ARM 프로세서 개요

Hancheol Cho

ARM (Advanced RISC Machine)

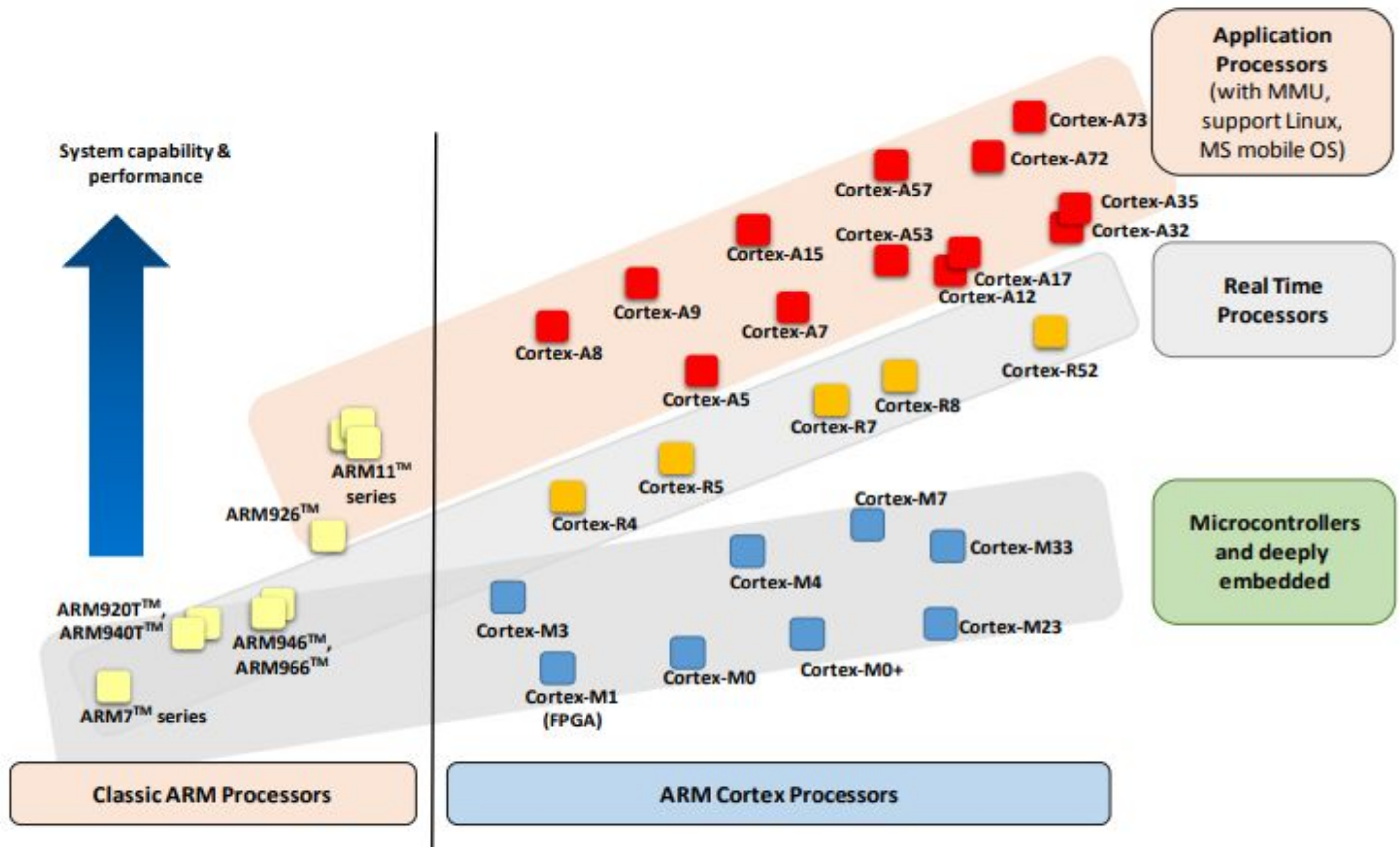
The ARM logo

Designer	ARM Holdings
Bits	32-bit, 64-bit
Introduced	1985; 32 years ago
Design	RISC
Type	Register-Register
Branching	Condition code, compare and branch
Open	Proprietary

ARM, originally **Acorn RISC Machine**, later **Advanced RISC Machine**, is a family of reduced instruction set computing (RISC) architectures for computer processors, configured for various environments. **British company ARM Holdings develops the architecture and licenses it to other companies**, who design their own products that implement one of those architectures

출처

ARM Processor Family



출처 : arm.com

ARM Architecture 분류

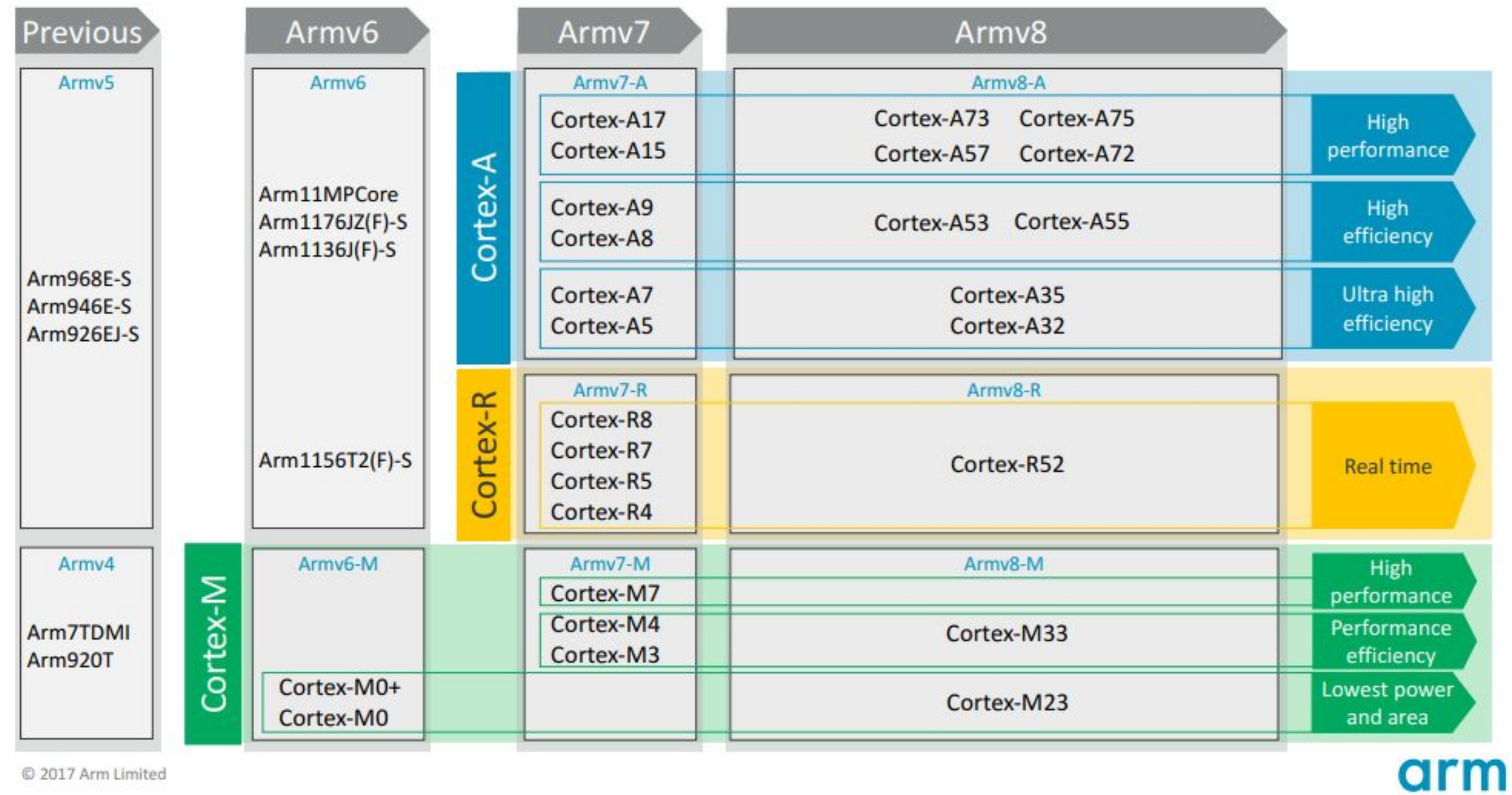
A The *Application* profile defines a VMSA based microprocessor architecture. It is targeted at high performance processors, capable of running full feature operating systems. It supports the ARM and Thumb instruction sets.

R The *Real-time* profile defines a PMSA based microprocessor architecture. It is targeted at systems that require deterministic timing and low interrupt latency. It supports the ARM and Thumb instruction sets.

M The *Microcontroller* profile provides low-latency interrupt processing accessible directly from high-level programming languages. It has a different exception handling model to the other profiles, implements a variant of the PMSA, and supports a variant of the Thumb instruction set only.

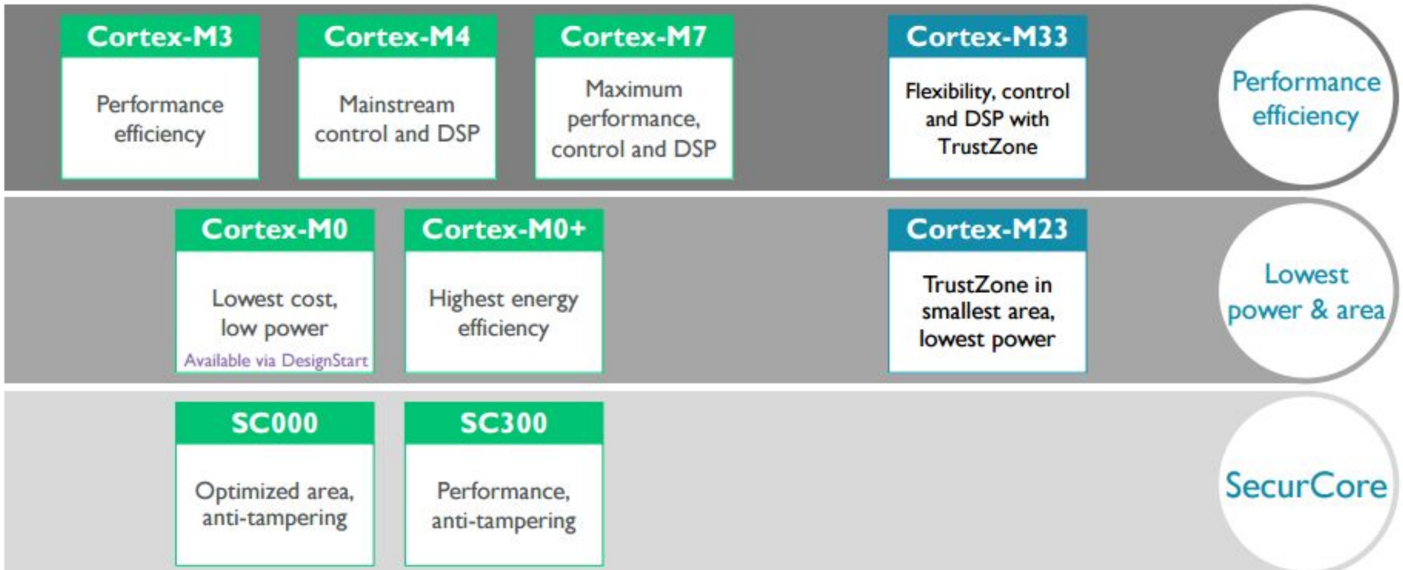
Profile	Architecture	Instruction Set	Processor
A-Profile	ARMv7-A	A32, T32	Cortex-A Series
R-Profile	ARMv7-R	A32, T32	Cortex-R Series
M-Profile	ARMv7-M	T32	Cortex-M Series
	ARMV6-M	T32	Cortex-M0 Series

ARM Processor



출처 : arm.com

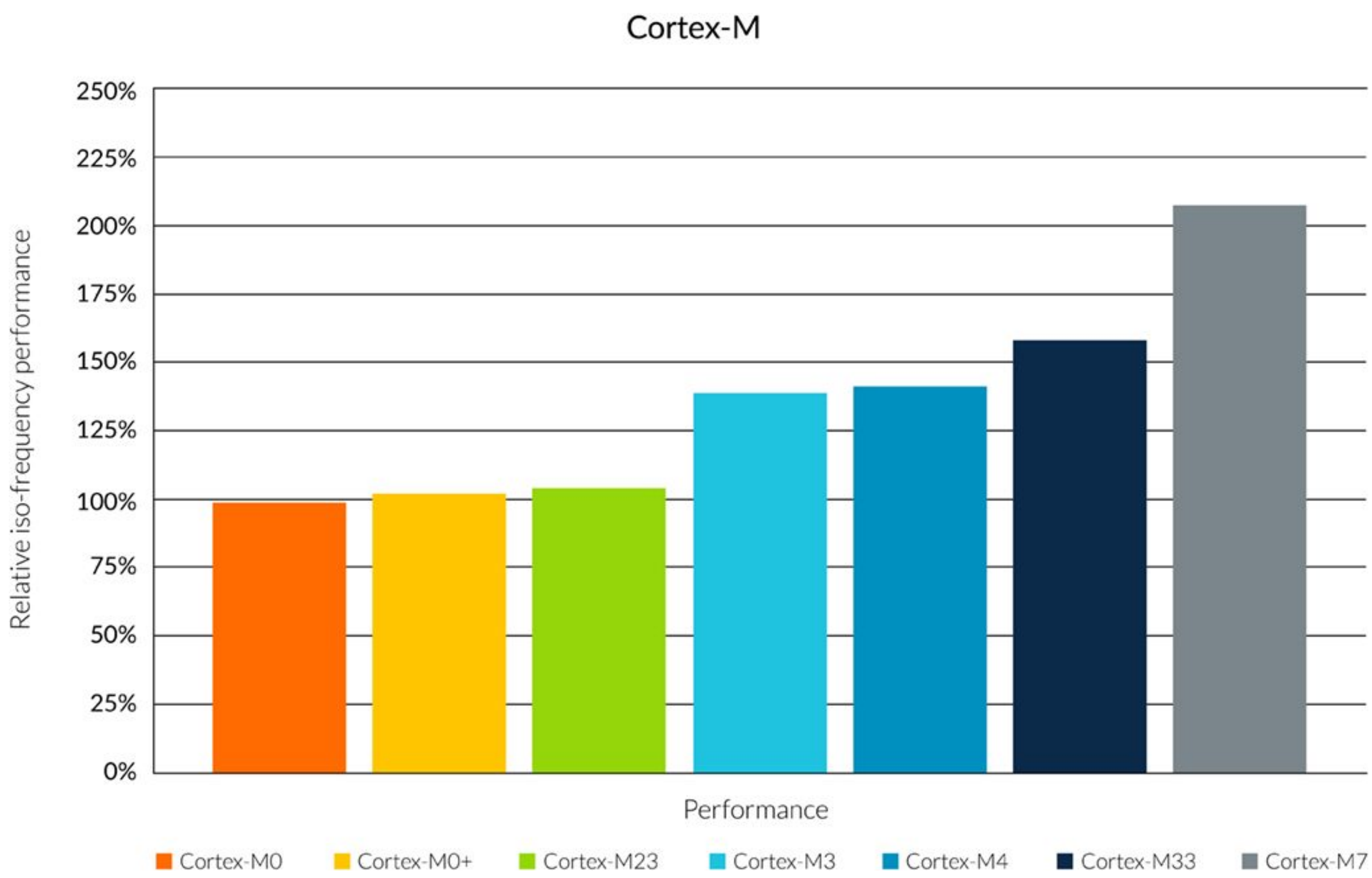
Cortex-M Processor



	Cortex-M0	Cortex-M0+	Cortex-M3	Cortex-M4	Cortex-M7
Instruction set architecture	ARMv6-M Thumb, Thumb-2	ARMv6-M Thumb, Thumb-2	ARMv7-M Thumb, Thumb-2	ARMv7-M Thumb, Thumb-2, DSP, FP (SP)	ARMv7-M Thumb, Thumb-2, DSP, FP (1. SP or 2. SP+DP)

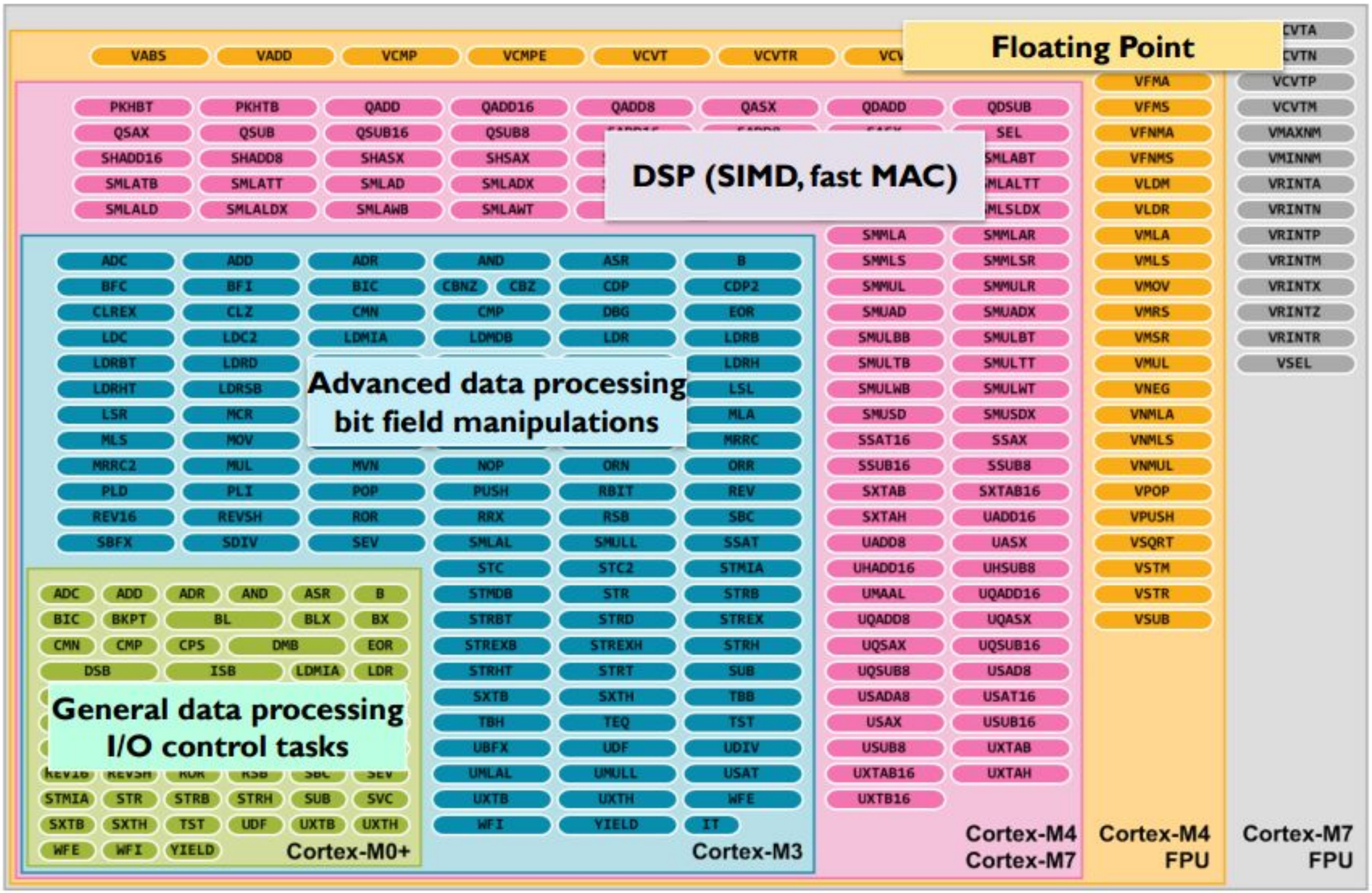
출처 : arm.com

Cortex-M Performance



출처 : arm.com

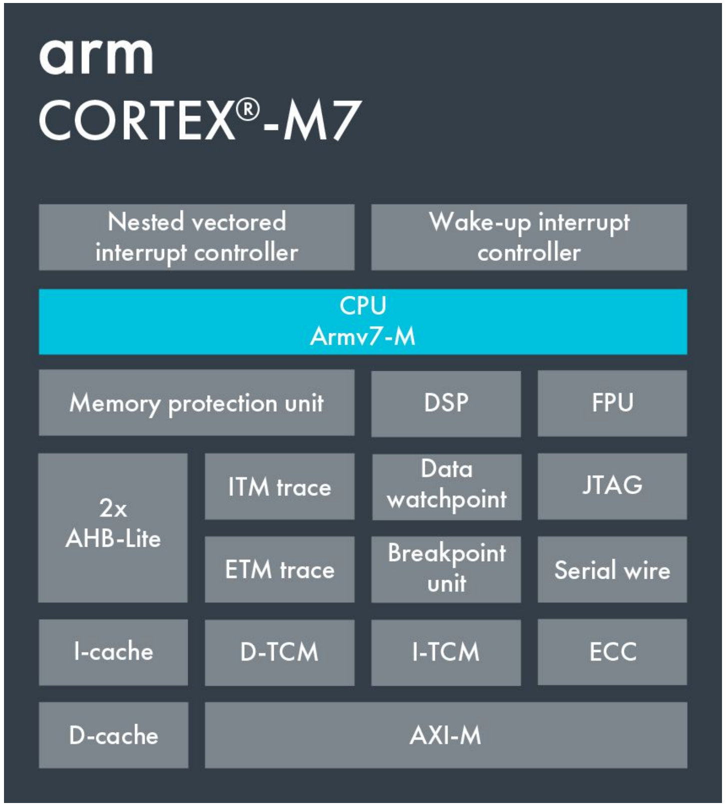
Cortex-M Instruction Set support



출처 : arm.com

Cortex-M7

- Cortex-M7은 캐시메모리가 있는것이 특징



Architecture	Harvard
ISA Support	Armv7-M
Pipeline	6-stage superscalar + branch prediction
DSP Extensions	Single cycle 16/32-bit MAC Single cycle dual 16-bit MAC 8/16-bit SIMD arithmetic Hardware Divide (2-12 Cycles)
Floating-Point Unit	Optional single and double precision floating point unit IEEE 754 compliant
Interconnect	64-bit AMBA4 AXI, AHB peripheral port
Instruction cache	0 to 64 kB, 2-way associative with optional ECC
Data cache	0 to 64 kB, 4-way associative with optional ECC
Instruction TCM	0 to 16 MB with optional ECC
Data TCM	0 to 16 MB with optional ECC
Memory Protection	Optional 8 or 16 region MPU with sub regions and background region
Interrupts	Non-maskable Interrupt (NMI) + 1 to 240 physical interrupts
Interrupt Priority Levels	8 to 256 priority levels
Wake-up Interrupt Controller	Up to 240 Wake-up Interrupts
Sleep Modes	Integrated WFI and WFE Instructions and Sleep On Exit capability. Sleep & Deep Sleep Signals. Optional Retention Mode with Arm Power Management Kit
Debug	Integrated Instructions
Debug	Optional JTAG and Serial Wire Debug ports. Up to 8 Breakpoints and 4 Watchpoints.
Trace	Optional Instruction Trace (ETM), Micro Trace Buffer (MTB), Data Trace (DWT), and Instrumentation Trace (ITM)

CMSIS

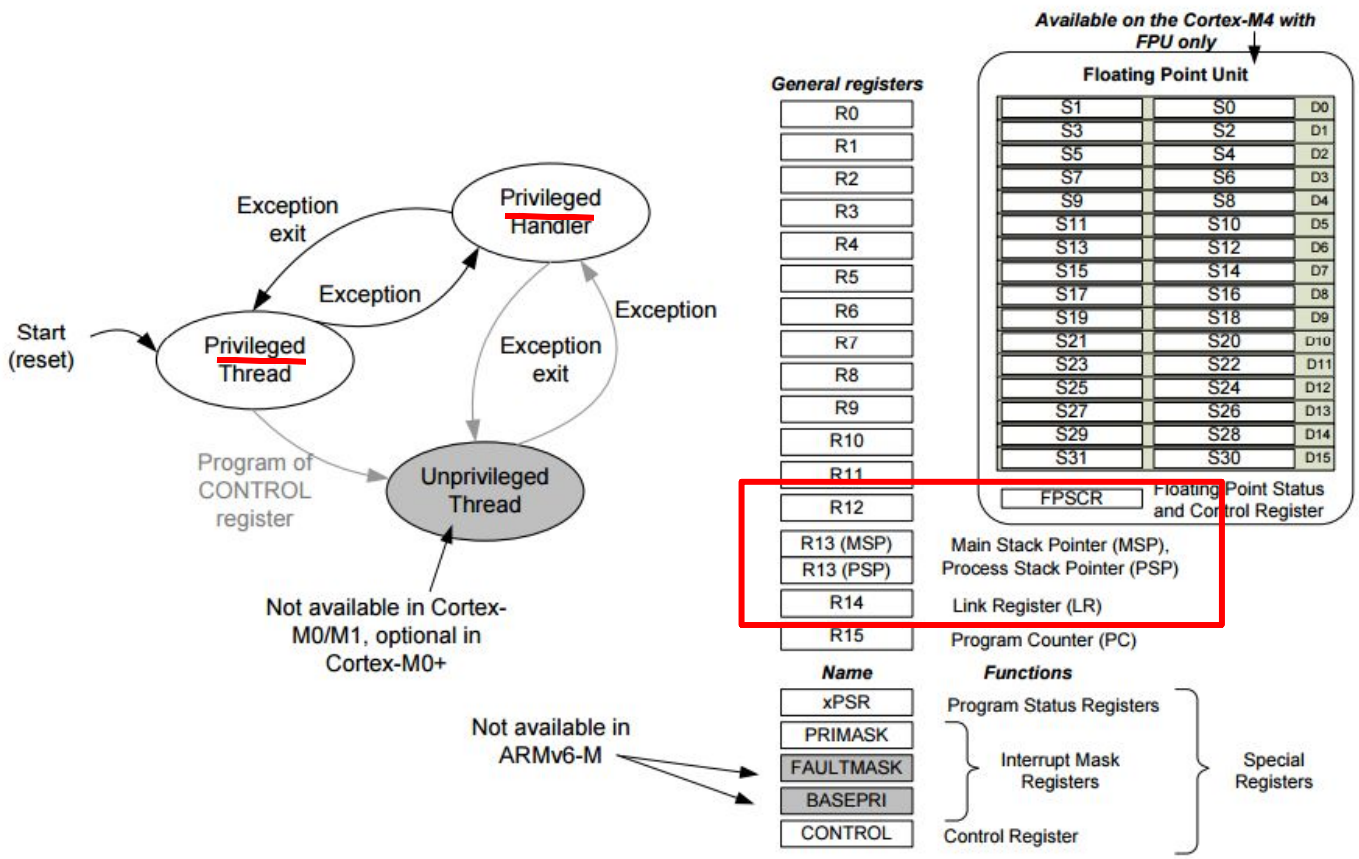
7.1 Why Cortex-M processors are easy to use

Although the Cortex-M processors are packed with features, they are also very easy to use. For instance, almost everything can be programmed in high-level language like C. Although there is a big variety of different Cortex-M processor-based products (e.g. with different memory size, peripherals, performance, packages, etc), the consistency of the architecture make it easy to start using a new Cortex-M processor once you have experience with one of them.

To make software development easier, and to enable better software reusability and portability, ARM developed the CMSIS-CORE, where CMSIS stands for Cortex Microcontroller Software Interface Standard. The CMSIS-CORE provides a standardized Hardware Abstraction Layer (HAL) for various features in the processors such as interrupt management control, using a set of APIs. The CMSIS-CORE is integrated in the device driver libraries from various microcontroller vendors, and is supported by various compilation suites.

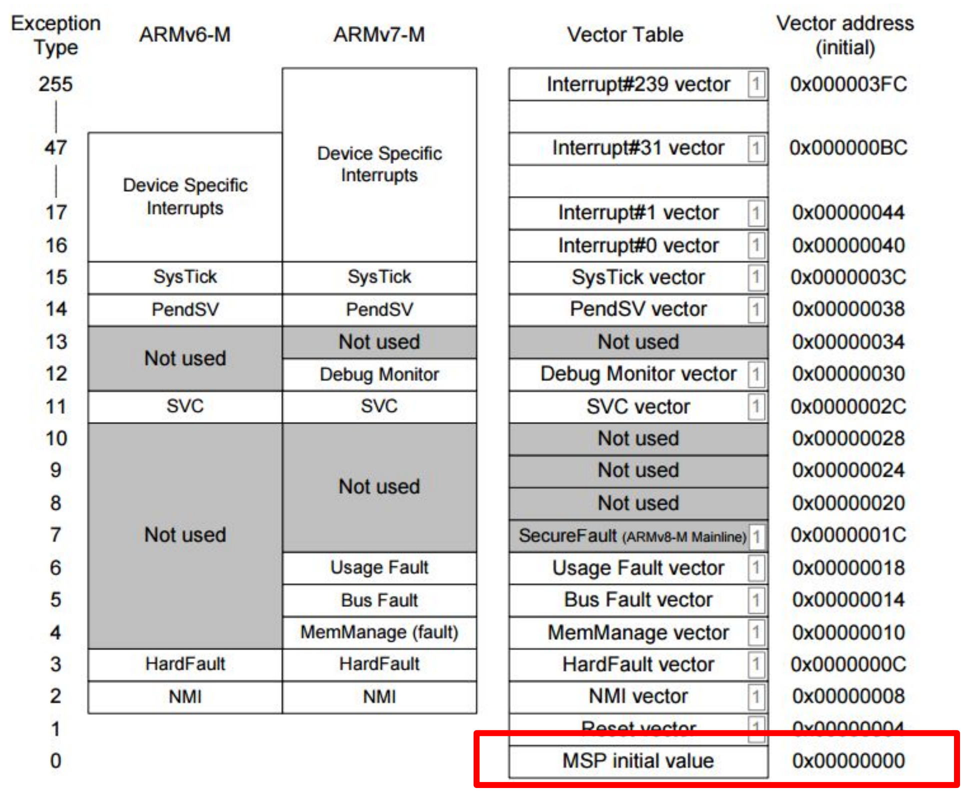
Beside from CMSIS-CORE, CMSIS also have a DSP software library (CMSIS-DSP). It provides various DSP functions and is optimized for Cortex-M4 and Cortex-M7 processors, and also supports other Cortex-M processors. Both CMSIS-CORE and CMSIS-DSP are free and can be downloaded from the GitHub (CMSIS 4, CMSIS 5), and are supported by multiple tool vendors.

Programmer's model



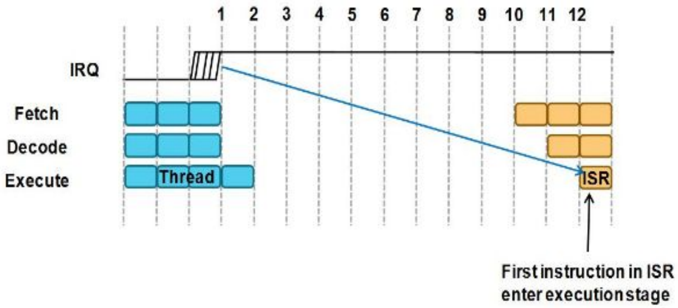
Exception Vector

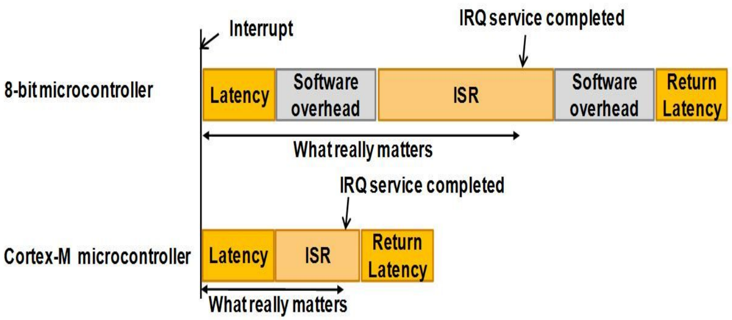
- 인터럽트 발생시 실행되는 함수 위치 저장
- 스택포인터 초기값을 지정 가능
    - 스타트업 코드도 C언어로 작성 가능



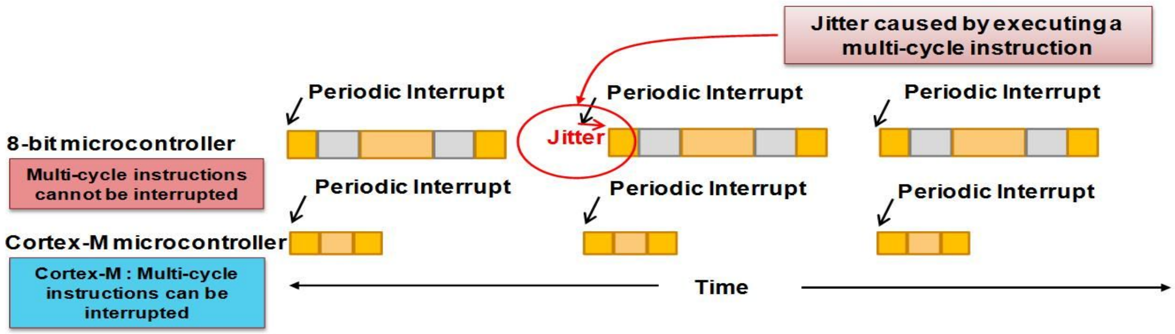
Interrupt Latency

- Interrupt Latency ?
  - 인터럽트 발생 후 실제 인터럽트 함수 실행시까지 걸리는 시간





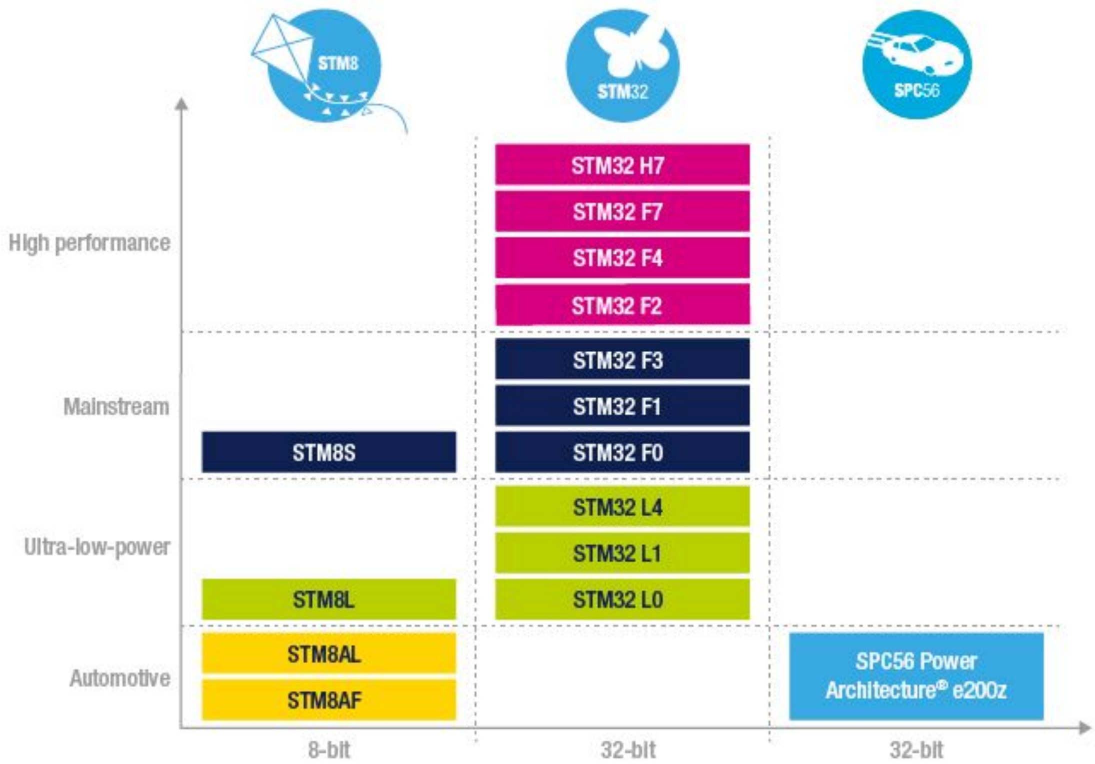
- 기존 마이크로 컨트롤러와 비교



출처 : [Link]

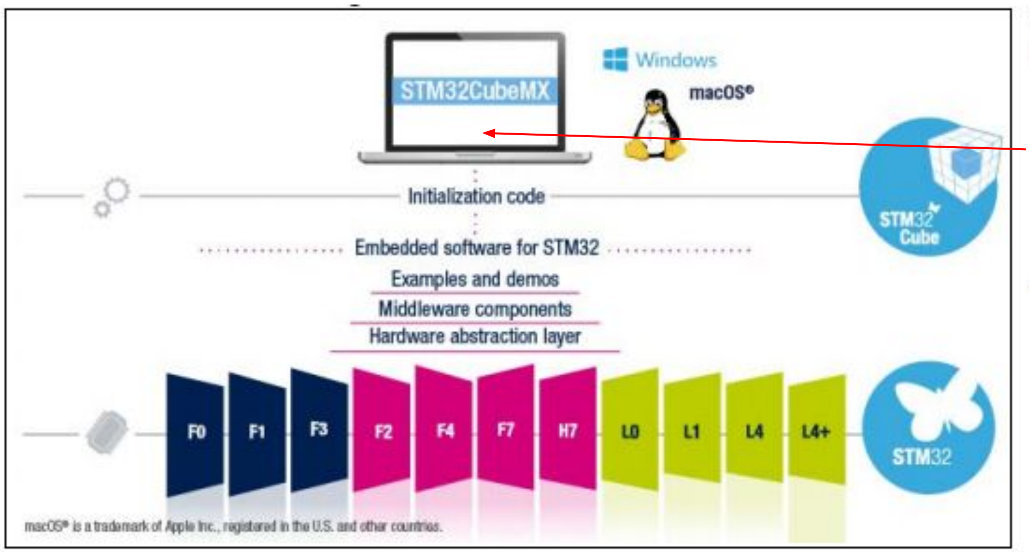
ST사의 MCU 구성

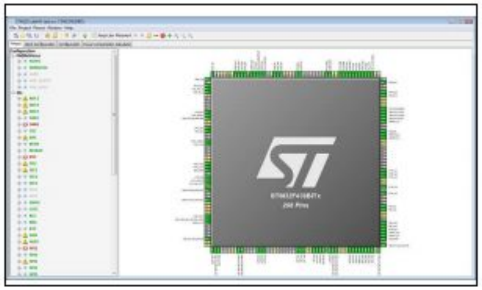
- 다양한 Line-up



STM32CubeMX, STM32Cube

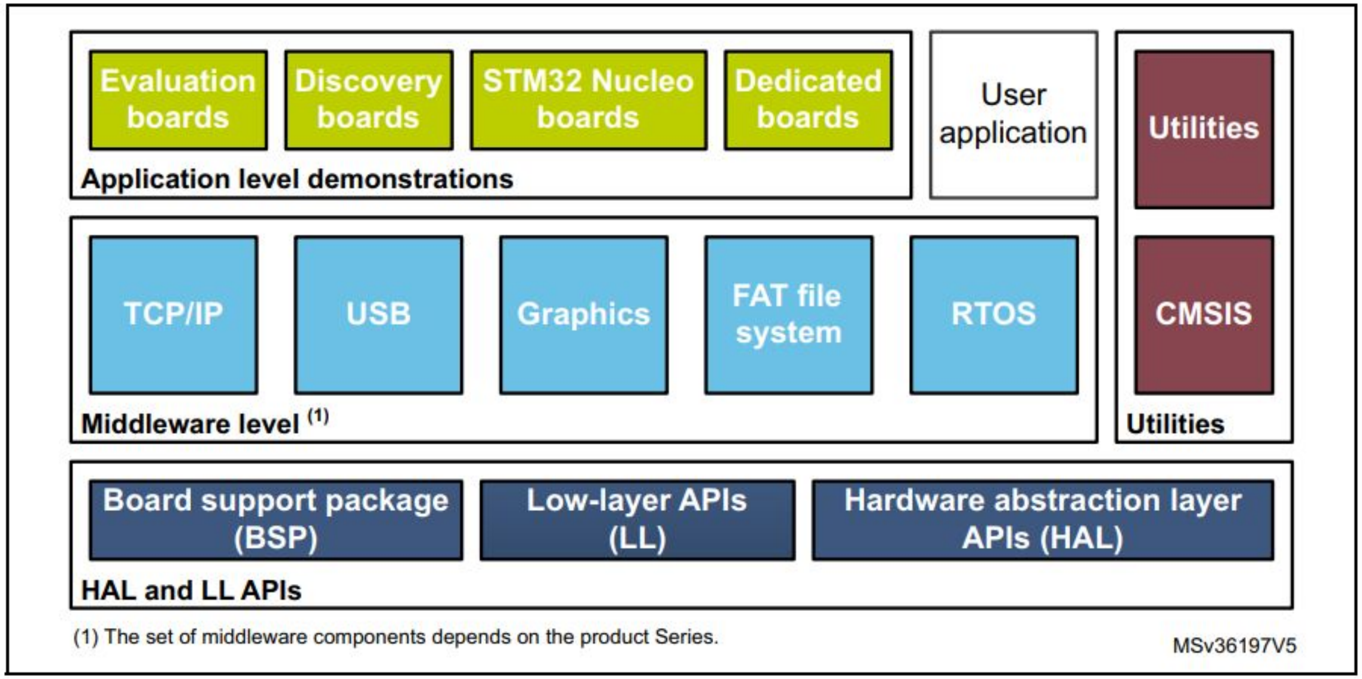
- STM32Cube 기반의 초기화 및 기능 함수 자동 생성





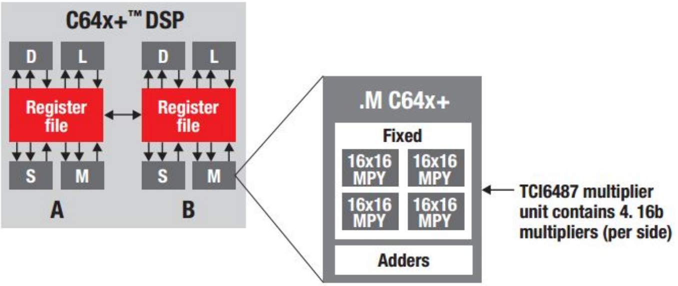
STM32Cube

- HAL 을 포함하여 다양한 하드웨어를 통일된 인터페이스로 사용 가능하도록 함.
- HAL 라이브러리가 무겁기 때문에 속도가 필요할때는 LL 라이브러리 사용 필요



DSP ?

- DSP 장점
    - 연산모듈이 8개 (최대 1Clock에 8개의 명령어 실행 가능)
    - Very-Long-Instruction-Word (VLIW)
    - 데이터 버스 최대 256bit



    - 소프트 파이프라인을 통한 병렬 실행

```
   MVKL L1DCC, A0     ; \
|| MVKL L1PCC, B0     ;  |__ Generate L1DCC pointer in A0
   MVKH L1DCC, A0     ;  |   and L1PCC pointer in B0
|| MVKH L1PCC, B0     ; /
|| MVK 1b, A1         ; \___ OPER encoding for 'freeze'
|| MVK 1b, B1         ; /    in both A1 and B1.
   STW A1, *A0        ; Write to L1DCC.OPER
|| STW B1, *B0        ; Write to L1PCC.OPER
   LDW *A0, A1        ; Get old freeze state into A1 from L1DCC
|| LDW *B0, B1        ; Get old freeze state into B1 from L1PCC
   NOP 4
  ; At this point, L1D and L1P are frozen.
  ; The old value of L1DCC.OPER is in bit 16 of A1.
  ; The old value of L1PCC.OPER is in bit 16 of B1.
```

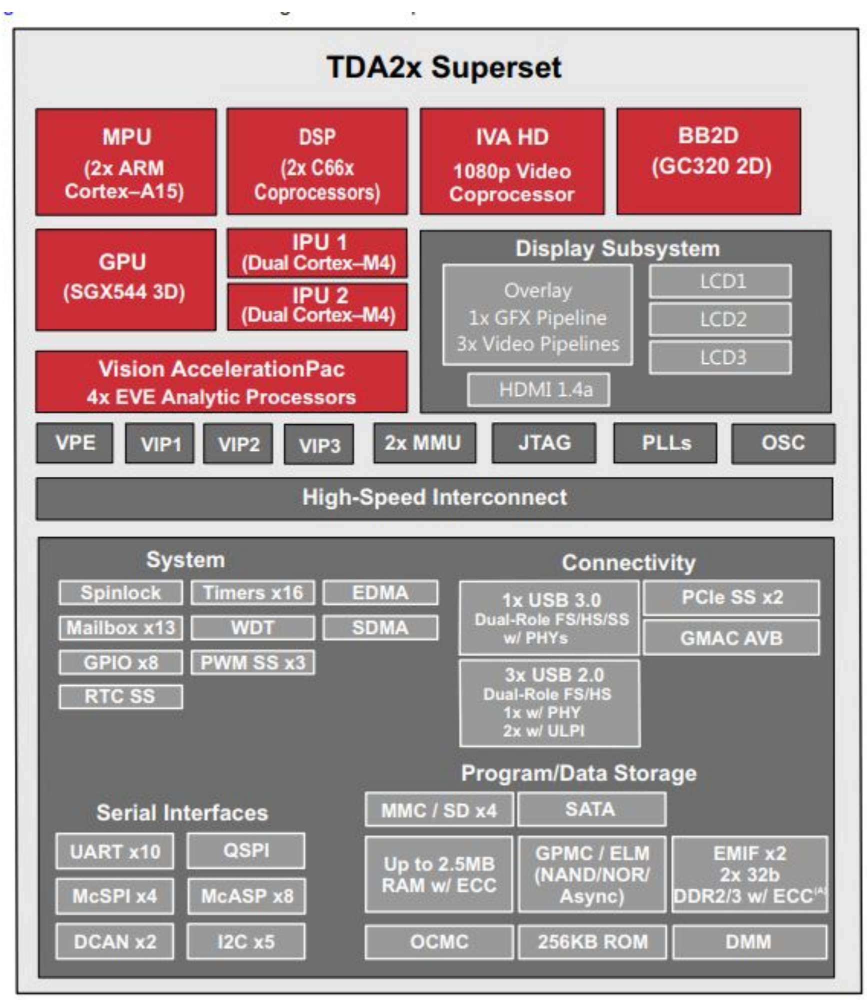
DSP ?

- DSP 장점
  - 다수의 연산 로직으로 병렬처리 가능(알고리즘 처리 속도 증가)
  - DMA 기능이 강력함
    - DMA 기능만으로도 일부 이미지 처리가 가능함

- DSP 단점
  - 인터럽트 발생시 명령어가 길어서 연산속도에 영향을 많음
    - 최적화시 기본적으로 인터럽트가 Disable됨으로 인터럽트 사용시에는 최적화 옵션 사용시 주의가 필요함
    - ARM 프로세서와 듀얼로 많이 사용
  - 최적화에 따른 속도 편차가 심함
    - 연산모듈은 8개이나 명령어 종류에 따른 동시 실행이 안되는 경우가 있음
    - 컴파일러 옵션만으로는 최적화의 한계가 있음으로 TI에서 제공하는 최적화 라이브러리 사용 권장
  - 캐시에 대한 영향이 크다
    - 명령어도 길고 데이터도 크기때문에 캐시 메모리에서 실행시와 외부메모리에서 실행시 속도 편차가 큼

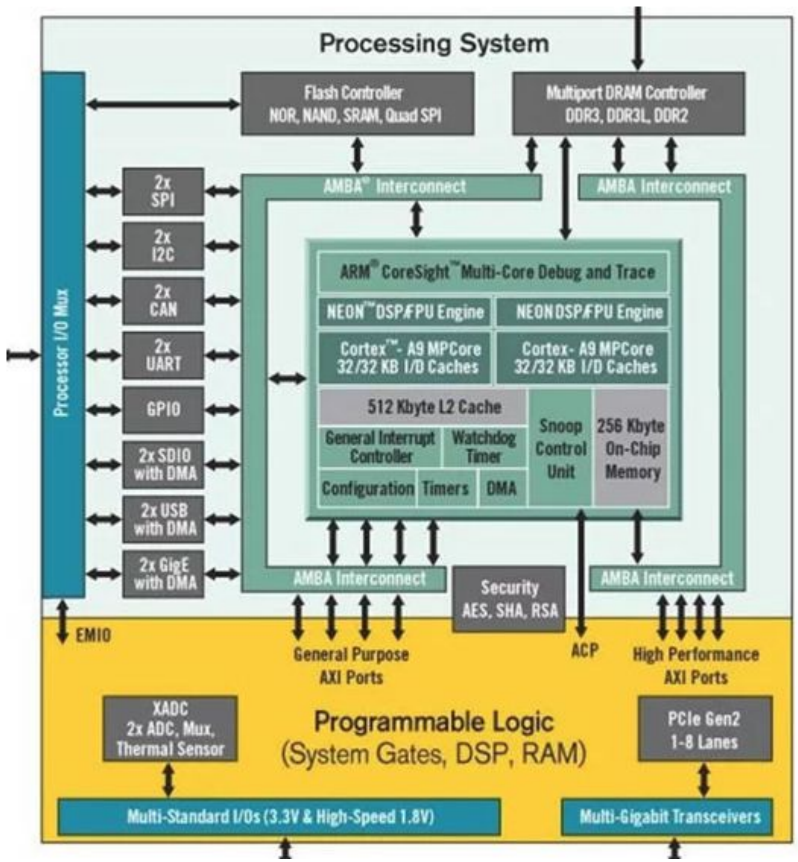
DSP ?

- DSP는 연산용으로 사용하고 ARM 코어 등을 사용



SoC FPGA

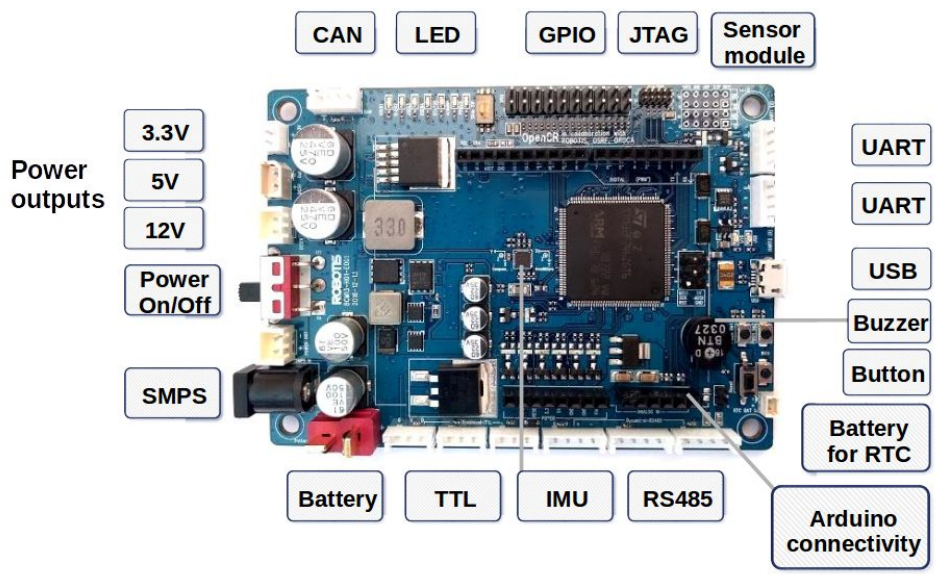
- FPGA 내부에 물리적인 ARM 코어등을 내장하여 FPGA와 ARM 코어 활용도 높임



<Xilinx Zynq>

OpenCR (Open-source Control Module for ROS)

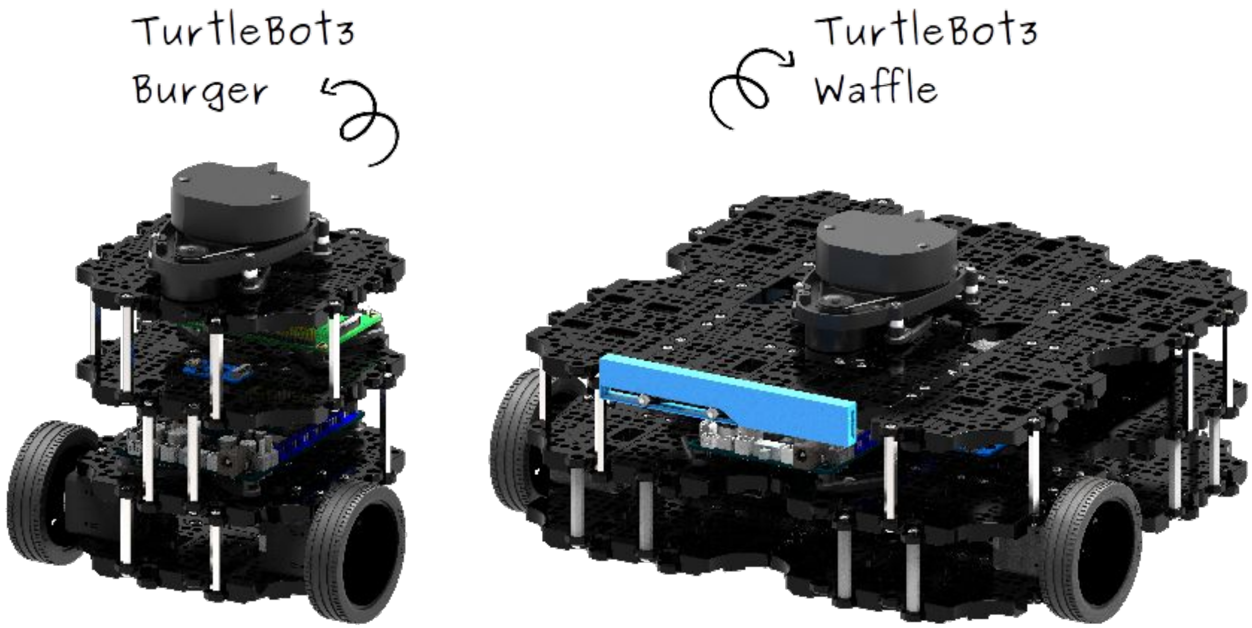
- STM32F746ZGT6 216Mhz, Cortex-M7, 1MB Flash, 320KB SRAM
- 아두이노 우노 핀 헤더
- 아두이노 IDE 개발환경 지원
- 다이나믹셀/올로/UART/CAN 인터페이스
- 배터리 입력 및 전원 출력(12V/5V/3.3V)



https://github.com/ROBOTIS-GIT/OpenCR/wiki

OpenCR (Open-source Control Module for ROS)

- Turtlebot3 Burger/Waffle의 제어기로 사용됨



OpenCR (Open-source Control Module for ROS)

- 활용 예제
  https://youtu.be/-_kBfIS6wJs

OpenCR (Open-source Control Module for ROS)

- 하드웨어 자료
  - https://github.com/ROBOTIS-GIT/OpenCR-Hardware
- 펌웨어 자료
  - https://github.com/ROBOTIS-GIT/OpenCR