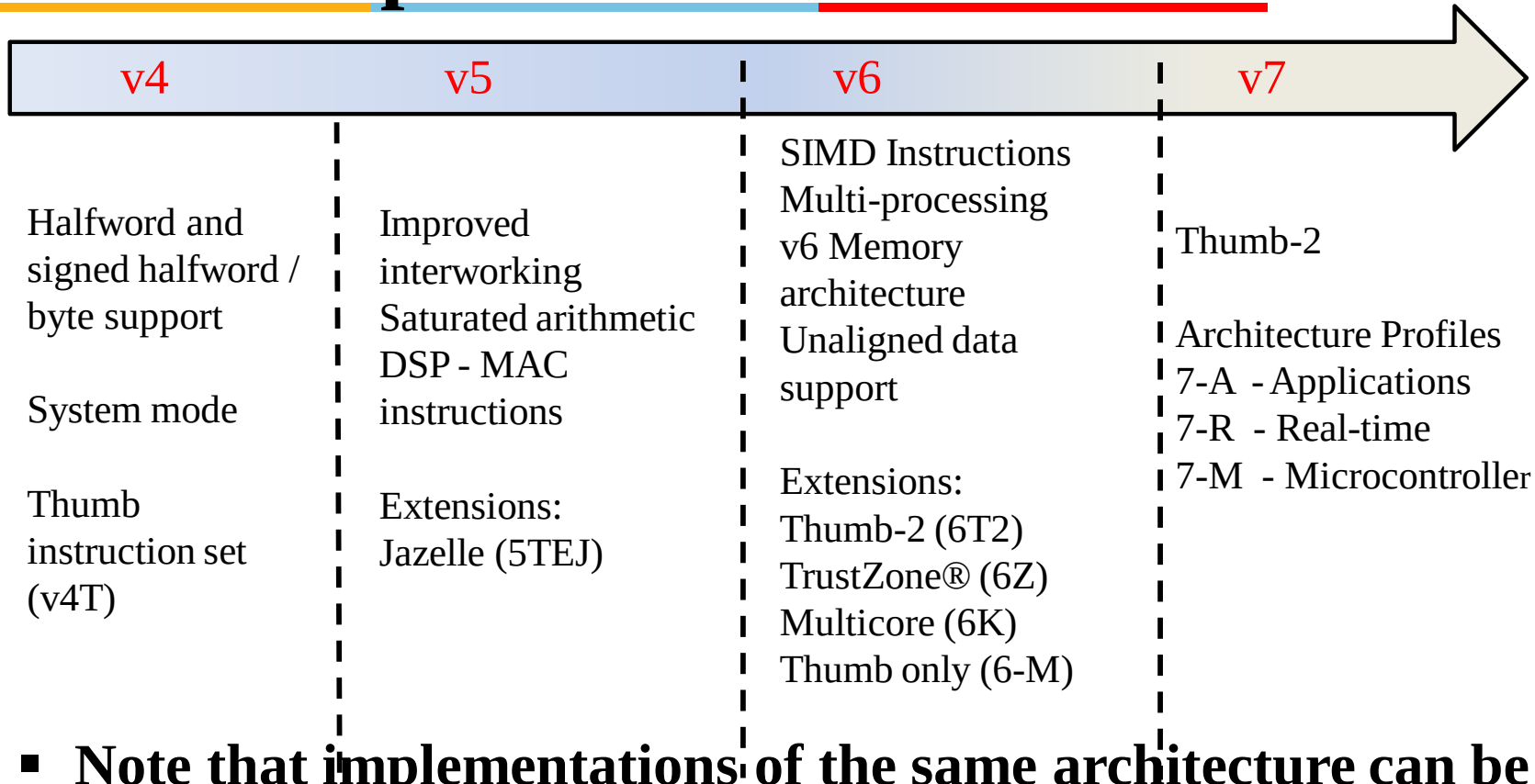
Introduction to ARM Cortex M4 Microcontroller

- Introduction to ARM CORTEX series: CORTEX A, R, M processors
- Firmware development using CMSIS Standard.
- Introduction to ARM CORTEX M4 microprocessor core, programmer model, Processor Modes, Memory Map, Introduction Arm Cortex-M cores,
- STM32F4xx Architecture, ARM STM Bus Architecture, STM32F4xx Clock and SYSCLK, Peripheral Clock, PLL clock, Interrupts and Exceptions in STM32F4xx.

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

ARM Processor Families

Cortex-A series (Application)

- High performance processors capable of full Operating System (OS) support;
- Applications include smartphones, digital TV, smart books, home gateways etc.

Cortex-R series (Real-time)

- High performance for real-time applications;
- High reliability
- Applications include automotive braking system, powertrains etc.

Cortex-M series (Microcontroller)

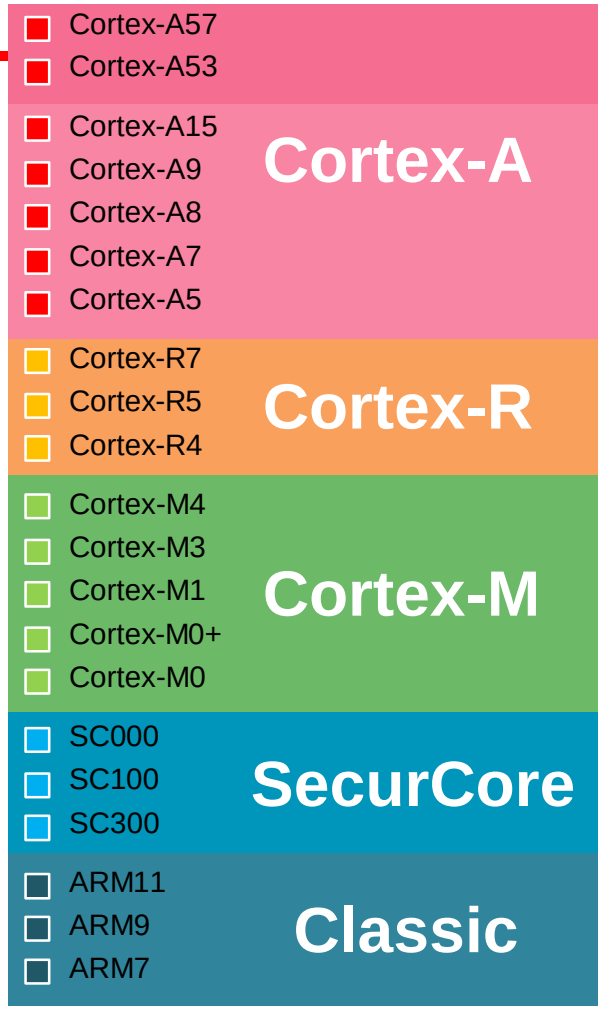
- Cost-sensitive solutions for deterministic microcontroller applications;
- Applications include microcontrollers, mixed signal devices, smart sensors, automotive body electronics and airbags;

SecurCore series

- High security applications.

Previous classic processors

- Include ARM7, ARM9, ARM11 families



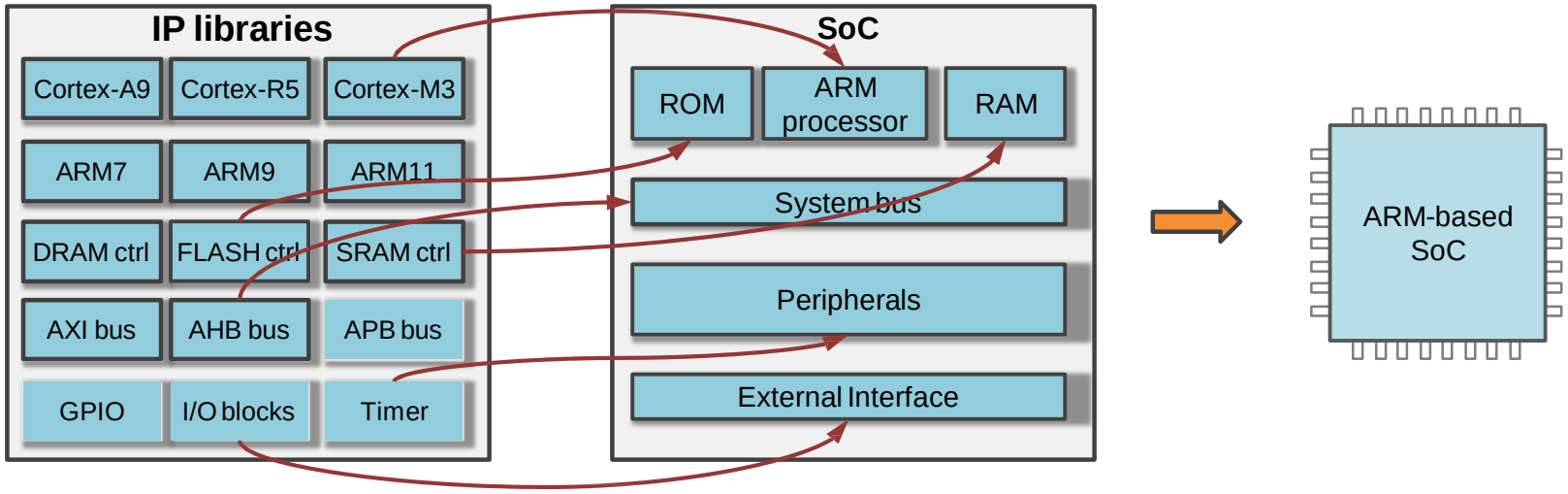
Cortex
Low-Power Leadership from ARM

As of Dec 2013

Source: ARM

Design an ARM-based SoC

- Select a set of IP cores from ARM and/or other third-party IP vendors
- Integrate IP cores into a single chip design
- Give design to semiconductor foundries for chip fabrication



Source: ARM

ARM Cortex-M Series

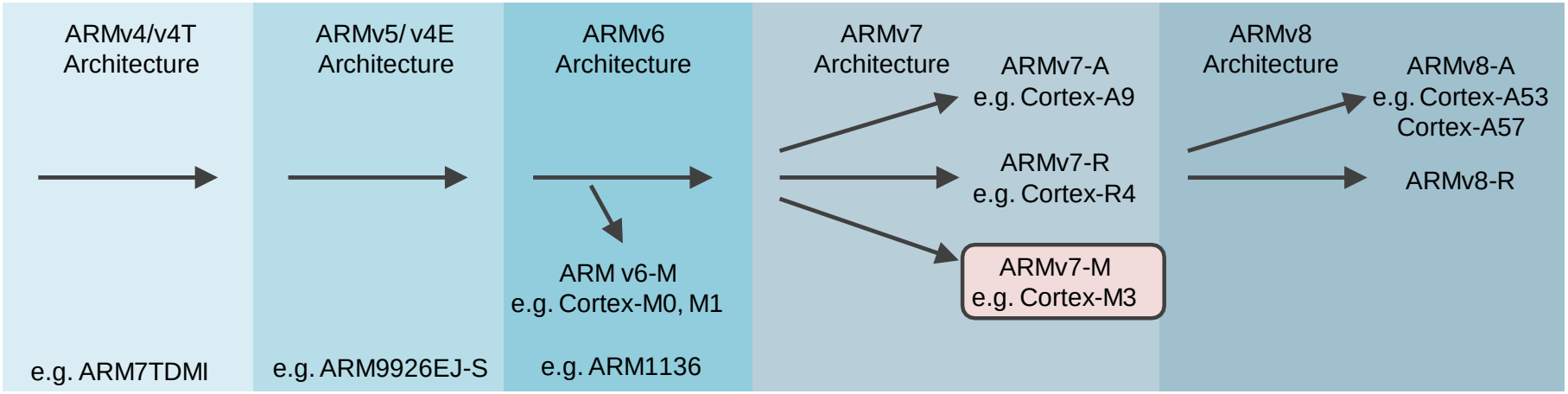
- Cortex-M series: Cortex-M0, M0+, M1, M3, M4,M7
- Energy-efficiency
  - Lower energy cost, longer battery life
- Smaller code
  - Lower silicon costs
- Ease of use
  - Faster software development and reuse
- Embedded applications
  - Smart metering, human interface devices, automotive and industrial control systems, white goods, consumer products and medical instrumentation

As of Dec 2013

Source: ARM

ARM Processors vs. ARM Architectures

- ARM architecture
  - Describes the details of instruction set, programmer's model, exception model, and memory map
  - Documented in the Architecture Reference Manual
- ARM processor
  - Developed using one of the ARM architectures
  - More implementation details, such as timing information
  - Documented in processor's Technical Reference Manual



As of Dec 2013

Source: ARM

ARM Cortex-M Series Family

Processor	ARM Architecture	Core Architecture	Thumb®	Thumb®-2	Hardware Multiply	Hardware Divide	Saturated Math	DSP Extensions	Floating Point
Cortex-M0	ARMv6-M	Von Neumann	Most	Subset	1 or 32 cycle	No	No	No	No
Cortex-M0+	ARMv6-M	Von Neumann	Most	Subset	1 or 32 cycle	No	No	No	No
Cortex-M1	ARMv6-M	Von Neumann	Most	Subset	3 or 33 cycle	No	No	No	No
Cortex-M3	ARMv7-M	Harvard	Entire	Entire	1 cycle	Yes	Yes	No	No
Cortex-M4	ARMv7E-M	Harvard	Entire	Entire	1 cycle	Yes	Yes	Yes	Optional

Source: ARM

ARM Cortex-M Series Family

- Latest generation Cortex-M processor is the Cortex-M55.
- The Cortex-M55 is the first processor built on the Armv8.1-M architecture with Arm Helium technology, a vector processing extension. It brings enhanced levels of machine learning and signal processing performance to the next wave of small embedded devices, including wearables, smart speakers, and more.
- Cortex-M55
- The Arm Cortex-M55 processor is Arm's most AI-capable Cortex-M processor and the first to feature Arm Helium vector processing technology.

Source: ARM

ARM Cortex-M Series Family

- Cortex-M35P
- A tamper-resistant Cortex-M processor with optional software isolation using TrustZone for Armv8-M.
- Cortex-M33
- The Arm Cortex-M33 processor has an ideal blend of real-time determinism, efficiency and security.
- Cortex-M23
- The Arm Cortex-M23 is the smallest and lowest power microcontroller with TrustZone security.
- Cortex-M7
- The Arm Cortex-M7 processor is the highest performance member of the energy-efficient Cortex-M processor family.

Source: ARM

Introduction to ARM Cortex-M Processors

- The Cortex-M3 processor was the first of the Cortex generation of processors, released by ARM in 2005.
- The Cortex-M4 processor was released in 2010.
- The Cortex-M4 processors use a 32-bit architecture.
- Internal registers in the register bank, the data path, and the bus interfaces are all 32 bits wide.
- The Instruction Set Architecture (ISA) in the Cortex-M processors is called the Thumb ISA and is based on Thumb-2 Technology which supports a mixture of 16-bit and 32-bit instructions.

Introduction to ARM Cortex M4 Architecture

- Cortex-M4 processors have:
- Three-stage pipeline design
- Harvard bus architecture with unified memory space: instructions and data use the same address space
- 32-bit addressing, supporting 4GB of memory space
- On-chip bus interfaces based on ARM AMBA (Advanced Microcontroller Bus Architecture) Technology, which allow pipelined bus operations for higher throughput
- An interrupt controller called NVIC (Nested Vectored Interrupt Controller) supporting up to 240 interrupt requests and from 8 to 256 interrupt priority levels (dependent on the actual device implementation)

Introduction to ARM Cortex M4 Architecture

- Cortex-M4 processors have:
- Support for various features for OS (Operating System) implementation such as a system tick timer, shadowed stack pointer
- Sleep mode support and various low power features
- Support for an optional MPU (Memory Protection Unit) to provide memory protection features like programmable memory, or access permission control
- Support for bit-data accesses in two specific memory regions using a feature called Bit Band
- The option of being used in single processor or multi-processor designs

Introduction to ARM Cortex M4 Architecture

- The ISA used in Cortex-M4 processors provides a wide range of instructions:
- General data processing, including hardware divide instructions
- Memory access instructions supporting 8-bit, 16-bit, 32-bit, and 64-bit data, as well as instructions for transferring multiple 32-bit data
- Instructions for bit field processing
- Multiply Accumulate (MAC) and saturate instructions
- Instructions for branches, conditional branches and function calls
- Instructions for system control, OS support, etc.

Introduction to ARM Cortex M4 Architecture

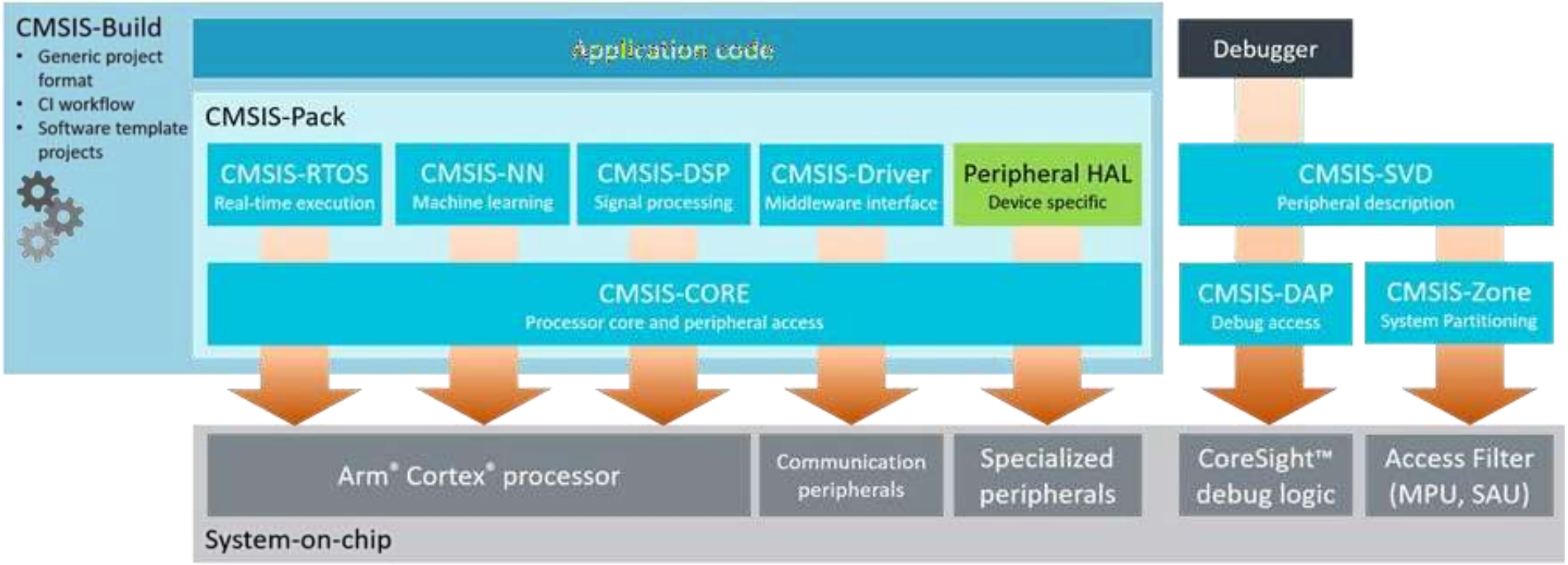
- In addition, the Cortex-M4 processor also supports:
- Single Instruction Multiple Data (SIMD) operations
- Additional fast MAC and multiply instructions
- Saturating arithmetic instructions
- Optional floating point instructions (single precision)

Firmware development using CMSIS Standard (Cortex Microcontroller Software Interface standard)

- The **CMSIS** is a set of tools, APIs, frameworks, and work flows that help to simplify software re-use, reduce the learning curve for microcontroller developers, speed-up project build and debug, and thus reduce the time to market for new applications.
- CMSIS started as a vendor-independent hardware abstraction layer Arm® Cortex®-M based processors and was later extended to support entry-level Arm Cortex-A based processors. To simplify access, CMSIS defines generic tool interfaces and enables consistent device support by providing simple software interfaces to the processor and the peripherals.
- CMSIS is defined in close cooperation with various silicon and software vendors and provides a common approach to interface to peripherals, real-time operating systems, and middleware components. It is intended to enable the combination of software components from multiple vendors.
- CMSIS is open-source and collaboratively developed on GitHub.

Firmware development using CMSIS Standard

(Cortex Microcontroller Software Interface standard)



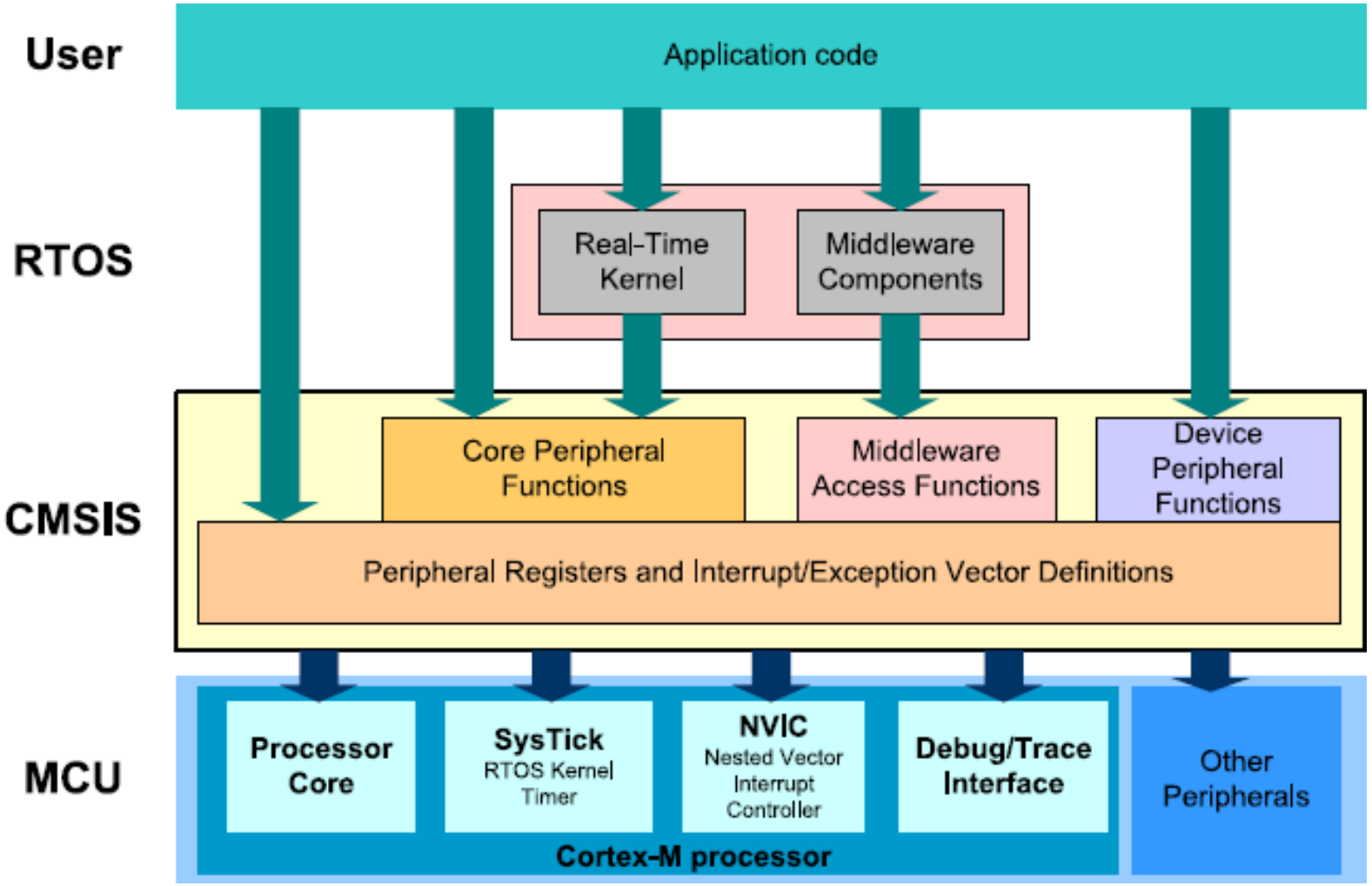
- CMSIS has been created to help the industry in standardization. It enables consistent software layers and device support across a wide range of development tools and microcontrollers. CMSIS is not a huge software layer that introduces overhead and does not define standard peripherals. The silicon industry can therefore support the wide variations of Arm Cortex processor-based devices with this common standard.

Firmware development using CMSIS Standard (Cortex Microcontroller Software Interface standard)

- The aims of CMSIS include:
- Enhanced software reusability - makes it easier to reuse software code in different Cortex-M projects, reducing time to market and verification efforts.
- Enhanced software compatibility - by having a consistent software infrastructure (e.g., API for processor core access functions, system initialization method, common style for defining peripherals), software from various sources can work together, reducing the risk in integration.
- Easy to learn - the CMSIS allows easy access to processor core features from the C language. In addition, once you learn to use one Cortex-M microcontroller product, starting to use another Cortex-M product is much easier because of the consistency in software setup.
- Toolchain independent - CMSIS-compliant device drivers can be used with various compilation tools, providing much greater freedom.
- Openness - the source code for CMSIS core files can be downloaded and accessed by everyone, and everyone can develop software products with CMSIS.

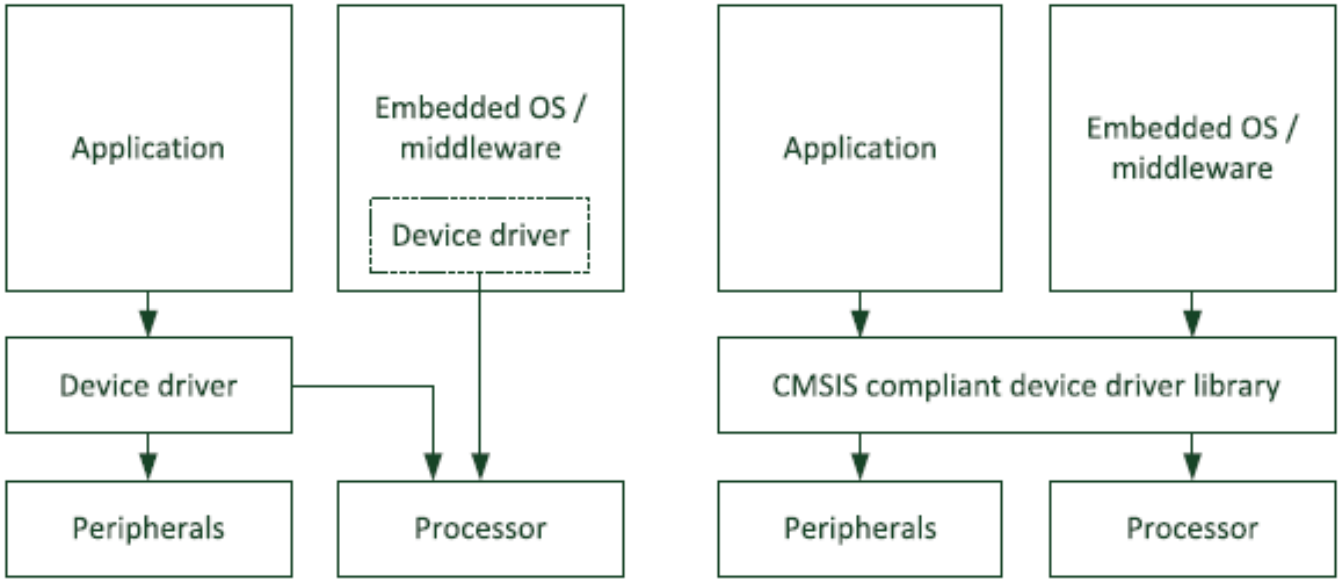
Firmware development using CMSIS Standard

(Cortex Microcontroller Software Interface standard)



Firmware development using CMSIS Standard

(Cortex Microcontroller Software Interface standard)

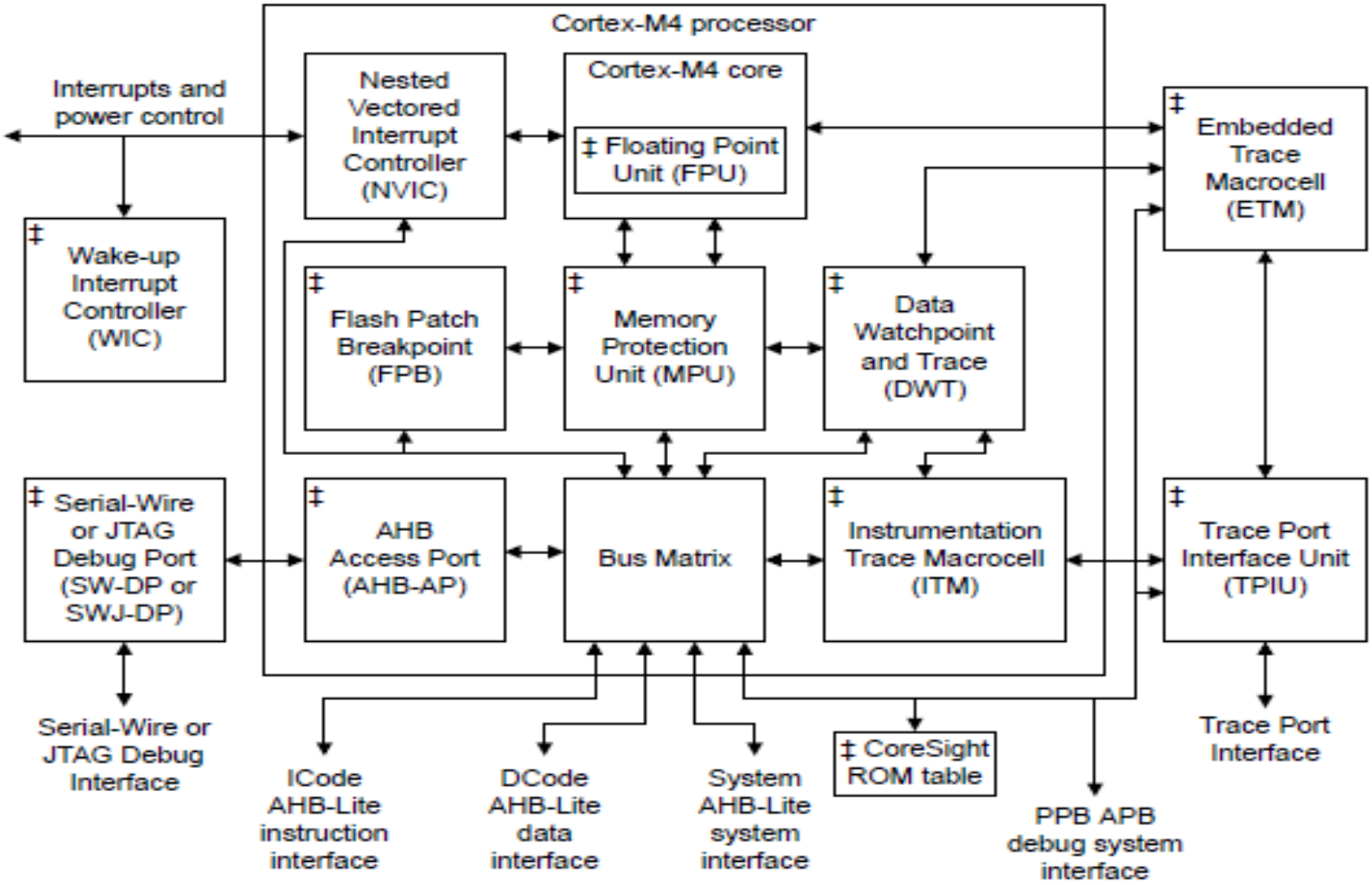


Without CMSIS, an embedded OS or middleware needs to include access functions to use processor features.

With CMSIS, an embedded OS or middleware can use standardized core access functions from device driver library

- CMSIS-Core avoids the need for middleware or OS to carry their own driver code.
- You can download the latest version of the CMSIS source package from http://www.arm.com/cmsis.

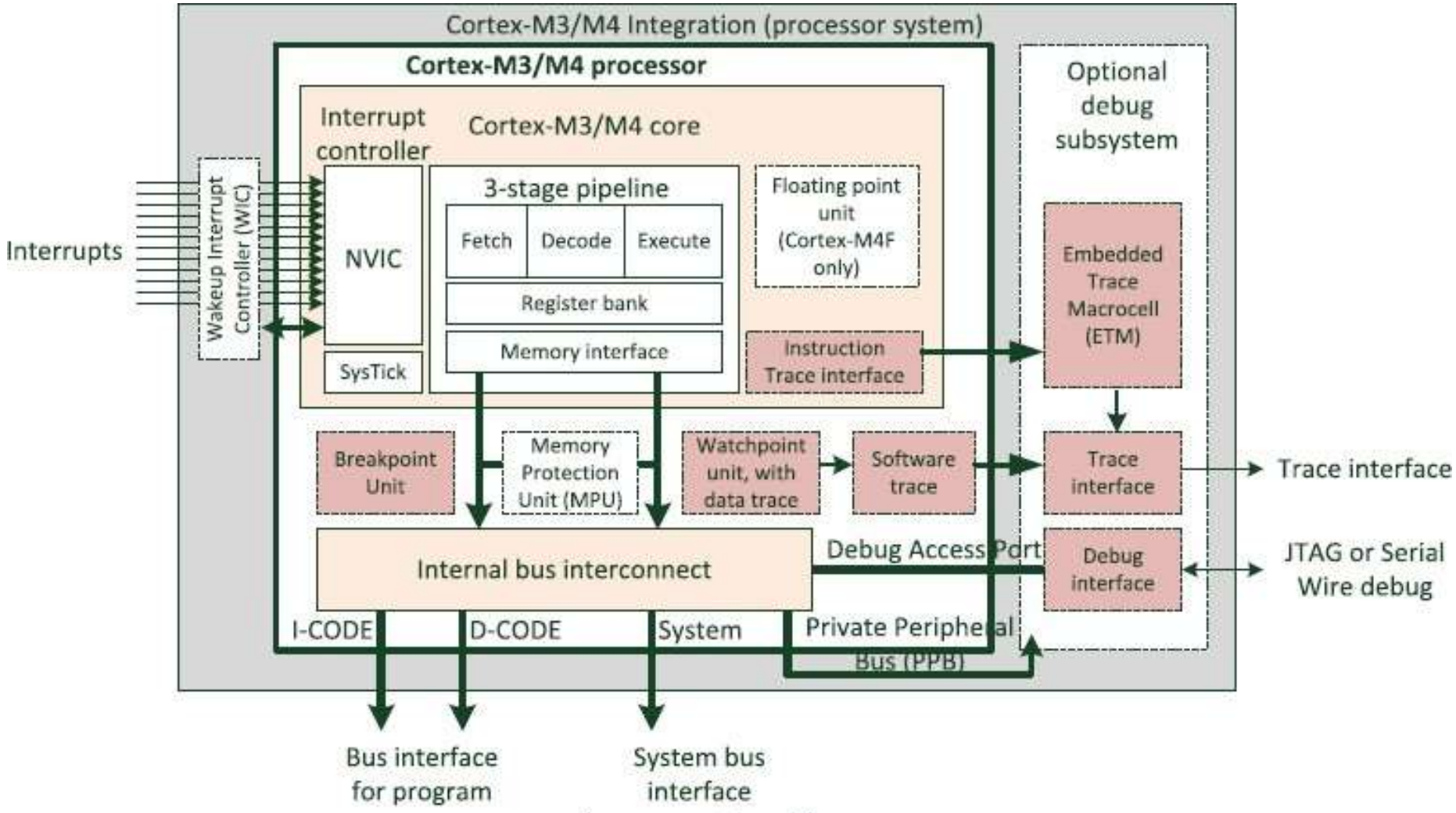
ARM Cortex M4 Architecture



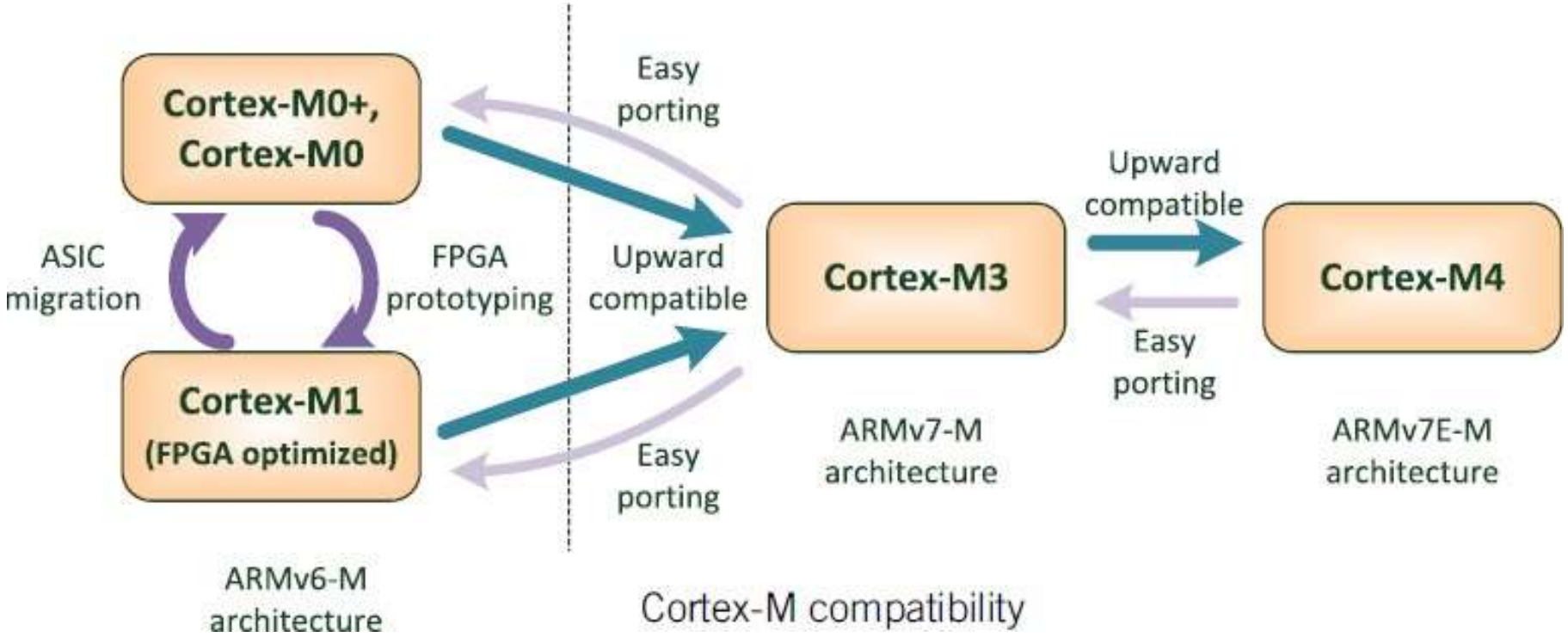
‡ Optional component

Block diagram showing the structure of the Cortex-M4 processor.

ARM Cortex M4 Architecture



ARM Cortex M Compatibility



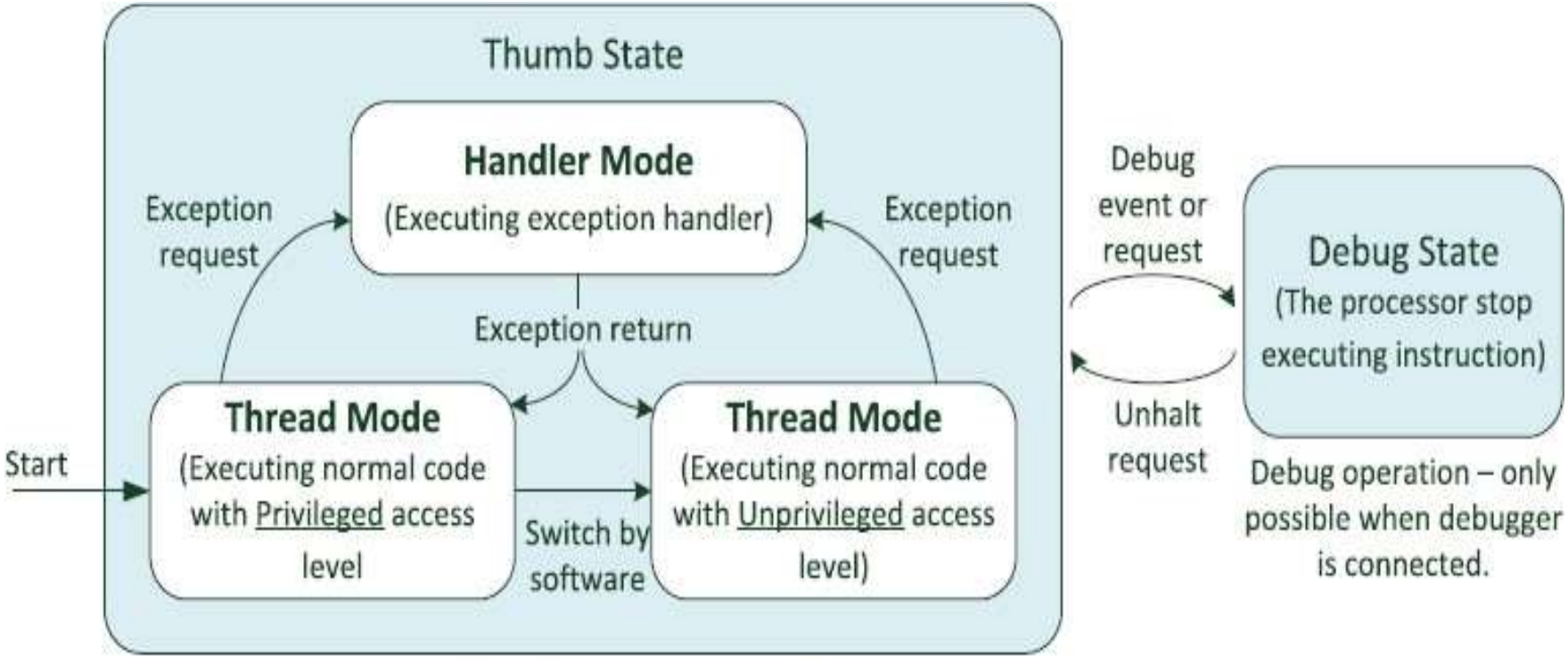
Programmer's model : Operation modes and states

- The Cortex-M4 processors have two operation states and two modes.
- In addition, the processors can have privileged and unprivileged access levels.
- The privileged access level can access all resources in the processor, while unprivileged access level means some memory regions are inaccessible, and a few operations cannot be used.
- The unprivileged access level might also be referred as "User" state, a term inherited from ARM7TDMI.

Programmer's model : Operation modes and states

- **Operation states :**
  - Debug state: When the processor is halted (e.g., by the debugger, or after hitting a breakpoint), it enters debug state and stops executing instructions.
  - Thumb state: If the processor is running program code (Thumb instructions), it is in the Thumb state.
  - Unlike classic ARM processors like ARM7TDMI, there is no ARM state because the Cortex-M processors do not support the ARM instruction set.

Programmer's model : Operation modes and states



Operation states and modes

Programmer's model : Operation modes and states

- **Operation modes**
  - **Handler mode:** When executing an exception handler such as an Interrupt Service Routine (ISR). When in handler mode, the processor always has privileged access level.
  - **Thread mode:** When executing normal application code, the processor can be either in privileged access level or unprivileged access level. This is controlled by a special register called "CONTROL."

Programmer's model : Operation modes and states

- Software can switch the processor in privileged Thread mode to unprivileged Thread mode.
- However, it cannot switch itself back from unprivileged to privileged.
- If this is needed, the processor has to use the exception mechanism to handle the switch.
- The separation of privileged and unprivileged access levels allows system designers to develop robust embedded systems by providing a mechanism to safeguard memory accesses to critical regions and by providing a basic security model.

Programmer's model : Operation modes and states

- For example, a system can contain an embedded OS kernel that executes in privileged access level, and application tasks which execute in unprivileged access level.
- In this way, we can set up memory access permissions using the Memory Protection Unit (MPU) to prevent an application task from corrupting memory and peripherals used by the OS kernel and other tasks.
- If an application task crashes, the remaining application tasks and the OS kernel can still continue to run.

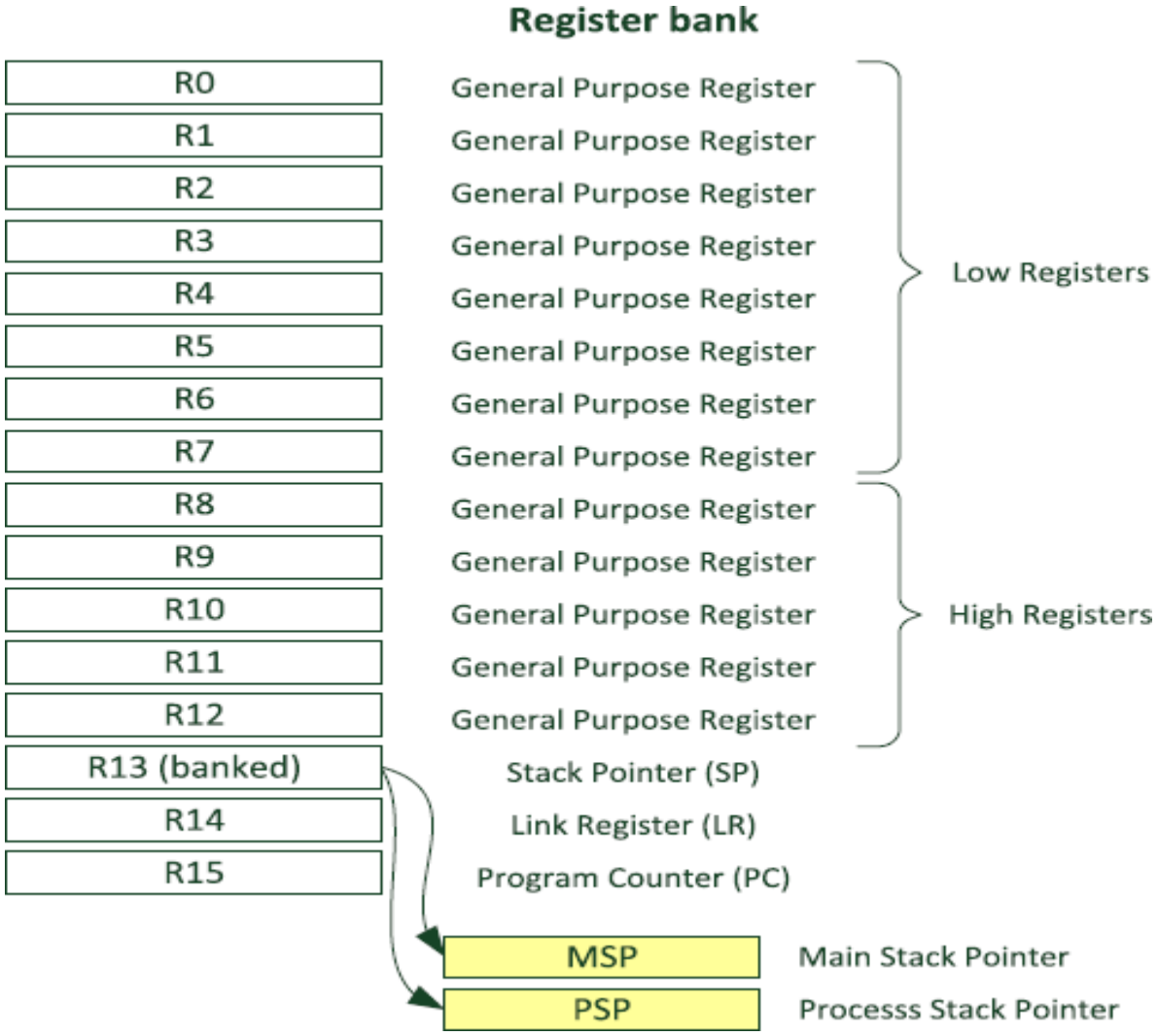
Programmer's model : Operation modes and states

- Thread mode and Handler mode have very similar programmer's models.
- However, Thread mode can switch to using a separate shadowed Stack Pointer (SP).
- Again, this allows the stack memory for application tasks to be separated from the stack used by the OS kernel, thus allowing better system reliability.
- By default, the Cortex-M processors start in privileged Thread mode and in Thumb state.
- In many simple applications, there is no need to use the unprivileged Thread model and the shadowed SP at all.

Programmer's model : Operation modes and states

- The debug state is used for debugging operations only.
- This state is entered by a halt request from the debugger, or by debug events generated from debug components in the processor.
- This state allows the debugger to access or change the processor register values.
- The system memory, including peripherals inside and outside the processor, can be accessed by the debugger in either Thumb state or debug state.

Programmer's model : Registers



- The register bank in the Cortex-M4 processors has 16 registers.
- Thirteen of them are general purpose 32-bit registers, and the other three have special uses

Programmer's model : Registers

- **R0 - R12**
- Registers R0 to R12 are general purpose registers.
- The first eight (R0 - R7) are also called low registers.
- Due to the limited available space in the instruction set, many 16-bit instructions can only access the low registers.
- The high registers (R8 - R12) can be used with 32-bit instructions, and a few with 16-bit instructions, like MOV (move). The initial values of R0 to R12 are undefined.

Programmer's model : Registers

- **R13, stack pointer (SP)**
- It is used for accessing the stack memory via PUSH and POP operations.
- Physically there are two different Stack Pointers: the Main Stack Pointer (MSP, or SP_main in some ARM documentation) is the default Stack Pointer. It is selected after reset, or when the processor is in Handler Mode.
- The other Stack Pointer is called the Process Stack Pointer (PSP, or SP_process in some ARM documentation). The PSP can only be used in Thread Mode. The selection of Stack Pointer is determined by a special register called CONTROL.

Programmer's model : Registers

- Both MSP and PSP are 32-bit, but the lowest two bits of the Stack Pointers (either MSP or PSP) are always zero, and writes to these two bits are ignored.
- In ARM Cortex-M processors, PUSH and POP are always 32-bit, and the addresses of the transfers in stack operations must be aligned to 32-bit word boundaries.
- For most cases, it is not necessary to use the PSP if the application doesn't require an embedded OS.
- Many simple applications can rely on the MSP completely.
- The PSP is normally used when an embedded OS is involved, where the stack for the OS kernel and application tasks are separated.
- The initial value of PSP is undefined, and the initial value of MSP is taken from the first word of the memory during the reset sequence.

Programmer's model : Registers

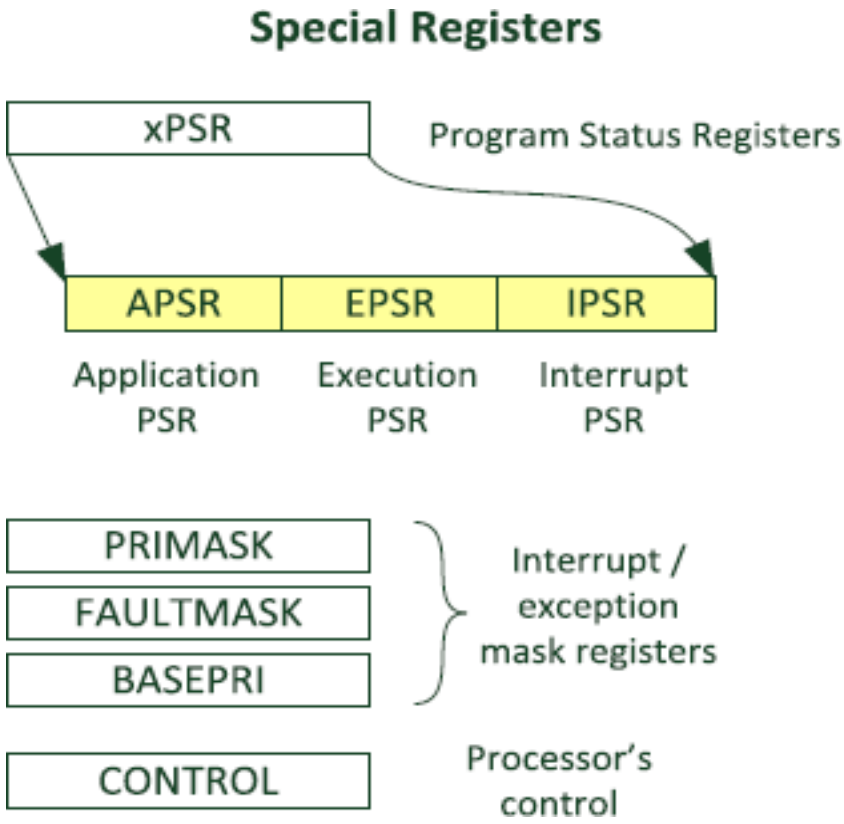
- **R14, link register (LR)**
- R14 is also called the Link Register (LR). This is used for holding the return address when calling a function or subroutine.
- At the end of the function or subroutine, the program control can return to the calling program and resume by loading the value of LR into the Program Counter (PC).
- When a function or subroutine call is made, the value of LR is updated automatically.
- If a function needs to call another function or subroutine, it needs to save the value of LR in the stack first.
- Otherwise, the current value in LR will be lost when the function call is made.

Programmer's model : Registers

- **R15, program counter (PC)**
- It is readable and writeable: a read returns the current instruction address plus 4 (this is due to the pipeline nature of the design, and compatibility requirement with the ARM7TDMI processor).
- Writing to PC (e.g., using data transfer/processing instructions) causes a branch operation.
- Since the instructions must be aligned to half-word or word addresses, the Least Significant Bit (LSB) of the PC is zero.

Programmer's model : Special registers

- **Program status registers**
- The Program Status Register is composed of three status registers:
- Application PSR (APSR)
- Execution PSR (EPSR)
- Interrupt PSR (IPSR)



Memory system: Memory system features

- 4GB linear address space, with 32-bit addressing, the ARM processors can access up to 4GB of memory space.
- While many embedded systems do not need more than 1MB of memory, the 32-bit addressing capability ensures future upgrade and expansion possibilities.
- The Cortex-M4 processors provide 32-bit buses using a generic bus protocol called AHB LITE.
- The bus allows connections to 32/16/8-bit memory devices with suitable memory interface controllers.

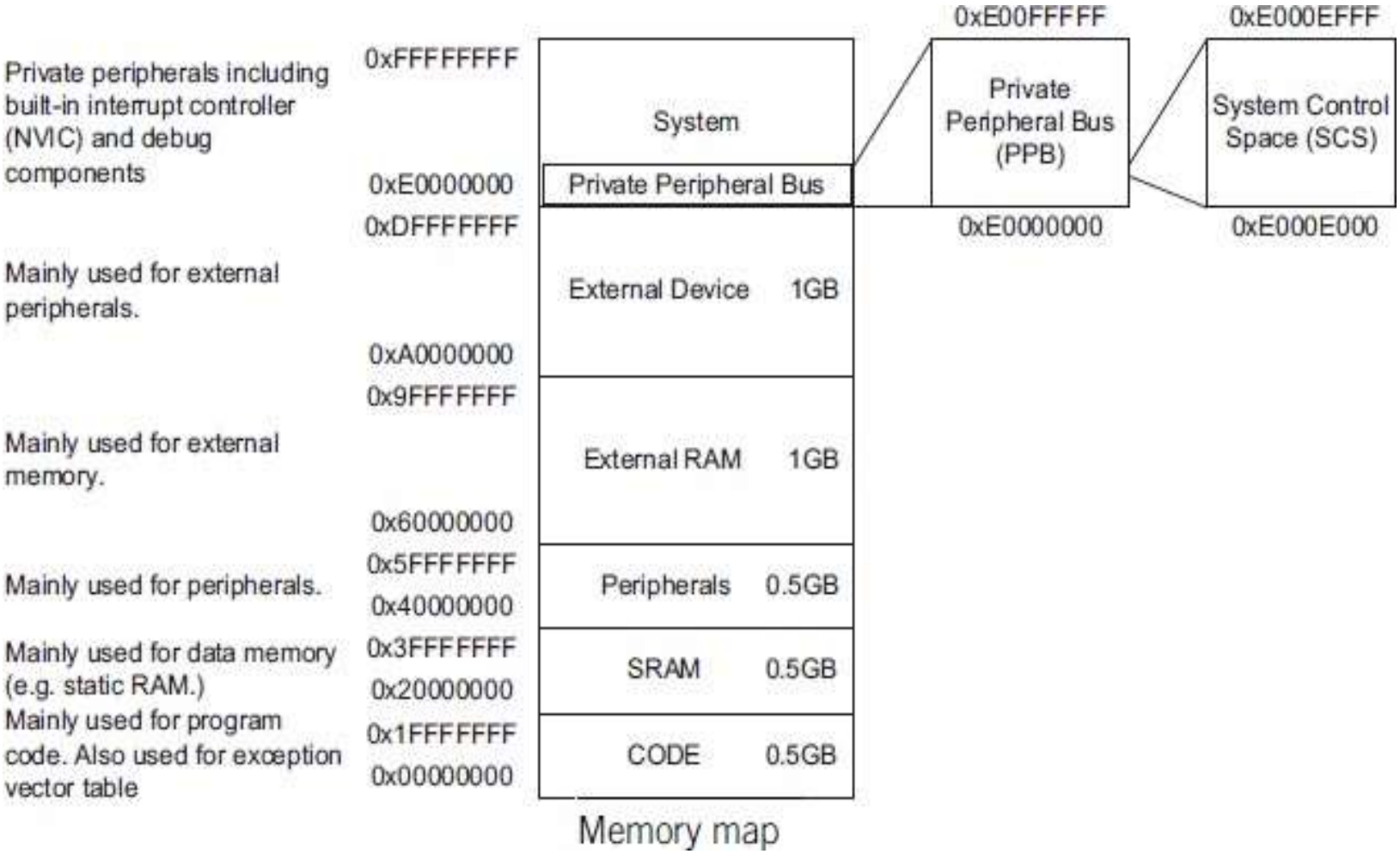
Memory system: Memory system features

- Support for little endian and big endian memory systems - The Cortex-M3 and Cortex-M4 processors can work with either little endian or big endian memory systems. In practice, a microcontroller product is normally designed with just one endian configuration.
- Bit band accesses (optional) - When the bit-band feature is included (determined by microcontroller/System-on-Chip vendors), two 1MB regions in the memory map are bit addressable via two bit-band regions. This allows atomic access to individual bits in SRAM or peripheral address space.

Memory map

- The 4GB address space of the Cortex-M processors is partitioned into a number of memory regions . The partitioning is based on typical usages so that different areas are designed to be used primarily for:
- Program code accesses (e.g., CODE region)
- Data accesses (e.g., SRAM region)
- Peripherals (e.g., Peripheral region)
- Processor's internal control and debug components (e.g., Private Peripheral Bus)
- The architecture also allows high flexibility to allow memory regions to be used for other purposes. For example, programs can be executed from the CODE as well as the SRAM region, and a microcontroller can also integrate SRAM blocks in CODE region.

Memory map



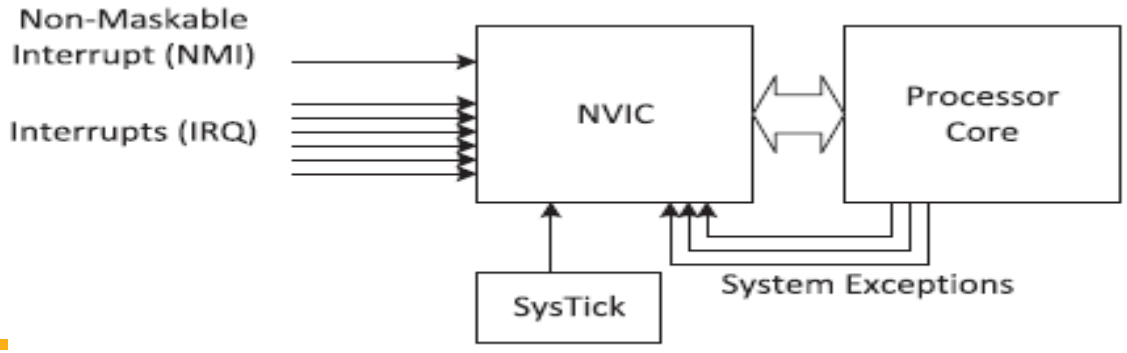
Memory map

Exceptions and interrupts

- In Cortex-M processors, there are a number of exception sources:
- Exceptions are processed by the NVIC. The NVIC can handle a number of Interrupt Requests (IRQs) and a Non-Maskable Interrupt (NMI) request.
- Usually IRQs are generated by on-chip peripherals or from external interrupt inputs though I/O ports.
- The NMI could be used by a watchdog timer or brownout detector (a voltage monitoring unit that warns the processor when the supply voltage drops below a certain level).
- Inside the processor there is also a timer called SysTick, which can generate a periodic timer interrupt request, which can be used by embedded OSs for timekeeping, or for simple timing control in applications that don't require an OS.

Exceptions and interrupts

- The processor itself is also a source of exception events.
- These could be fault events that indicate system error conditions, or exceptions generated by software to support embedded OS operations.
- As opposed to classic ARM processors such as the ARM7TDMI, there is no FIQ (Fast Interrupt) in the Cortex-M processor. However, the interrupt latency of Corex-M4 is very low, only 12 clock cycles, so this does not cause problems.
- Reset is a special kind of exception. When the processor exits from a reset, it executes the reset handler in Thread mode (rather than Handler mode as in other exceptions). Also the exception number in IPSR is read as zero.



Various exception sources

Exceptions and interrupts

Exception Types

Exception Number	CMSIS Interrupt Number	Exception Type	Priority	Function
1	—	Reset	–3 (Highest)	Reset
2	–14	NMI	–2	Non-Maskable interrupt
3	–13	HardFault	–1	All classes of fault, when the corresponding fault handler cannot be activated because it is currently disabled or masked by exception masking
4	–12	MemManage	Settable	Memory Management fault; caused by MPU violation or invalid accesses (such as an instruction fetch from a non-executable region)
5	–11	BusFault	Settable	Error response received from the bus system; caused by an instruction prefetch abort or data access error
6	–10	Usage fault	Settable	Usage fault; typical causes are invalid instructions or invalid state transition attempts (such as trying to switch to ARM state in the Cortex-M3)
7–10	—	—	—	Reserved
11	–5	SVC	Settable	Supervisor Call via SVC instruction
12	–4	Debug monitor	Settable	Debug monitor – for software based debug (often not used)
13	—	—	—	Reserved
14	–2	PendSV	Settable	Pendable request for System Service
15	–1	SYSTICK	Settable	System Tick Timer
16–255	0–239	IRQ	Settable	IRQ input #0–239

Nested vectored interrupt controller (NVIC)

- The NVIC is a part of the Cortex-M processor. It is programmable and its registers are located in the System Control Space (SCS) of the memory map.
- The NVIC handles the exceptions and interrupt configurations, prioritization, and interrupt masking.
- The NVIC has the following features:
  - Flexible exception and interrupt management
  - Nested exception/interrupt support
  - Vectored exception/interrupt entry
  - Interrupt masking

Nested vectored interrupt controller (NVIC)

- **Nested exception/interrupt support**
- Each exception has a priority level. Some exceptions, such as interrupts, have programmable priority levels and some others (e.g., NMI) have a fixed priority level.
- When an exception occurs, the NVIC will compare the priority level of this exception to the current level.
- If the new exception has a higher priority, the current running task will be suspended.
- Some of the registers will be stored on the stack memory, and the processor will start executing the exception handler of the new exception.
- This process is called “preemption.” When the higher priority exception handler is complete, it is terminated with an exception return operation and the processor automatically restores the registers from stack and resumes the task that was running previously.
- This mechanism allows nesting of exception services without any software overhead.

Vector table

Exception Type	CMSIS Interrupt Number	Address Offset	Vectors
18 - 255	2 - 239	0x48 – 0x3FF	IRQ #2 - #239 1
17	1	0x44	IRQ #1 1
16	0	0x40	IRQ #0 1
15	-1	0x3C	SysTick 1
14	-2	0x38	PendSV 1
NA	NA	0x34	Reserved
12	-4	0x30	Debug Monitor 1
11	-5	0x2C	SVC 1
NA	NA	0x28	Reserved
NA	NA	0x24	Reserved
NA	NA	0x20	Reserved
NA	NA	0x1C	Reserved
6	-10	0x18	Usage fault 1
4	-11	0x14	Bus Fault 1
4	-12	0x10	MemManage Fault 1
3	-13	0x0C	HardFault 1
2	-14	0x08	NMI 1
1	NA	0x04	Reset 1
NA	NA	0x00	Initial value of MPS

Exception types (LSB of exception vectors should be set to 1 to indicate Thumb state)

System control block (SCB)

- One part of the processor that is merged into the NVIC unit is the SCB.
- The SCB contains various registers for:
  - Controlling processor configurations (e.g., low power modes)
  - Providing fault status information (fault status registers)
  - Vector table relocation (VTOR)
- The SCB is memory-mapped. Similar to the NVIC registers, the SCB registers are accessible from the System Control Space (SCS).

Debug

- There are two types of interfaces provided in the Cortex-M processors: debug and trace.
- The debug interface allows a debug adaptor to connect to a Cortex-M microcontroller to control the debug features and access the memory space on the chip.
- The Cortex-M processor supports the traditional JTAG protocol, which uses either 4 or 5 pins, or a newer 2-pin protocol called Serial Wire Debug (SWD).
- The SWD protocol was developed by ARM, and can handle the same debug features as in JTAG in just two pins, without any loss of debug performance.
- The two protocols can use the same connector, with JTAG TCK shared with the Serial Wire clock, and JTAG TMS shared with the Serial Wire Data, which is bidirectional

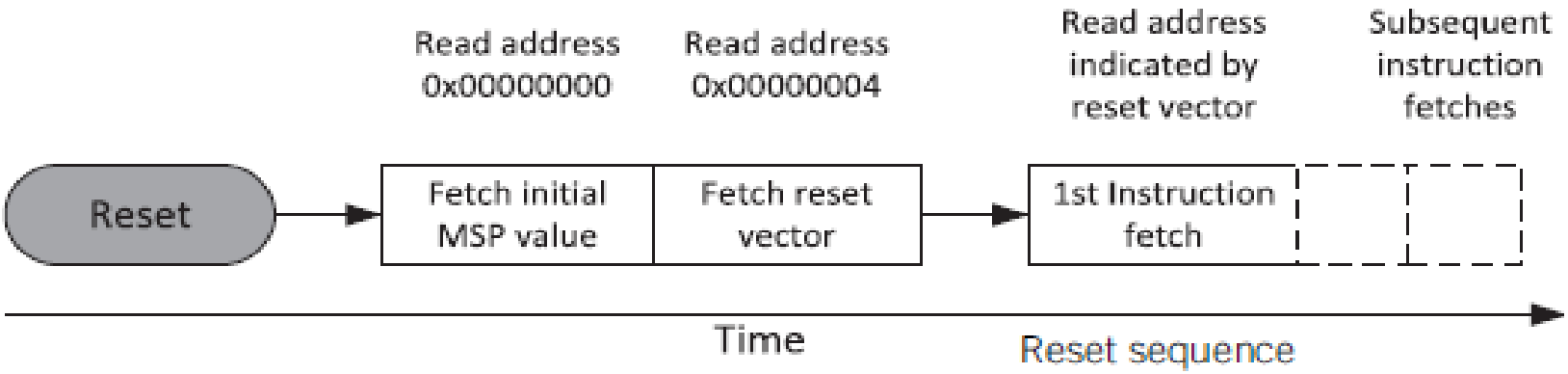
Debug

- The trace interface is used to collect information from the processor during runtime such as data, event, profiling information, or even complete details of program execution.
- Two types of trace interface are supported: a single pin protocol called Serial Wire Viewer (SWV) and a multi-pin protocol called Trace Port .
- SWV is a low-cost solution that has a lower trace data bandwidth limit.
- However, the bandwidth is still large enough to handle capturing of selective data trace, event trace, and basic profiling.
- The output signal, which is called Serial Wire Output (SWO), can be shared with the JTAG TDO pin so that you only need one standard JTAG/SWD connector for both debug and trace.
- The Trace Port mode requires one clock pin and several data pins

Reset and reset sequence

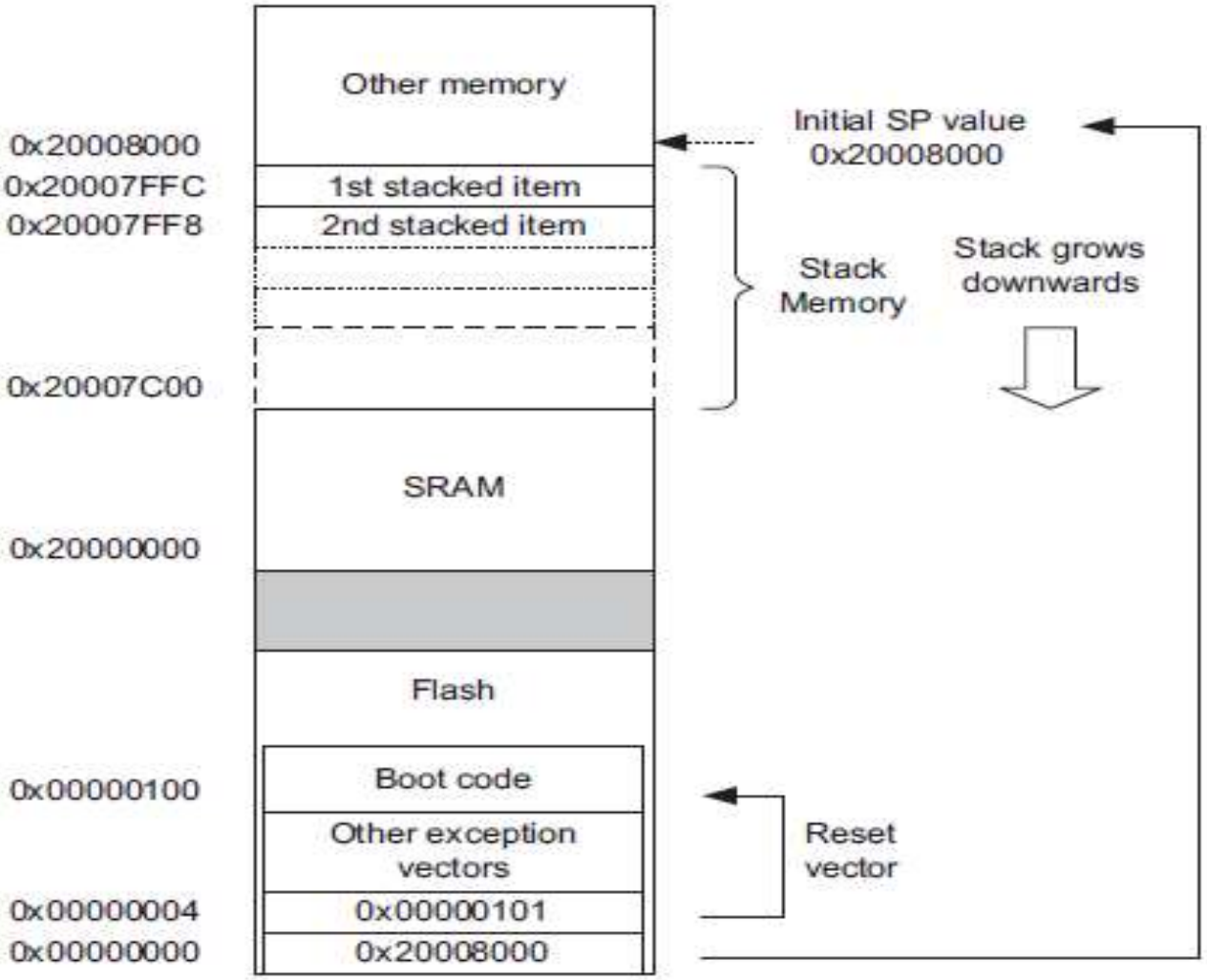
- In typical Cortex-M microcontrollers, there can be three types of reset:
- Power on reset - reset everything in the microcontroller. This includes the processor and its debug support component and peripherals.
- System reset - reset just the processor and peripherals, but not the debug support component of the processor.
- Processor reset - reset the processor only.
- The duration of Power on reset and System reset depends on the microcontroller design.
- In some cases the reset lasts a number of milli seconds as the reset controller needs to wait for a clock source such as a crystal oscillator to stabilize.

Reset and reset sequence



- The setup of the MSP is necessary because some exceptions such as the NMI or HardFault handler could potentially occur shortly after the reset, and the stack memory and hence the MSP will then be needed to push some of the processor status to the stack before exception handling.
- Because the stack operations in the Cortex-M3 or Cortex-M4 processors are based on full descending stack (SP decrement before store), the initial SP value should be set to the first memory after the top of the stack region. For example, if you have a stack memory range from 0x20007C00 to 0x20007FFF (1Kbytes), the initial stack value should be set to 0x20008000

Reset and reset sequence



Initial Stack Pointer value and Initial Program Counter value example

Introduction to STM32F40XX

- STM32F407xx family is based on the high-performance ARM® Cortex®-M4 32-bit RISC core operating at a frequency of up to 168 MHz. The Cortex-M4 core features a Floating point unit (FPU) single precision which supports all ARM single-precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU) which enhances application security.
- STM32F407xx family incorporates high-speed embedded memories (Flash memory up to 1 Mbyte, up to 192 Kbytes of SRAM), up to 4 Kbytes of backup SRAM, and an extensive range of enhanced I/Os and peripherals connected to two APB buses, three AHB buses and a 32-bit multi-AHB bus matrix.

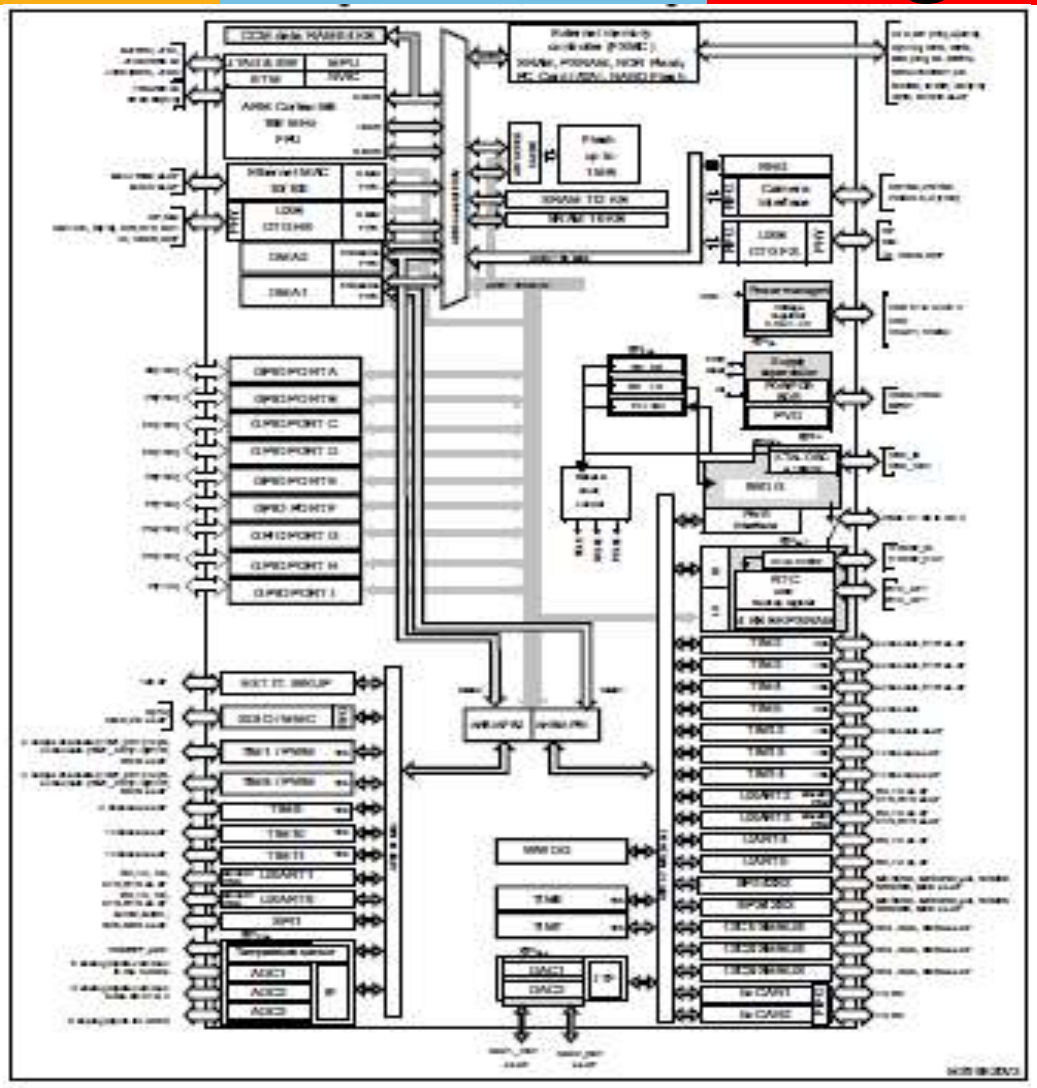
Introduction to STM32F40XX

- All devices offer three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers for motor control, two general-purpose 32-bit timers. a true random number generator (RNG). They also feature standard and advanced communication interfaces.
  - Up to three I2Cs
  - Three SPIs, two I2Ss full duplex. To achieve audio class accuracy, the I2S peripherals can be clocked via a dedicated internal audio PLL or via an external clock to allow synchronization.
  - Four USARTs plus two UARTs
  - An USB OTG full-speed and a USB OTG high-speed with full-speed capability (with the ULPI),
  - Two CANs
  - An SDIO/MMC interface
  - Ethernet and the camera interface

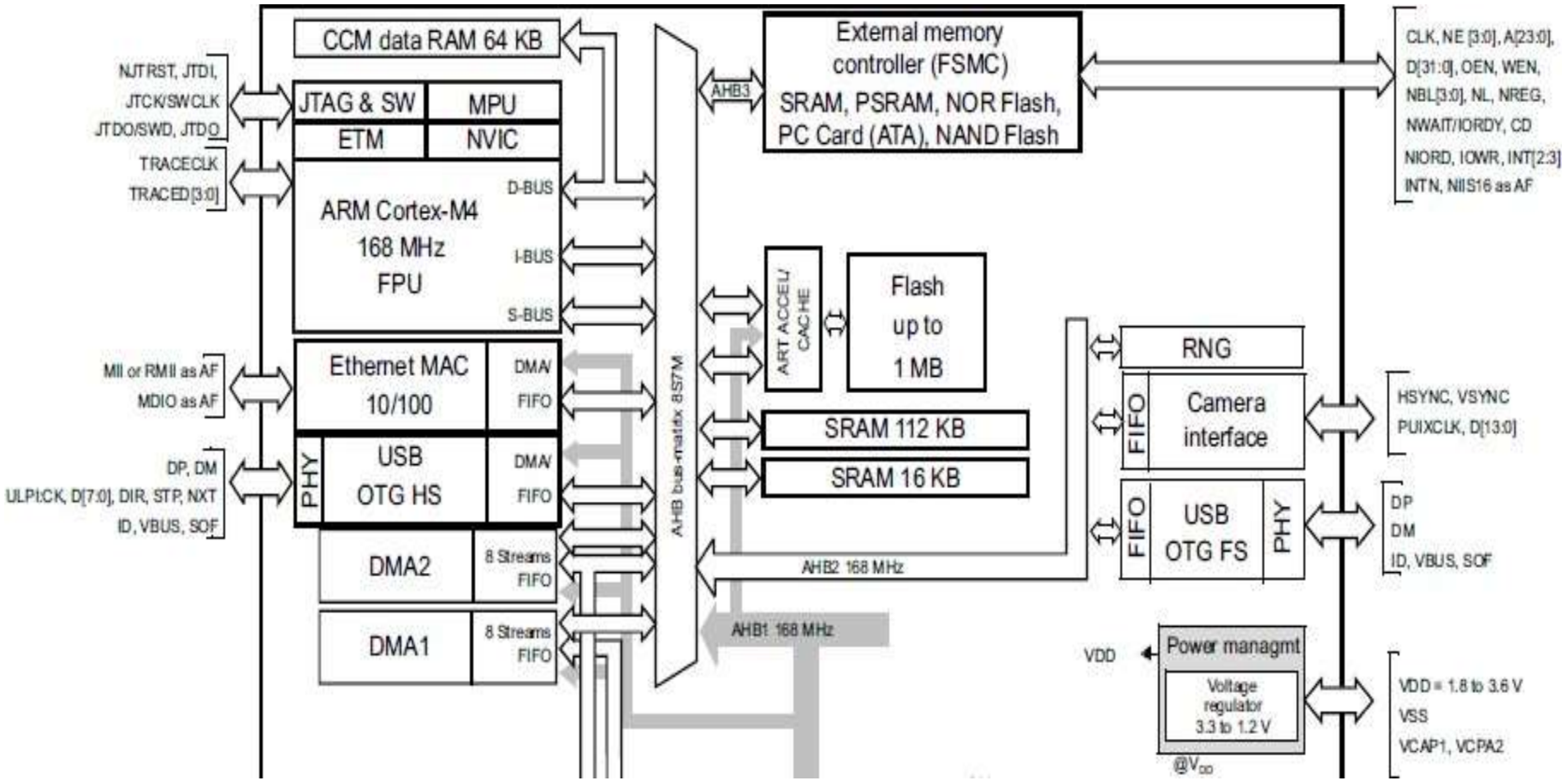
STM32F40XX Applications

- STM32F407xx microcontroller family suitable for a wide range of applications:
- Motor drive and application control
- Medical equipment
- Industrial applications: PLC, inverters, circuit breakers
- Printers, and scanners
- Alarm systems, video intercom, and HVAC
- Home audio appliances

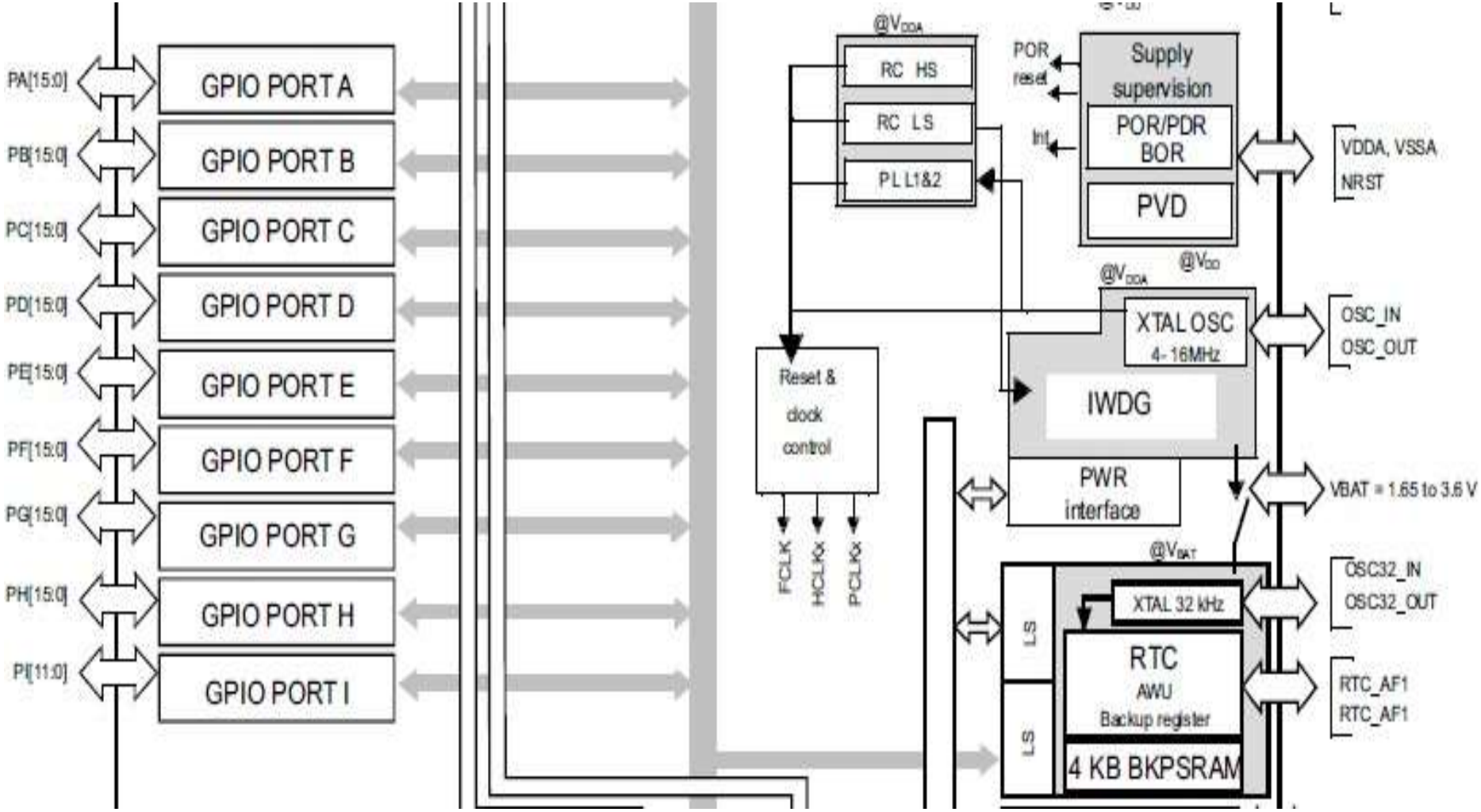
STM32F40XX Block diagram



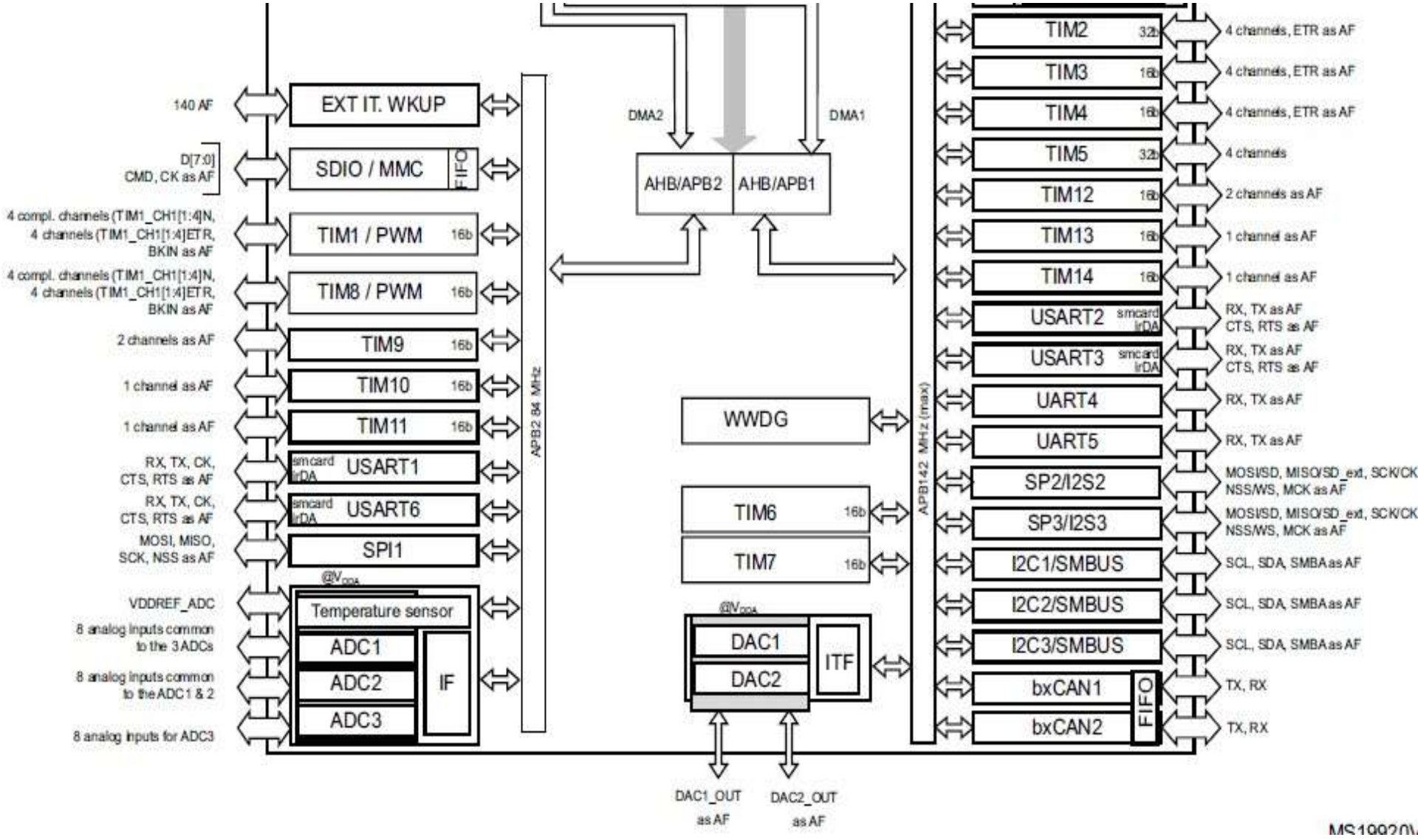
STM32F40XX Block diagram



STM32F40XX Block diagram



STM32F40XX Block diagram



Adaptive real-time memory accelerator (ART Accelerator™)

- The ART Accelerator™ is a memory accelerator which is optimized for STM32 industry-standard ARM® Cortex®-M4 with FPU processors. It balances the inherent performance advantage of the ARM Cortex-M4 with FPU over Flash memory technologies, which normally requires the processor to wait for the Flash memory at higher frequencies.
- To release the processor full 210 DMIPS performance at this frequency, the accelerator implements an instruction prefetch queue and branch cache, which increases program execution speed from the 128-bit Flash memory. Based on CoreMark benchmark, the performance achieved thanks to the ART accelerator is equivalent to 0 wait state program execution from Flash memory at a CPU frequency up to 168 MHz.

Memory protection unit

- The memory protection unit (MPU) is used to manage the CPU accesses to memory to prevent one task to accidentally corrupt the memory or resources used by any other active task. This memory area is organized into up to 8 protected areas that can in turn be divided up into 8 subareas. The protection area sizes are between 32 bytes and the whole 4 gigabytes of addressable memory.
- The MPU is especially helpful for applications where some critical or certified code has to be protected against the misbehavior of other tasks. It is usually managed by an RTOS (real-time operating system). If a program accesses a memory location that is prohibited by the MPU, the RTOS can detect it and take action. In an RTOS environment, the kernel can dynamically update the MPU area setting, based on the process to be executed.

Embedded Flash memory

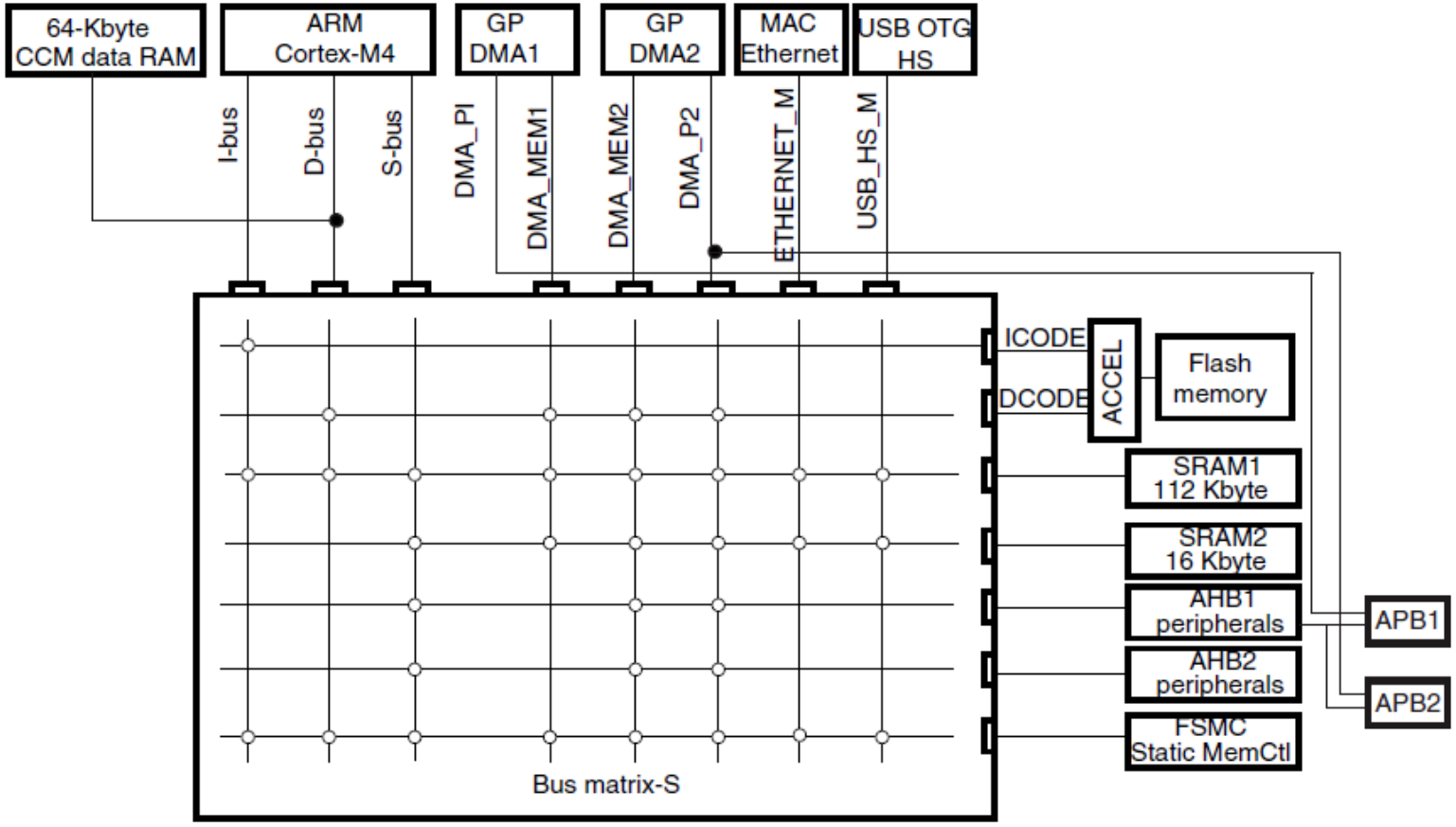
- The STM32F40xxx devices embed a Flash memory of 512 Kbytes or 1 Mbytes available for storing programs and data.
- **CRC (cyclic redundancy check) calculation unit**
- The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from a 32-bit data word and a fixed generator polynomial.
- Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the EN/IEC 60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a software signature during runtime, to be compared with a reference signature generated at link-time and stored at a given memory location.

Embedded SRAM

- All STM32F40xxx products embed:
- Up to 192 Kbytes of system SRAM including 64 Kbytes of CCM (core coupled memory) data RAM RAM memory is accessed (read/write) at CPU clock speed with 0 wait states.
- 4 Kbytes of backup SRAM This area is accessible only from the CPU. Its content is protected against possible unwanted write accesses, and is retained in Standby or VBAT mode.

Multi-AHB bus matrix

- The 32-bit multi-AHB bus matrix interconnects all the masters (CPU, DMAs, Ethernet, USB HS) and the slaves (Flash memory, RAM, FSMC, AHB and APB peripherals) and ensures a seamless and efficient operation even when several high-speed peripherals work simultaneously.



DMA controller (DMA)

- The devices feature two general-purpose dual-port DMAs (DMA1 and DMA2) with 8 streams each. They are able to manage memory-to-memory, peripheral-to-memory and memory-to-peripheral transfers. They feature dedicated FIFOs for APB/AHB peripherals, support burst transfer and are designed to provide the maximum peripheral bandwidth (AHB/APB).
- The two DMA controllers support circular buffer management, so that no specific code is needed when the controller reaches the end of the buffer. The two DMA controllers also have a double buffering feature, which automates the use and switching of two memory buffers without requiring any special code.

DMA controller (DMA)

- Each stream is connected to dedicated hardware DMA requests, with support for software trigger on each stream. Configuration is made by software and transfer sizes between source and destination are independent.
- The DMA can be used with the main peripherals:
  - SPI and I2S
  - I2C
  - USART
  - General-purpose, basic and advanced-control timers TIMx • DAC
  - SDIO
  - Camera interface (DCMI)
  - ADC

Flexible static memory controller (FSMC)

- The FSMC is embedded in the STM32F405xx and STM32F407xx family. It has four Chip Select outputs supporting the following modes: PCCard/Compact Flash, SRAM, PSRAM, NOR Flash and NAND Flash.
- Functionality overview:
- Write FIFO
- Maximum FSMC_CLK frequency for synchronous accesses is 60 MHz.

LCD parallel interface

- The FSMC can be configured to interface seamlessly with most graphic LCD controllers. It supports the Intel 8080 and Motorola 6800 modes, and is flexible enough to adapt to specific LCD interfaces. This LCD parallel interface capability makes it easy to build cost-effective graphic applications using LCD modules with embedded controllers or high performance solutions using external controllers with dedicated acceleration.

Nested vectored interrupt controller (NVIC)

- The STM32F407xx embed a nested vectored interrupt controller able to manage 16 priority levels, and handle up to 82 maskable interrupt channels plus the 16 interrupt lines of the Cortex®-M4 with FPU core.
- This hardware block provides flexible interrupt management features with minimum interrupt latency.
- Closely coupled NVIC gives low-latency interrupt processing
- Interrupt entry vector table address passed directly to the core
- Allows early processing of interrupts
- Processing of late arriving, higher-priority interrupts
- Support tail chaining
- Processor state automatically saved
- Interrupt entry restored on interrupt exit with no instruction overhead

This hardware block provides flexible interrupt management features with minimum interrupt latency.

External interrupt/event controller (EXTI)

- The external interrupt/event controller consists of 23 edge-detector lines used to generate interrupt/event requests.
- Each line can be independently configured to select the trigger event (rising edge, falling edge, both) and can be masked independently.
- A pending register maintains the status of the interrupt requests.
- The EXTI can detect an external line with a pulse width shorter than the Internal APB2 clock period.
- Up to 140 GPIOs can be connected to the 16 external interrupt lines.

Clocks and startup

- On reset the 16 MHz internal RC oscillator is selected as the default CPU clock. The 16 MHz internal RC oscillator is factory-trimmed to offer 1% accuracy over the full temperature range.
- The application can then select as system clock either the RC oscillator or an external 4-26 MHz clock source.
- This clock can be monitored for failure.
- If a failure is detected, the system automatically switches back to the internal RC oscillator and a software interrupt is generated (if enabled).
- This clock source is input to a PLL thus allowing to increase the frequency up to 168 MHz. Similarly, full interrupt management of the PLL clock entry is available when necessary (for example if an indirectly used external oscillator fails).

Clocks and startup

- Several prescalers allow the configuration of the three AHB buses, the high-speed APB (APB2) and the low-speed APB (APB1) domains.
- The maximum frequency of the three AHB buses is 168 MHz while the maximum frequency of the high-speed APB domains is 84 MHz.
- The maximum allowed frequency of the low-speed APB domain is 42 MHz.
- The devices embed a dedicated PLL (PLLI2S) which allows to achieve audio class performance.
- In this case, the I2S master clock can generate all standard sampling frequencies from 8 kHz to 192 kHz.

Boot modes

- At startup, boot pins are used to select one out of three boot options:
  - Boot from user Flash
  - Boot from system memory
  - Boot from embedded SRAM
- The boot loader is located in system memory. It is used to reprogram the Flash memory by using USART1 (PA9/PA10), USART3 (PC10/PC11 or PB10/PB11), CAN2 (PB5/PB13), USB OTG FS in Device mode (PA11/PA12) through DFU (device firmware upgrade).

Power supply schemes

- VDD = 1.8 to 3.6 V: external power supply for I/Os and the internal regulator (when enabled), provided externally through VDD pins.
- VSSA, VDDA = 1.8 to 3.6 V: external analog power supplies for ADC, DAC, Reset blocks, RCs and PLL. VDDA and VSSA must be connected to VDD and VSS, respectively.
- VBAT = 1.65 to 3.6 V: power supply for RTC, external clock 32 kHz oscillator and backup registers (through power switch) when VDD is not present.

Real-time clock (RTC), backup SRAM and backup registers

- The backup domain of the STM32F407xx includes: •
  - The real-time clock (RTC)
  - 4 Kbytes of backup SRAM
  - 20 backup registers
- The real-time clock (RTC) is an independent BCD timer/counter. Dedicated registers contain the second, minute, hour (in 12/24 hour), week day, date, month, year, in BCD (binary-coded decimal) format. Correction for 28, 29 (leap year), 30, and 31 day of the month are performed automatically. The RTC provides a programmable alarm and programmable periodic interrupts with wakeup from Stop and Standby modes. The sub-seconds value is also available in binary format.
- It is clocked by a 32.768 kHz external crystal, resonator or oscillator, the internal low-power RC oscillator or the high-speed external clock divided by 128.

Low-power modes

- **Sleep mode** In Sleep mode, only the CPU is stopped. All peripherals continue to operate and can wake up the CPU when an interrupt/event occurs.
- **Stop mode** The Stop mode achieves the lowest power consumption while retaining the contents of SRAM and registers. All clocks in the V12 domain are stopped, the PLL, the HSI RC and the HSE crystal oscillators are disabled. The voltage regulator can also be put either in normal or in low-power mode. The device can be woken up from the Stop mode by any of the EXTI line (the EXTI line source can be one of the 16 external lines, the PVD output, the RTC alarm / wakeup / tamper / time stamp events, the USB OTG FS/HS wakeup or the Ethernet wakeup).

Low-power modes

- **Standby mode** The Standby mode is used to achieve the lowest power consumption. The internal voltage regulator is switched off so that the entire V_{12} domain is powered off. The PLL, the HSI RC and the HSE crystal oscillators are also switched off. After entering Standby mode, the SRAM and register contents are lost except for registers in the backup domain and the backup SRAM when selected. The device exits the Standby mode when an external reset (NRST pin), an IWDG reset, a rising edge on the WKUP pin, or an RTC alarm / wakeup / tamper /time stamp event occurs. The standby mode is not supported when the embedded voltage regulator is bypassed and the V_{12} domain is controlled by an external power.

VBAT operation

- The VBAT pin allows to power the device VBAT domain from an external battery, an external supercapacitor, or from VDD when no external battery and an external supercapacitor are present.
- VBAT operation is activated when VDD is not present.
- The VBAT pin supplies the RTC, the backup registers and the backup SRAM.
- *Note: When the microcontroller is supplied from VBAT, external interrupts and RTC alarm/events do not exit it from VBAT operation.*
- *When PDR_ON pin is not connected to VDD (internal reset OFF), the VBAT functionality is no more available and VBAT pin should be connected to VDD.*

More features

- **Timers and watchdogs**
- STM32F407xx devices include two advanced-control timers, eight general-purpose timers, two basic timers and two watchdog timers.
- **Inter-integrated circuit interface (I^2C)**
- Up to three I^2C bus interfaces can operate in multimaster and slave modes. They can support the Standard-mode (up to 100 kHz) and Fast-mode (up to 400 kHz).
- **Universal synchronous/asynchronous receiver transmitters (USART)**
- The STM32F405xx and STM32F407xx embed four universal synchronous/asynchronous receiver transmitters (USART1, USART2, USART3 and USART6) and two universal asynchronous receiver transmitters (UART4 and UART5).

More features

- **Serial peripheral interface (SPI)**
- The STM32F40xxx feature up to three SPIs in slave and master modes in full-duplex and simplex communication modes. SPI1 can communicate at up to 42 Mbits/s, SPI2 and SPI3 can communicate at up to 21 Mbit/s.
- **Inter-integrated sound (I2S)**
- Two standard I2S interfaces (multiplexed with SPI2 and SPI3) are available. They can be operated in master or slave mode, in full duplex and half-duplex communication modes, and can be configured to operate with a 16-/32-bit resolution as an input or output channel.
- **Audio PLL (PLLI2S)**
- The devices feature an additional dedicated PLL for audio I2S application. It allows to achieve error-free I2S sampling clock accuracy without compromising on the CPU performance, while using USB peripherals.

More features

- **Secure digital input/output interface (SDIO)**
- An SD/SDIO/MMC host interface is available, that supports MultiMediaCard System Specification Version 4.2 in three different databus modes: 1-bit (default), 4-bit and 8-bit.
- The interface allows data transfer at up to 48 MHz, and is compliant with the SD Memory Card Specification Version 2.0.
- **Ethernet MAC interface with dedicated DMA and IEEE 1588 support**
- The STM32F407xx devices provide an IEEE-802.3-2002-compliant media access controller (MAC) for ethernet LAN communications through an industry-standard medium-independent interface (MII) or a reduced medium-independent interface (RMII). The STM32F407xx requires an external physical interface device (PHY) to connect to the physical LAN bus (twisted-pair, fiber, etc.). the PHY is connected to the STM32F407xx MII port using 17 signals for MII or 9 signals for RMII, and can be clocked using the 25 MHz (MII) from the STM32F407xx.

More features

- **Controller area network (bxCAN)**
- The two CANs are compliant with the 2.0A and B (active) specifications with a bitrate up to 1 Mbit/s. They can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers.
- **Universal serial bus on-the-go full-speed (OTG_FS)**
- The STM32F407xx embed an USB OTG full-speed device/host/OTG peripheral with integrated transceivers. The USB OTG FS peripheral is compliant with the USB 2.0 specification and with the OTG 1.0 specification. It has software-configurable endpoint setting and supports suspend/resume. The USB OTG full-speed controller requires a dedicated 48 MHz clock that is generated by a PLL connected to the HSE oscillator.

More features

- Digital camera interface (DCMI)
- Random number generator (RNG)
- General-purpose input/outputs (GPIOs)
- Analog-to-digital converters (ADCs)
- Temperature sensor
- Digital-to-analog converter (DAC)
- Serial wire JTAG debug port (SWJ-DP)
- Embedded Trace Macrocell™

Thank You…