M4?

MCU

Ease of use of C programming
Interrupt handling
Ultra-low power

DSP

Harvard architecture
Single-cycle MAC
Barrel shifter

STM32 product series

4 product series

Common core peripherals and architecture:

- Communication peripherals: USART, SPI, I²C
- Multiple general-purpose timers
- Integrated reset and brown-out warning
- Multiple DMA
- 2x watchdogs Real-time clock
- Integrated regulator PLL and clock circuit
- External memory interface (FSMC)
- Dual 12-bit DAC
- Up to 3x 12-bit ADC (up to 0.41 µs)
- Main oscillator and 32 kHz oscillator
- Low-speed and high-speed internal RC oscillators
- -40 to +85 °C and up to 105 °C operating temperature range
- Low voltage 2.0 to 3.6 V or 1.65/1.7 to 3.6 V (depending on series) 5.0 V tolerant I/Os
- Temperature sensor

+

STM32 F4 series - High performance with DSP (STM32F405/415/407/417)

168 MHz Cortex-M4 with DSP and FPU	Up to 192-Kbyte SRAM	Up to 1-Mbyte Flash	2x USB 2.0 OTG FS/HS	3-phase MC timer	2x CAN 2.0B	SDIO 2x I²S audio Camera IF	Ethernet IEEE 1588	Crypto/hash processor and RNG

STM32 F2 series - High performance (STM32F205/215/207/217)

120 MHz Cortex-M3 CPU	Up to 128-Kbyte SRAM	Up to 1-Mbyte Flash	2x USB 2.0 OTG FS/HS	3-phase MC timer	2x CAN 2.0B	SDIO 2x I²S audio Camera IF	Ethernet IEEE 1588	Crypto/hash processor and RNG

STM32 F1 series - Connectivity line (STM32F105/107)

72 MHz Cortex-M3 CPU	Up to 64-Kbyte SRAM	Up to 256-Kbyte Flash	USB 2.0 OTG FS	3-phase MC timer	2x CAN 2.0B	2x I²S audio	Ethernet IEEE 1588

STM32 F1 series - Performance line (STM32F103)

72 MHz Cortex-M3 CPU	Up to 96-Kbyte SRAM	Up to 1-Mbyte Flash	USB FS device	3-phase MC timer	CAN 2.0B	SDIO 2x I²S

STM32 F1 series - USB Access line (STM32F102)

48 MHz Cortex-M3 CPU	Up to 16-Kbyte SRAM	Up to 128-Kbyte Flash	USB FS device

STM32 F1 series - Access line (STM32F101)

36 MHz Cortex-M3 CPU	Up to 80-Kbyte SRAM	Up to 1-Mbyte Flash

STM32 F1 series - Value line (STM32F100)

24 MHz Cortex-M3 CPU	Up to 32-Kbyte SRAM	Up to 512-Kbyte Flash	3-phase MC timer	CEC

STM32 L1 series - Ultra-low-power (STM32F151/152)

32 MHz Cortex-M3 CPU	Up to 48-Kbyte SRAM	Up to 384-Kbyte Flash	USB FS device	Data EEPROM up to 12 Kbytes	LCD 8x40 4x44	Comparator	BOR MSI VScal	AES 128-bit

STM32 F4 block diagram

Feature highlight

- 168 MHz Cortex-M4 CPU
  - Floating point unit (FPU)
  - ART Accelerator ™
  - Multi-level AHB bus matrix
- 1-Mbyte Flash,
  192-Kbyte SRAM
- 1.7 to 3.6 V supply
- RTC: <1 µA typ, sub second accuracy
- 2x full duplex I²S
- 3x 12-bit ADC
  0.41 µs/2.4 MSPS
- 168 MHz timers

Notes:
1. HS requires an external PHY connected to the ULPI interface
2. Crypto/hash processor on STM32F417 and STM32F415