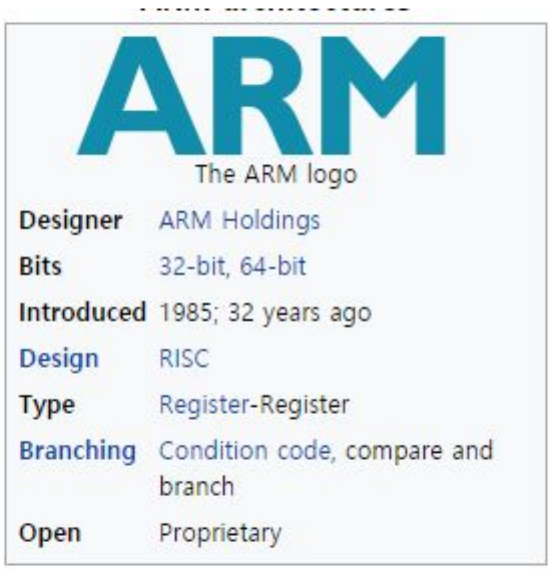
ARM 프로세서 개요

Hancheol Cho

ARM (Advanced RISC Machine)

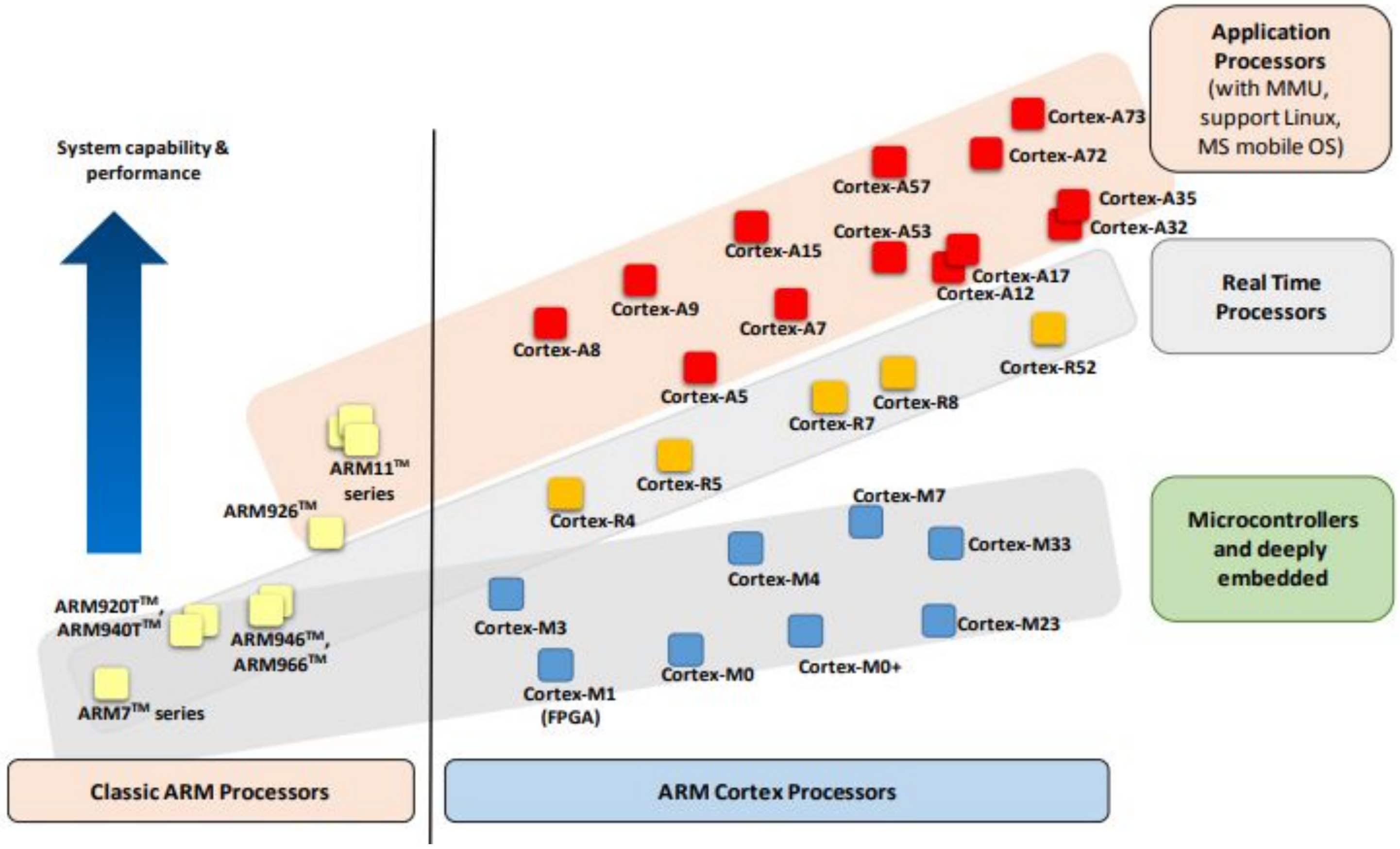

The ARM logo

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

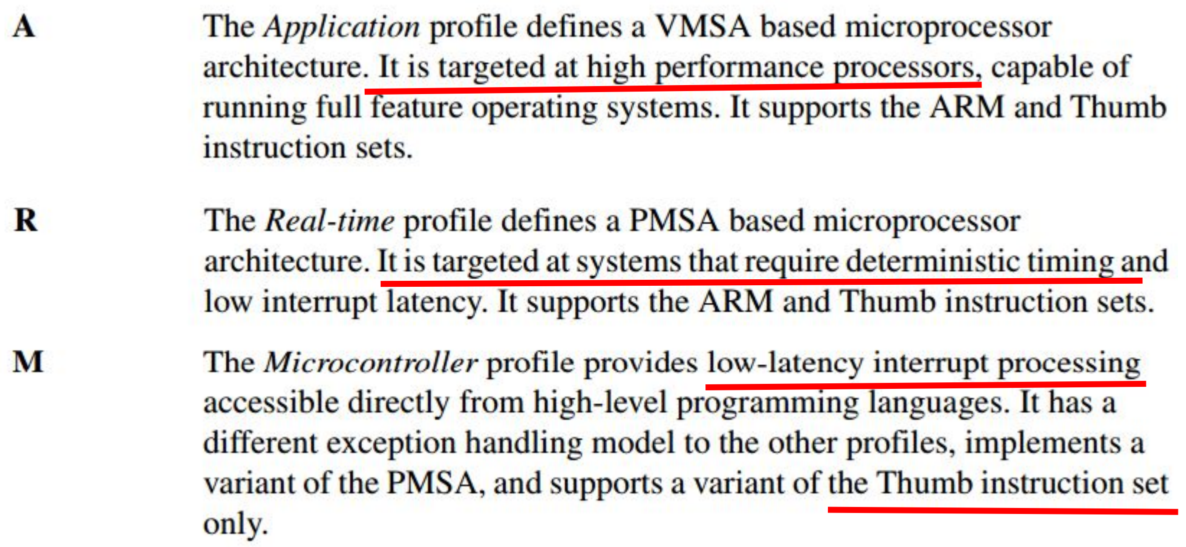
출처

ARM Processor Family



출처 : arm.com

ARM Architecture 분류



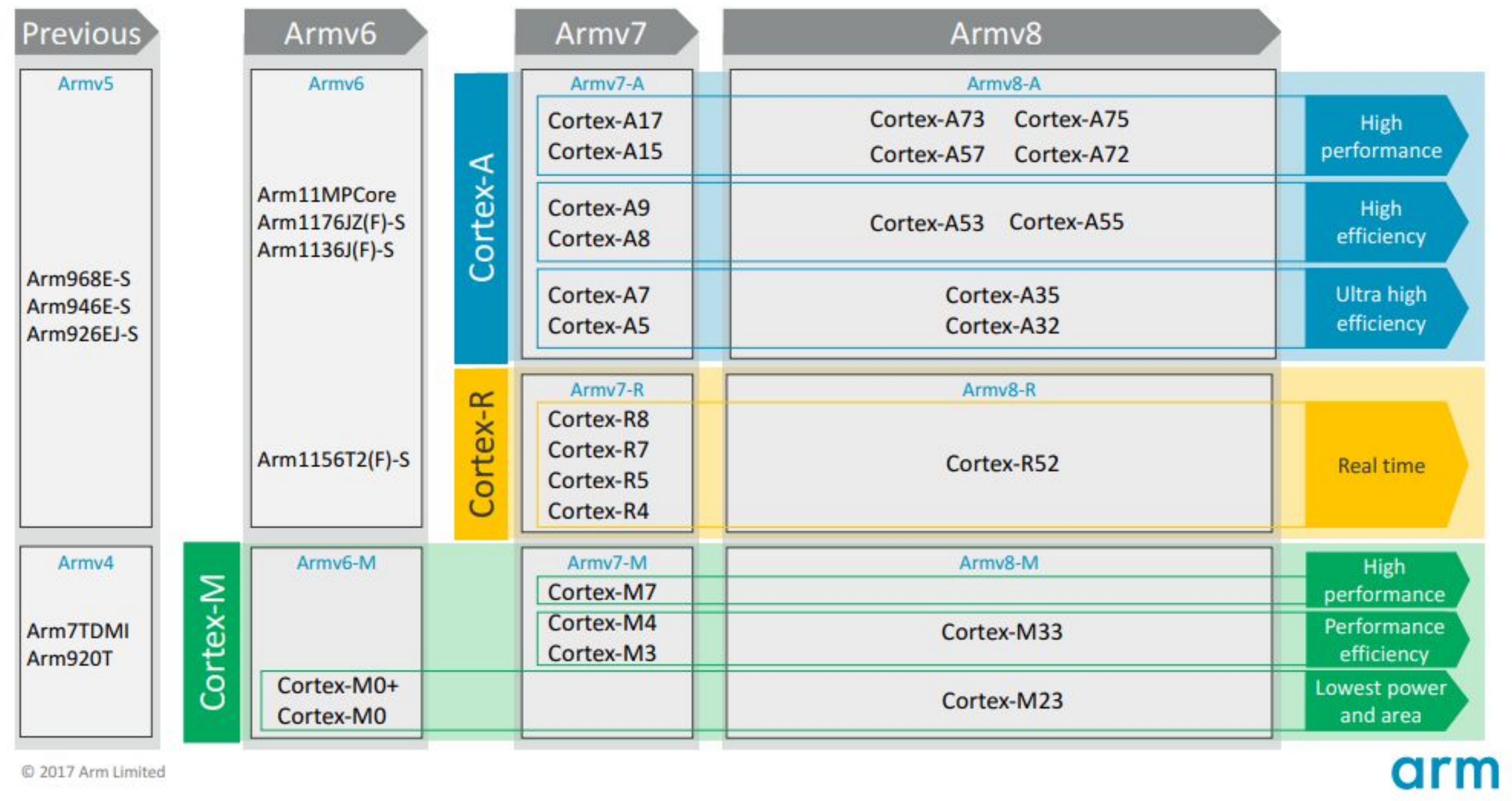
A The *Application* profile defines a VMSA based microprocessor architecture. It is targeted at high performance processors, capable of running full feature operating systems. It supports the ARM and Thumb instruction sets.

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

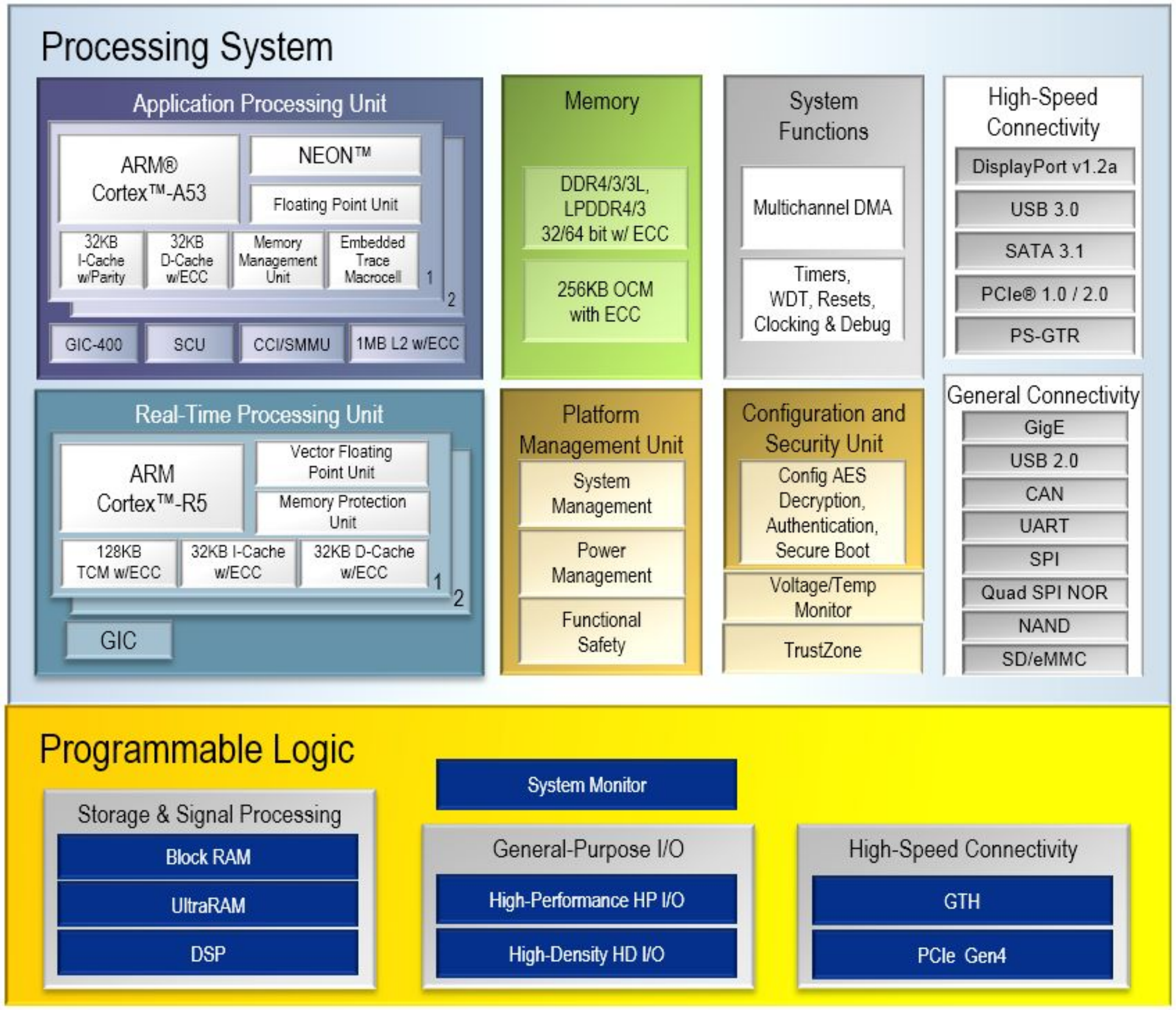


		Previous	Armv6	Armv7	Armv8	
Cortex-A		Armv5	Armv6	Armv7-A	Armv8-A	
			Arm11MPCore Arm1176JZ(F)-S Arm1136J(F)-S	Cortex-A17 Cortex-A15	Cortex-A73 Cortex-A75 Cortex-A57 Cortex-A72	High performance
		Arm968E-S Arm946E-S Arm926EJ-S		Cortex-A9 Cortex-A8	Cortex-A53 Cortex-A55	High efficiency
				Cortex-A7 Cortex-A5	Cortex-A35 Cortex-A32	Ultra high efficiency
Cortex-R				Armv7-R	Armv8-R	
			Arm1156T2(F)-S	Cortex-R8 Cortex-R7 Cortex-R5 Cortex-R4	Cortex-R52	Real time
	Cortex-M	Armv4	Armv6-M	Armv7-M	Armv8-M	
		Arm7TDMI Arm920T		Cortex-M7		High performance
				Cortex-M4 Cortex-M3	Cortex-M33	Performance efficiency
			Cortex-M0+ Cortex-M0		Cortex-M23	Lowest power and area

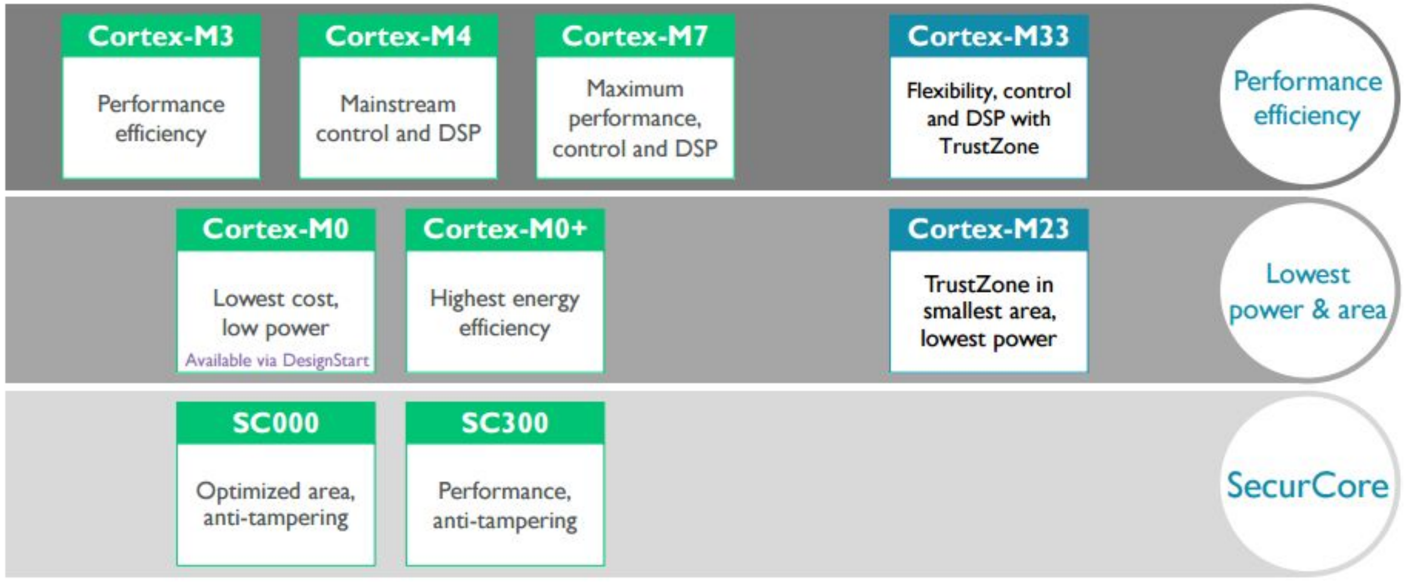
arm

출처 : arm.com

Cortex-R5 적용 예 - Xilinx Zynq



Cortex-M Processor

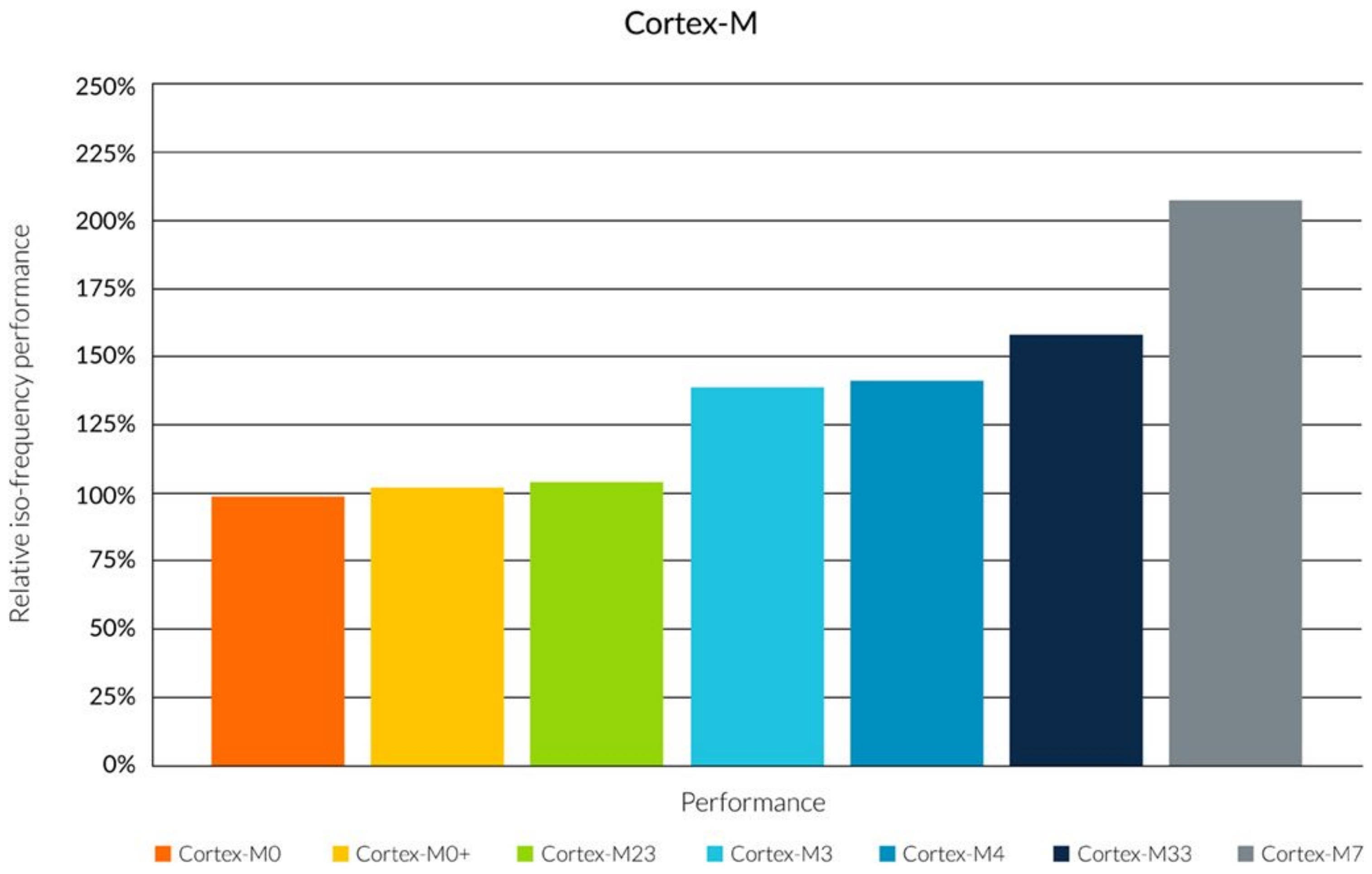


Cortex-M3
Performance efficiency
Cortex-M4
Mainstream control and DSP
Cortex-M7
Maximum performance, control and DSP
Cortex-M33
Flexibility, control and DSP with TrustZone
Performance efficiency
Cortex-M0
Lowest cost, low power
Available via DesignStart
Cortex-M0+
Highest energy efficiency
Cortex-M23
TrustZone in smallest area, lowest power
Lowest power & area
SC000
Optimized area, anti-tampering
SC300
Performance, anti-tampering
SecurCore

	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)

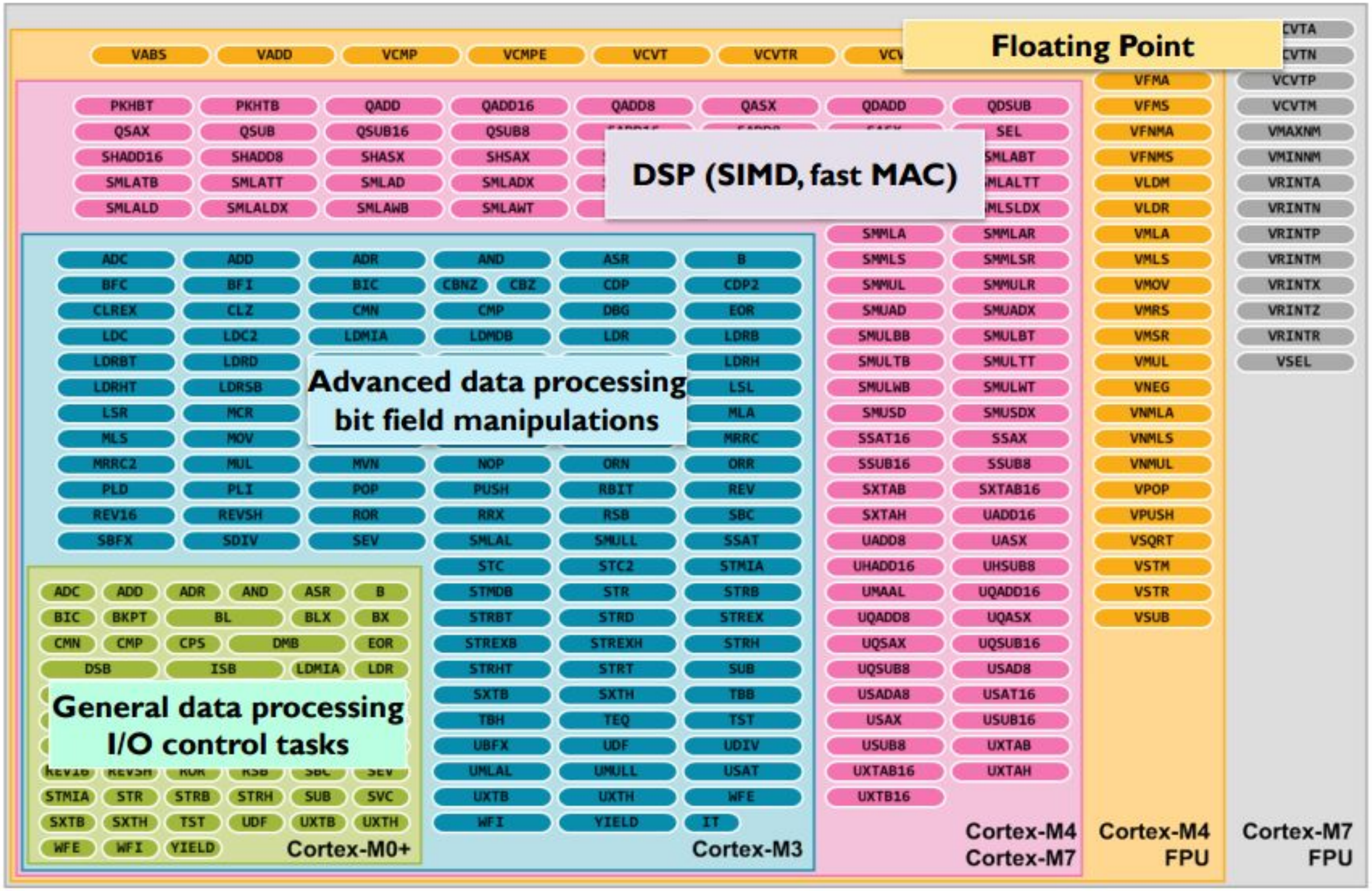
출처 : arm.com

Cortex-M Performance



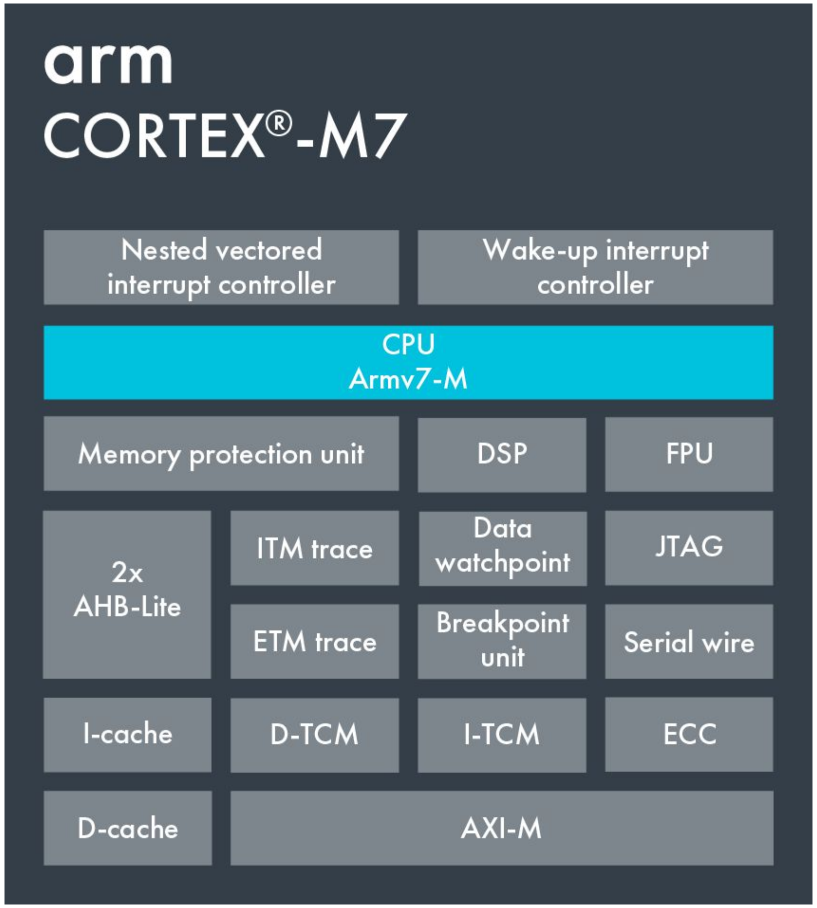
출처 : arm.com

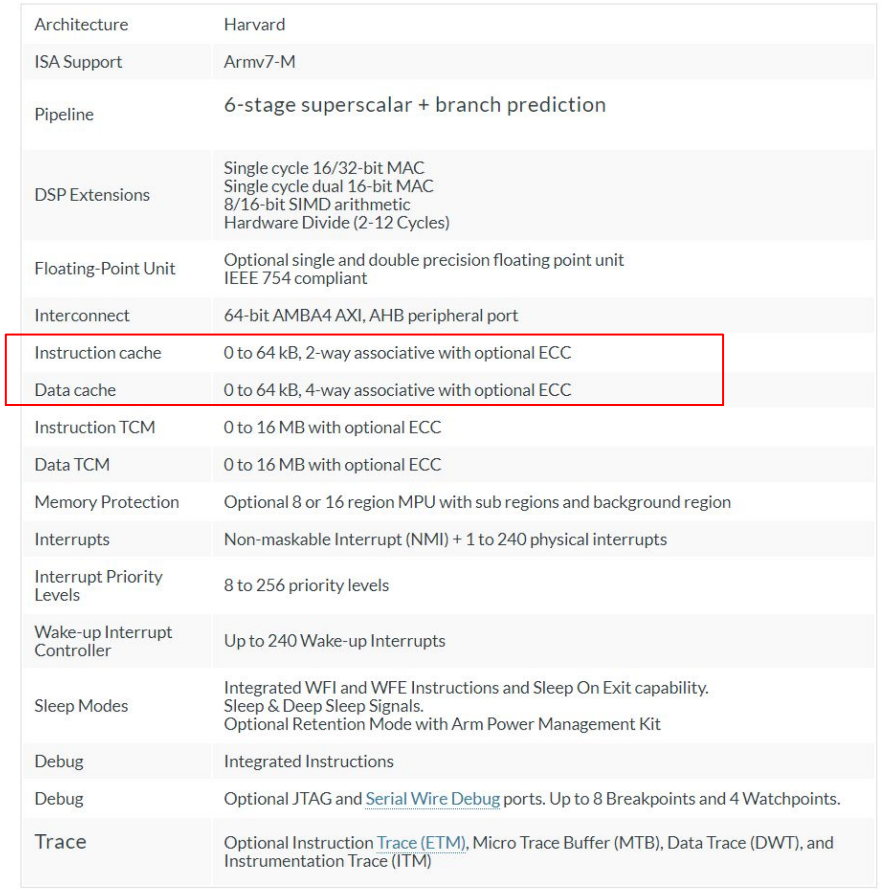
Cortex-M Instruction Set support


Floating Point
VABS VADD VCMP VCMPE VCVT VCVTR
VFMA VFMS VFNMA VFNMS VLDM VLDR VMLA VMLS VMOV VMRS VMSR VMUL VNEG VNMLA VNMLS VNMUL VPOP VPUSH VSQRT VSTM VSTR VSUB
VCVTA VCVTN VCVTP VCVTM VMAXNM VMINNM VRINTA VRINTN VRINTP VRINTM VRINTX VRINTZ VRINTR VSEL
DSP (SIMD, fast MAC)
PKHBT PKHTB QADD QADD16 QADD8 QASX QDADD QDSUB
QSAX QSUB QSUB16 QSUB8 SEL
SHADD16 SHADD8 SHASX SHSAX SMLABT
SMLATB SMLATT SMLAD SMLADX SMLALTT
SMLALD SMLALDX SMLAWB SMLAWT SMLSLDX
SMMLA SMMLAR SMMLS SMMLSR SMMUL SMMULR SMUAD SMUADX SMULBB SMULBT SMULTB SMULTT SMULWB SMULWT SMUSD SMUSDX SSAT16 SSAX SSUB16 SSUB8 SXTAB SXTAB16 SXTAH UADD16 UADD8 UASX UHADD16 UHSUB8 UMAAL UQADD16 UQADD8 UQASX UQSAX UQSUB16 UQSUB8 USAD8 USADA8 USAT16 USAX USUB16 USUB8 UXTAB UXTAB16 UXTAH UXTB16
Advanced data processing
bit field manipulations
ADC ADD ADR AND ASR B
BFC BFI BIC CBNZ CBZ CDP CDP2
CLREX CLZ CMN CMP DBG EOR
LDC LDC2 LDMIA LDMDB LDR LDRB
LDRBT LDRD LDRH
LDRHT LDRSB LSL
LSR MCR MLA
MLS MOV MRRC
MRRC2 MUL MVN NOP ORN ORR
PLD PLI POP PUSH RBIT REV
REV16 REVSH ROR RRX RSB SBC
SBFX SDIV SEV SMLAL SMULL SSAT
STC STC2 STMIA
STMDB STR STRB
STRBT STRD STREX
STREXB STREXH STRH
STRHT STRT SUB
SXTB SXTH TBB
TBH TEQ TST
UBFX UDF UDIV
UMLAL UMULL USAT
UXTB UXTH WFE
WFI YIELD IT
General data processing
I/O control tasks
ADC ADD ADR AND ASR B
BIC BKPT BL BLX BX
CMN CMP CPS DMB EOR
DSB ISB LDMIA LDR
REV16 REVSH ROR RSB SBC SEV
STMIA STR STRB STRH SUB SVC
SXTB SXTH TST UDF UXTB UXTH
WFE WFI YIELD
Cortex-M0+
Cortex-M3
Cortex-M4
Cortex-M7
Cortex-M4
FPU
Cortex-M7
FPU

출처 : arm.com

Cortex-M7





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)

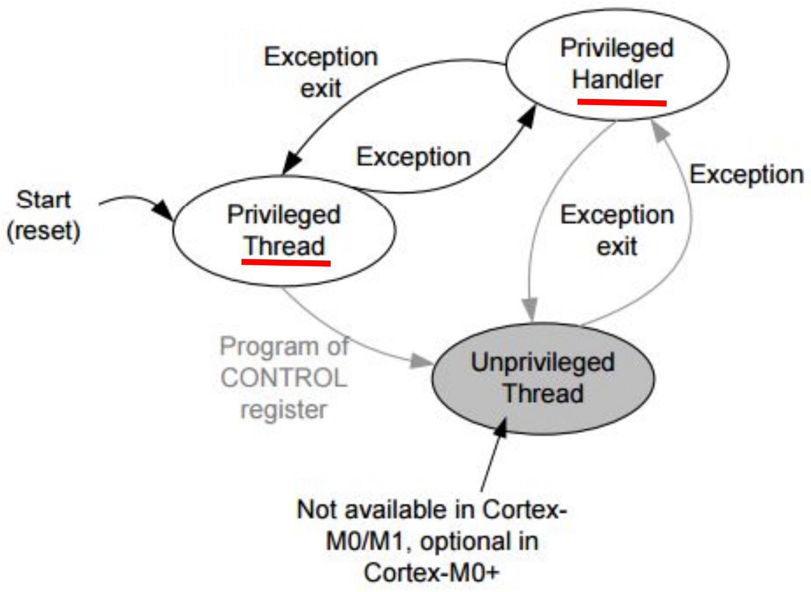
7.1 Why Cortex-M processors are easy to use

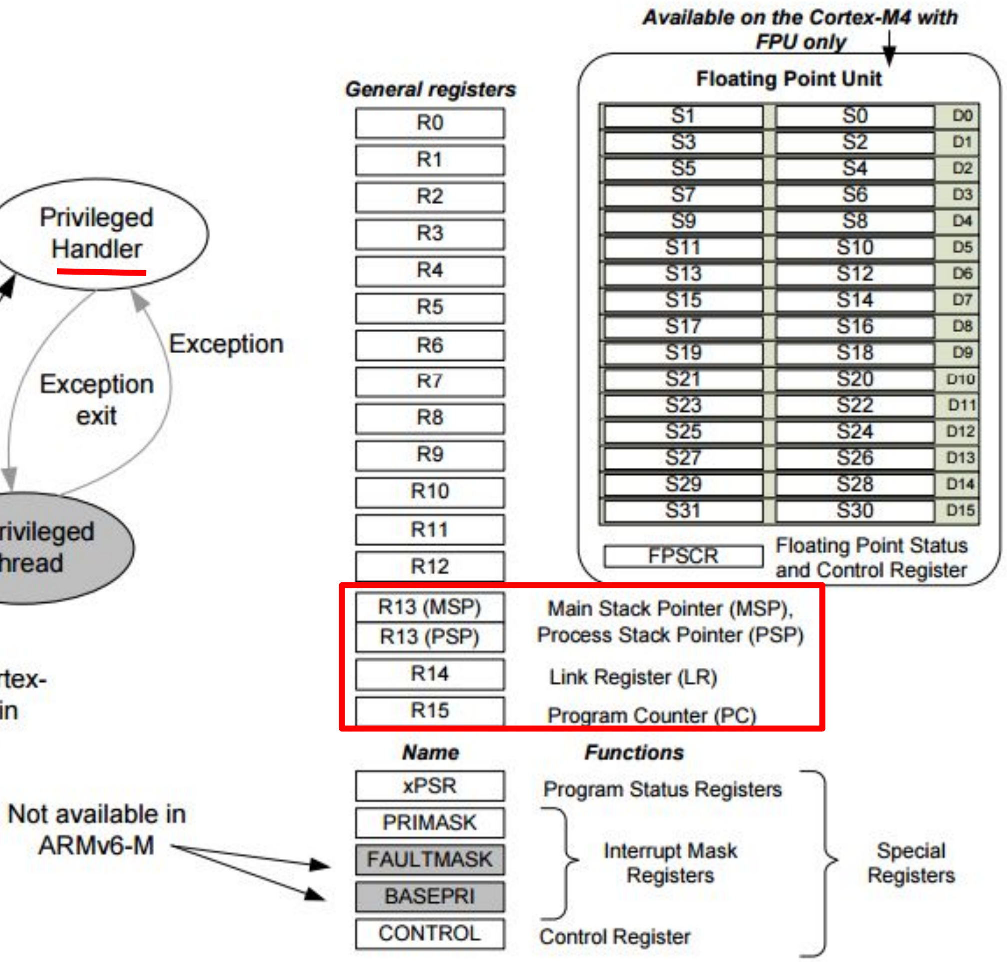
Although the Cortex-M processors are packed with features, they are also very easy to use. For instance, almost everything can be programmed in high-level language like C. Although there is a big variety of different Cortex-M processor-based products (e.g. with different memory size, peripherals, performance, 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

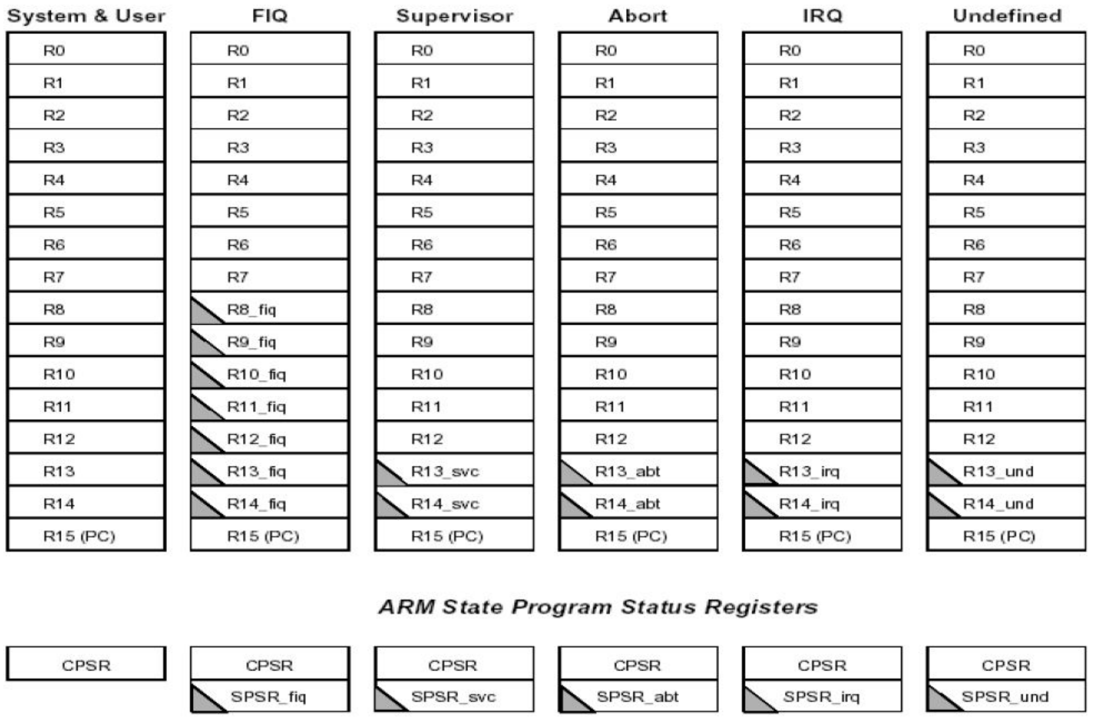




Real Time OS ?

Context

- 스레드/태스크가 실행되기 위한 최소한의 데이터
  - CPU 레지스터, 스택포인터 등등..

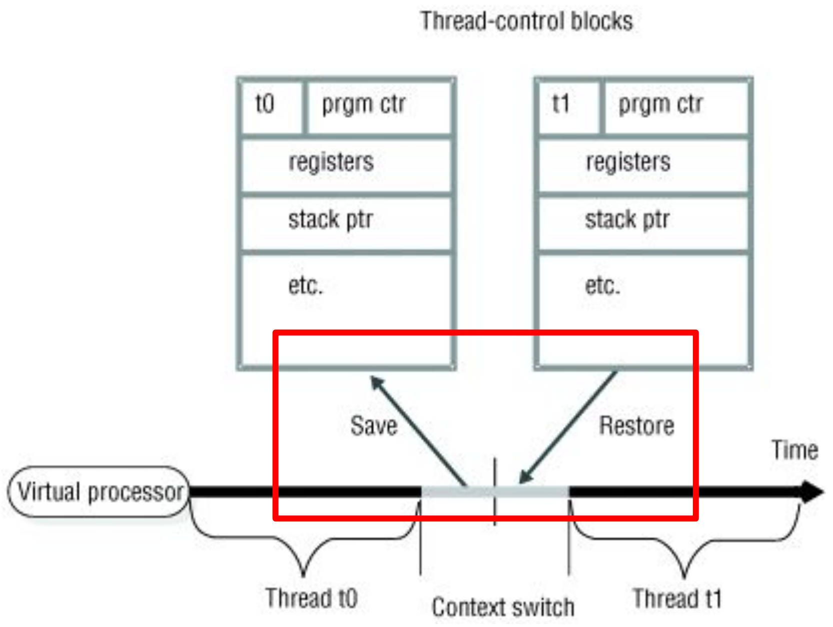


System & User	FIQ	Supervisor	Abort	IRQ	Undefined
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8_fiq	R8	R8	R8	R8
R9	R9_fiq	R9	R9	R9	R9
R10	R10_fiq	R10	R10	R10	R10
R11	R11_fiq	R11	R11	R11	R11
R12	R12_fiq	R12	R12	R12	R12
R13	R13_fiq	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_fiq	R14_svc	R14_abt	R14_irq	R14_und
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)

ARM State Program Status Registers

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

Context Switching



출처

스케줄러

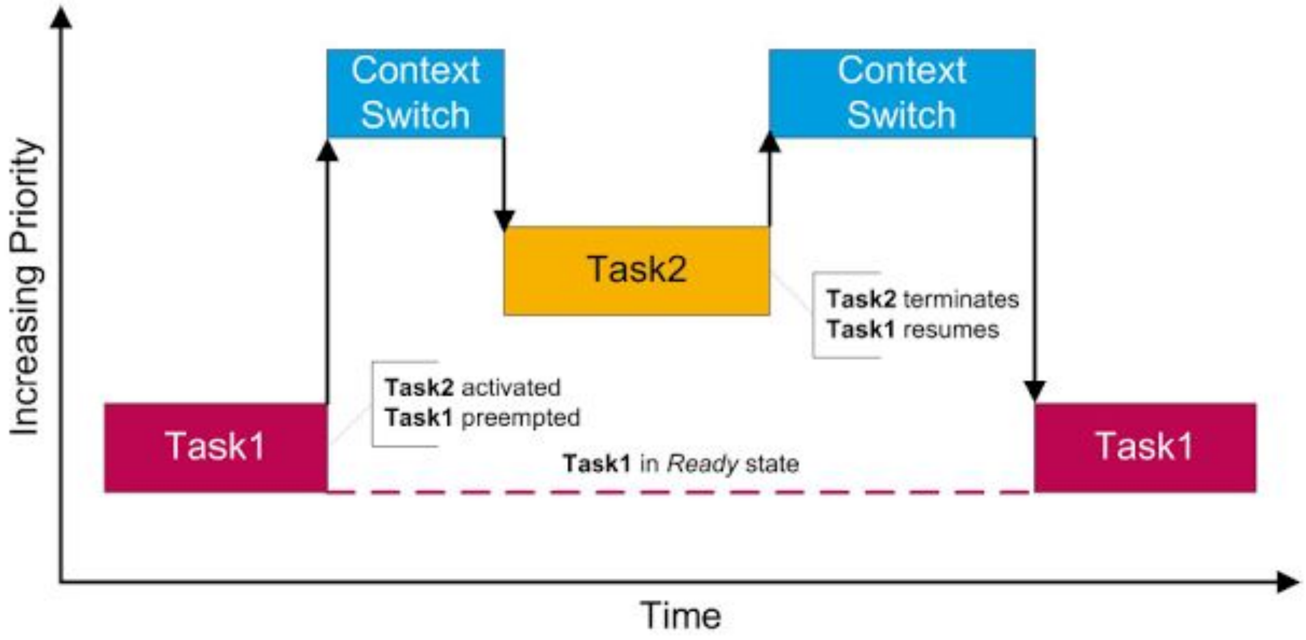
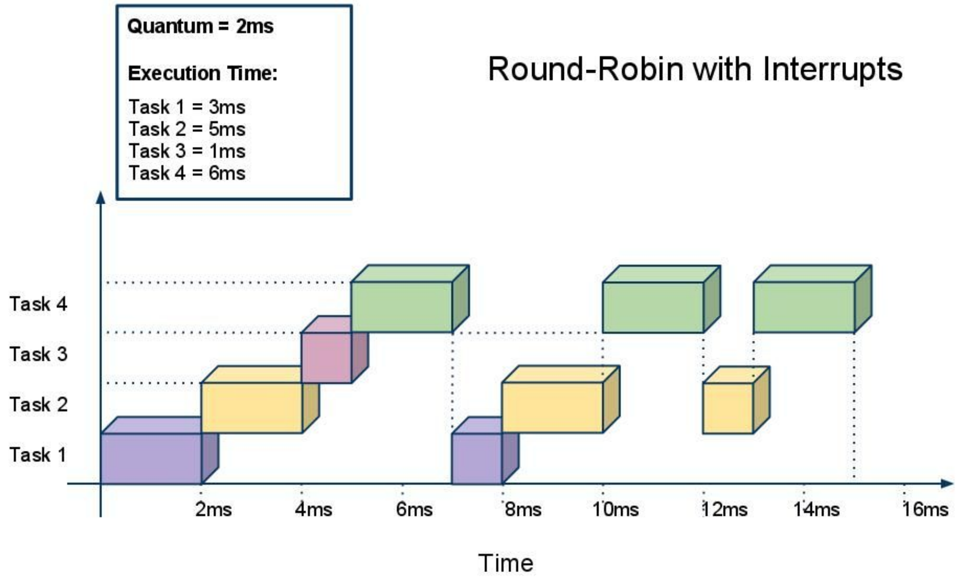


Figure 4.1: Preemptive scheduling of tasks

출처

스케줄러


Quantum = 2ms
Execution Time:
Task 1 = 3ms
Task 2 = 5ms
Task 3 = 1ms
Task 4 = 6ms
Round-Robin with Interrupts
Task 4
Task 3
Task 2
Task 1
2ms
4ms
6ms
8ms
10ms
12ms
14ms
16ms
Time

출처

RTOS

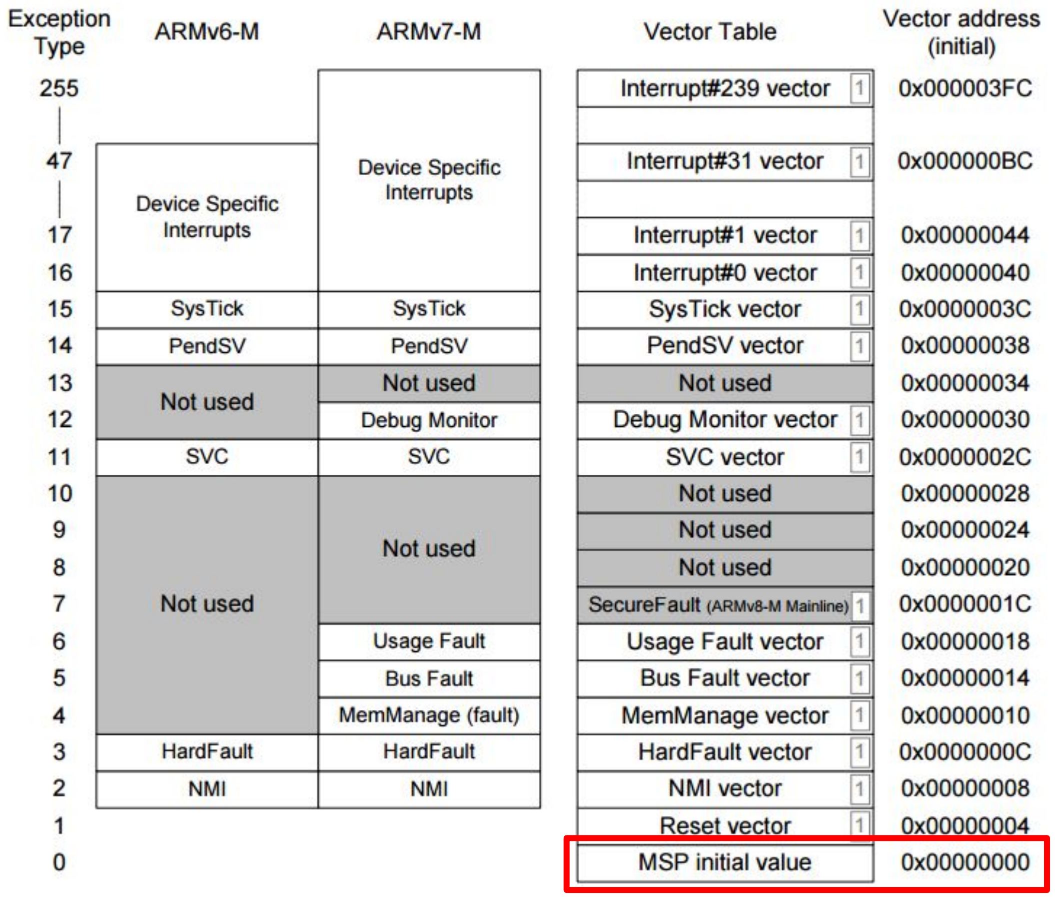
µC/OS™
RTOS and Stacks

freeRTOS

VxWorks

Exception Vector

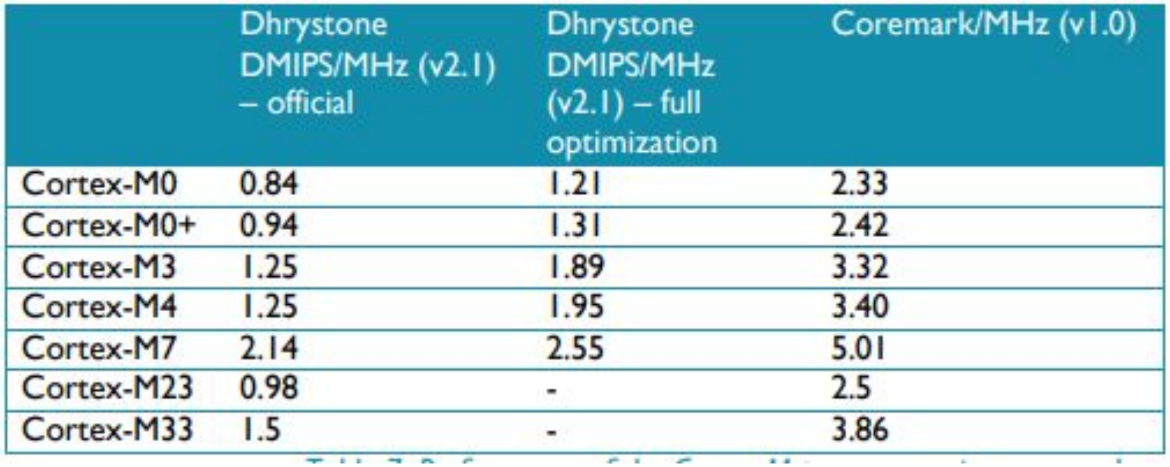
- 스택포인터 초기값을 지정 가능
  - 스타트업 코드도 C언어로 작성 가능



Exception Type	ARMv6-M	ARMv7-M	Vector Table		Vector address (initial)
255		Device Specific Interrupts	Interrupt#239 vector	1	0x000003FC
47	Device Specific Interrupts		Interrupt#31 vector	1	0x000000BC
17			Interrupt#1 vector	1	0x00000044
16			Interrupt#0 vector	1	0x00000040
15	SysTick	SysTick	SysTick vector	1	0x0000003C
14	PendSV	PendSV	PendSV vector	1	0x00000038
13	Not used	Not used	Not used		0x00000034
12		Debug Monitor	Debug Monitor vector	1	0x00000030
11	SVC	SVC	SVC vector	1	0x0000002C
10	Not used	Not used	Not used		0x00000028
9			Not used		0x00000024
8			Not used		0x00000020
7			SecureFault (ARMv8-M Mainline)	1	0x0000001C
6		Usage Fault	Usage Fault vector	1	0x00000018
5		Bus Fault	Bus Fault vector	1	0x00000014
4		MemManage (fault)	MemManage vector	1	0x00000010
3	HardFault	HardFault	HardFault vector	1	0x0000000C
2	NMI	NMI	NMI vector	1	0x00000008
1			Reset vector	1	0x00000004
0			MSP initial value		0x00000000

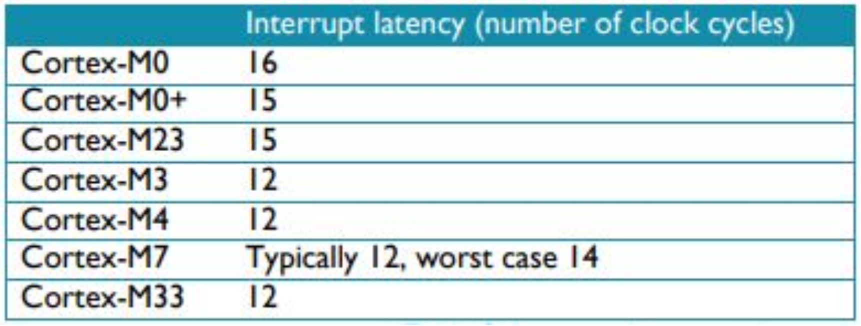
성능비교

- 데이터 처리 속도



	Dhrystone DMIPS/MHz (v2.1) – official	Dhrystone DMIPS/MHz (v2.1) – full optimization	Coremark/MHz (v1.0)
Cortex-M0	0.84	1.21	2.33
Cortex-M0+	0.94	1.31	2.42
Cortex-M3	1.25	1.89	3.32
Cortex-M4	1.25	1.95	3.40
Cortex-M7	2.14	2.55	5.01
Cortex-M23	0.98	-	2.5
Cortex-M33	1.5	-	3.86

- Interrupt Latency

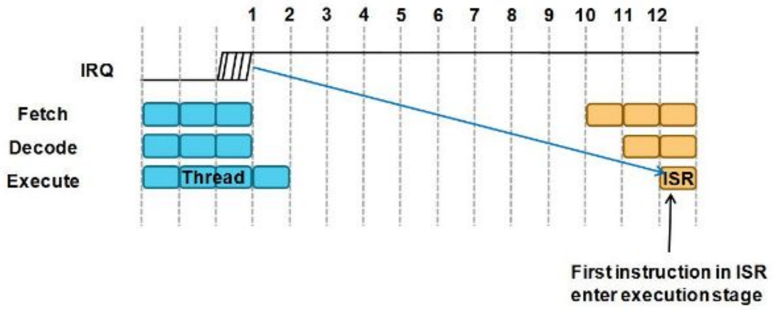


	Interrupt latency (number of clock cycles)
Cortex-M0	16
Cortex-M0+	15
Cortex-M23	15
Cortex-M3	12
Cortex-M4	12
Cortex-M7	Typically 12, worst case 14
Cortex-M33	12

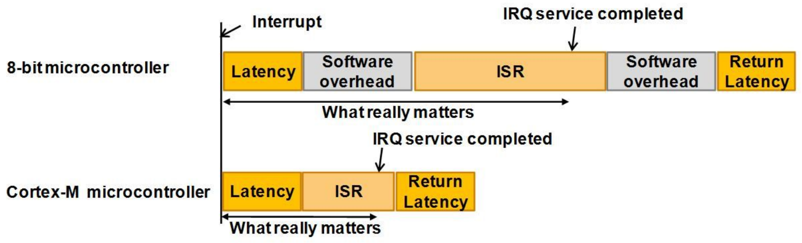
출처 : arm.com

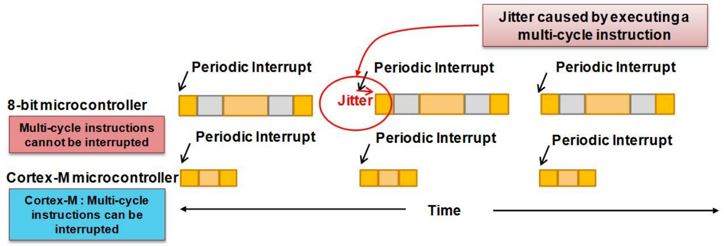
Interrupt Latency

- Interrupt Latency ?



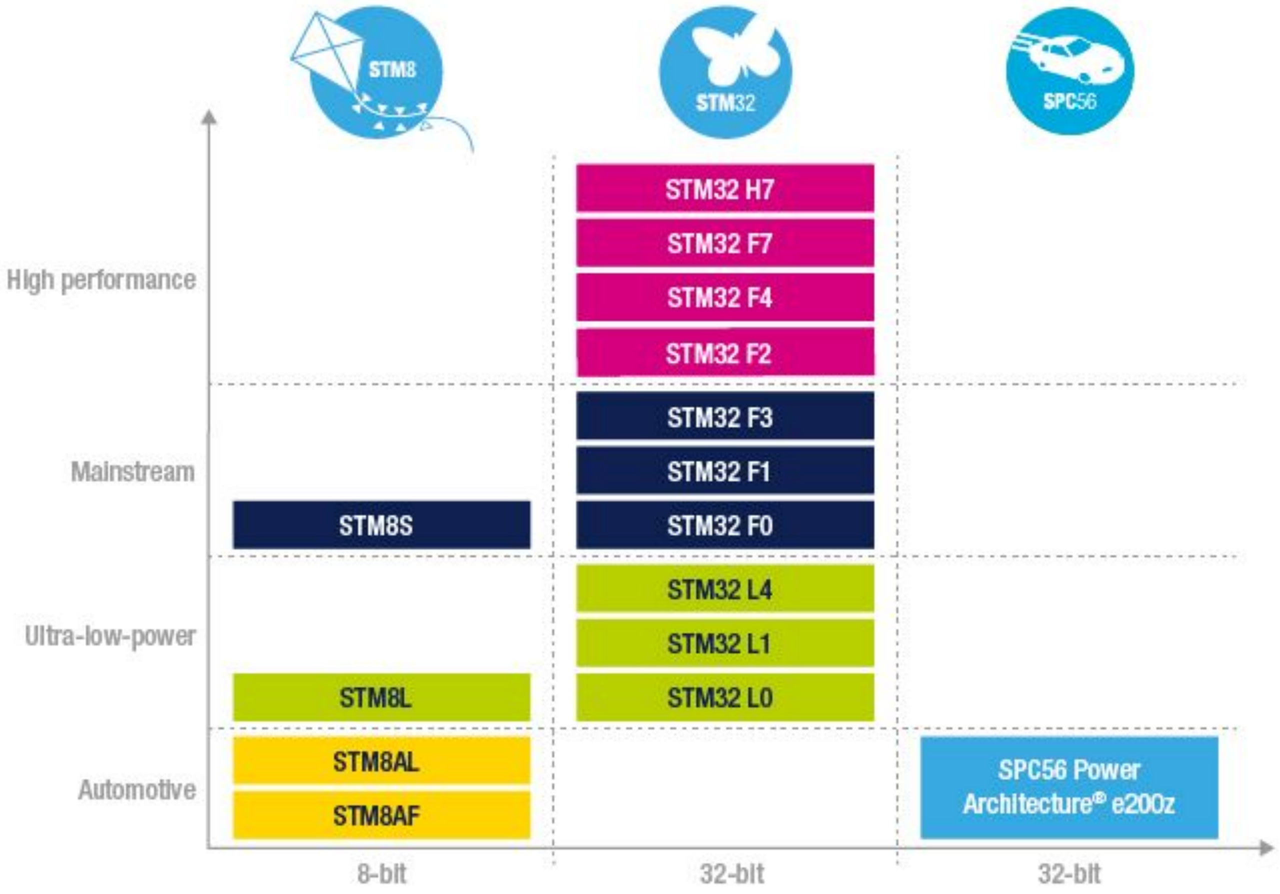
- 기존 마이크로 컨트롤러와 비교



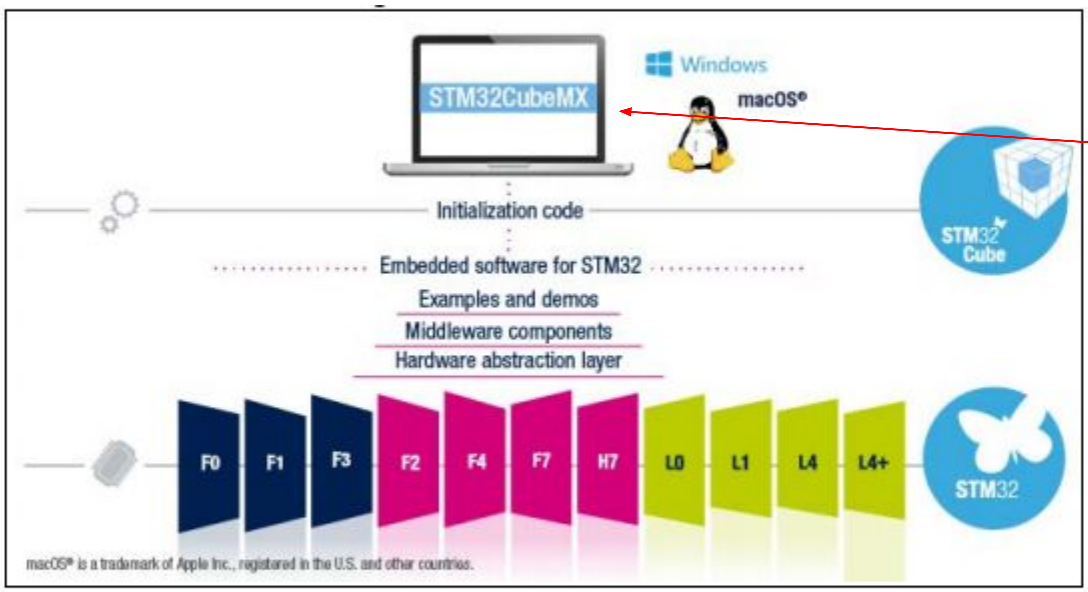


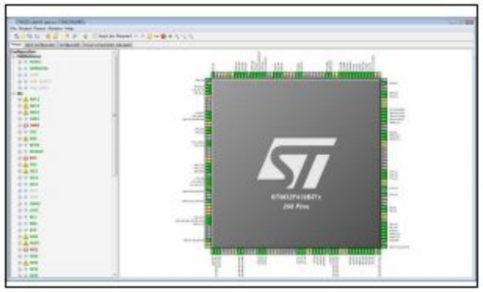
출처 : Link

ST사의 MCU 구성

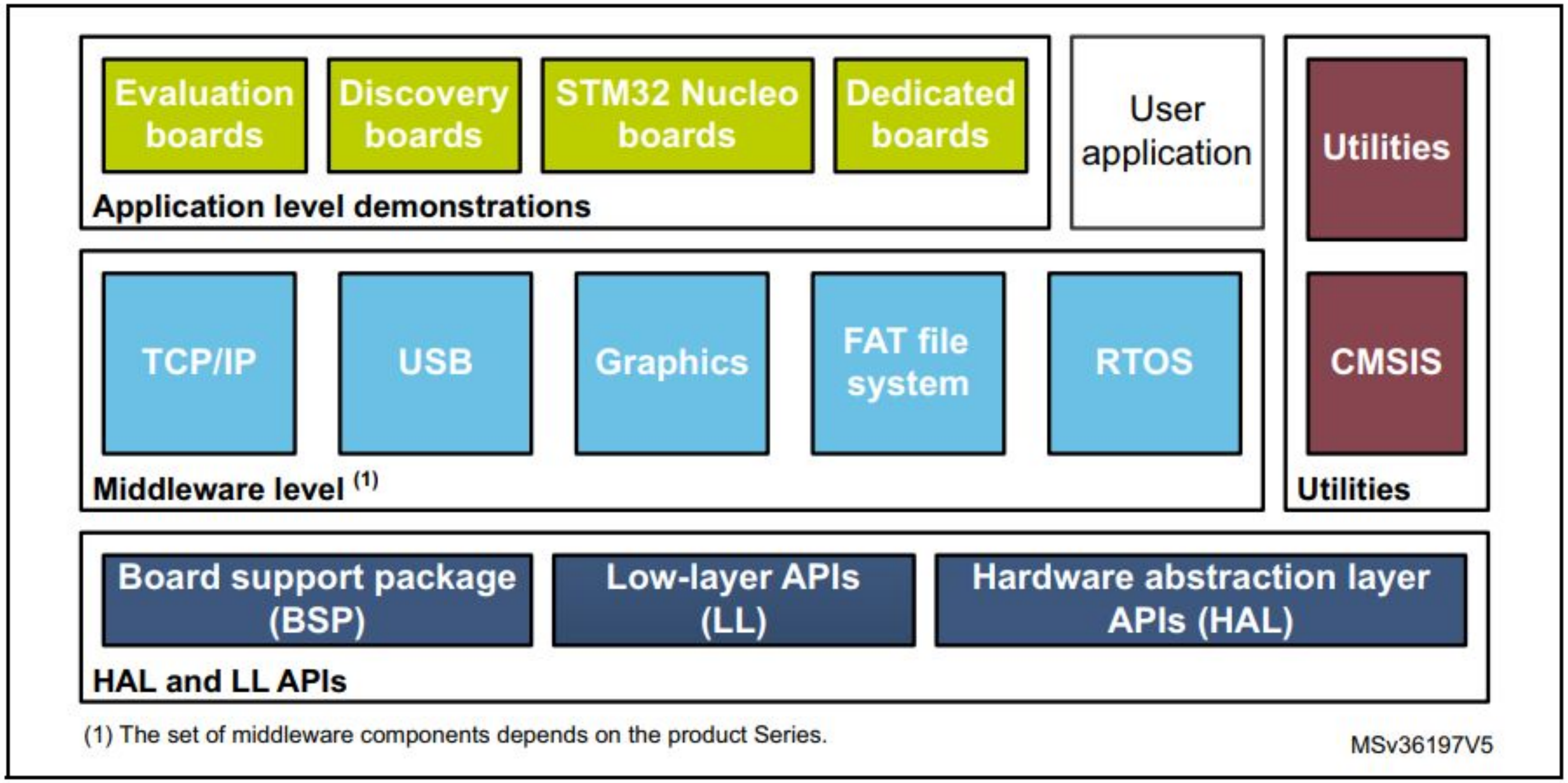

STM8
STM32
SPC56
High performance
Mainstream
Ultra-low-power
Automotive
STM32 H7
STM32 F7
STM32 F4
STM32 F2
STM32 F3
STM32 F1
STM32 F0
STM8S
STM32 L4
STM32 L1
STM32 L0
STM8L
STM8AL
STM8AF
SPC56 Power Architecture® e200z
8-bit
32-bit
32-bit

STM32CubeMX, STM32Cube


STM32CubeMX
Windows
macOS®
Initialization code
STM32 Cube
Embedded software for STM32
Examples and demos
Middleware components
Hardware abstraction layer
F0
F1
F3
F2
F4
F7
H7
L0
L1
L4
L4+
STM32
macOS® is a trademark of Apple Inc., registered in the U.S. and other countries.

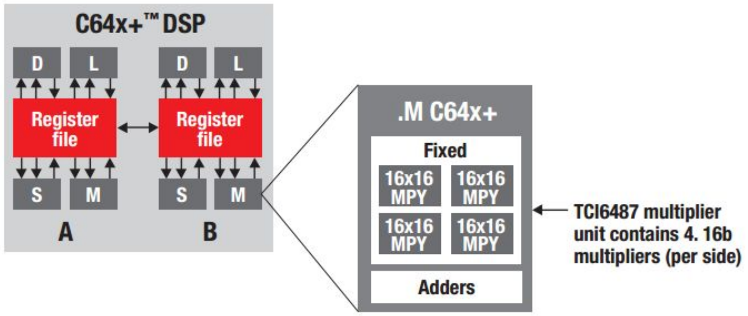


STM32Cube

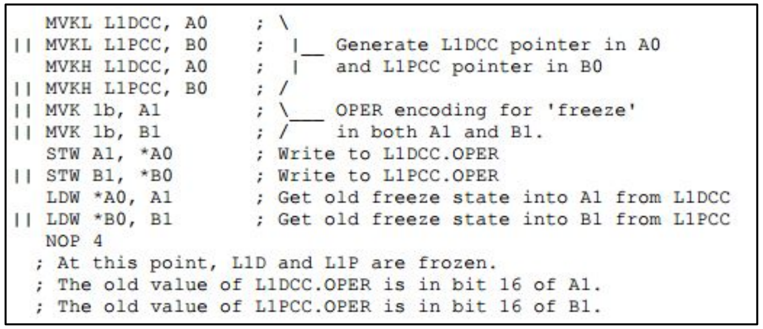

Evaluation boards
Discovery boards
STM32 Nucleo boards
Dedicated boards
User application
Utilities
Application level demonstrations
TCP/IP
USB
Graphics
FAT file system
RTOS
CMSIS
Middleware level (1)
Utilities
Board support package (BSP)
Low-layer APIs (LL)
Hardware abstraction layer APIs (HAL)
HAL and LL APIs
(1) The set of middleware components depends on the product Series.
MSv36197V5

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



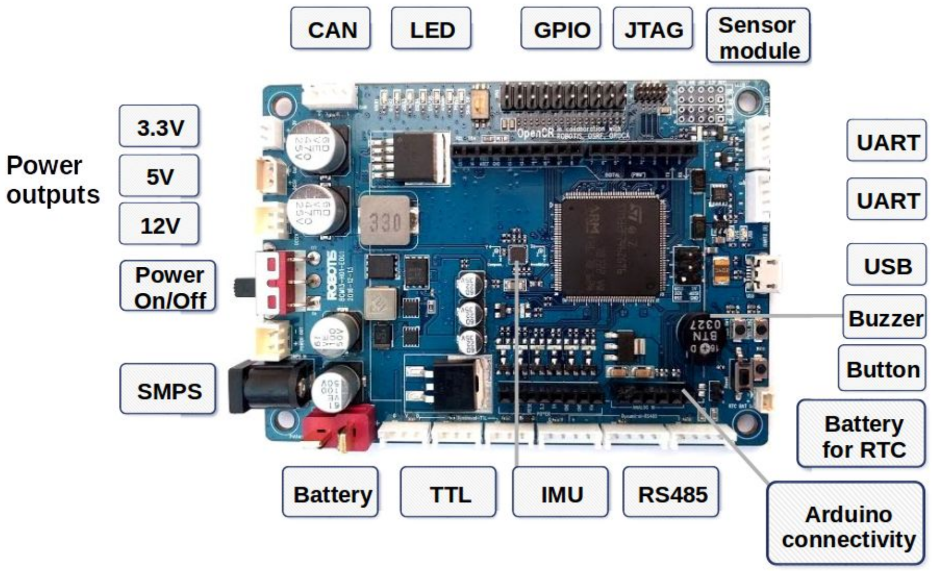
```
   MVKL L1DCC, A0     ; \
|| MVKL L1PCC, B0     ;  |__ Generate L1DCC pointer in A0
   MVKH L1DCC, A0     ;  |   and L1PCC pointer in B0
|| MVKH L1PCC, B0     ; /
|| MVK 1b, A1         ; \___ OPER encoding for 'freeze'
|| MVK 1b, B1         ; /    in both A1 and B1.
   STW A1, *A0        ; Write 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
  - 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

OpenCR (Open-source Control Module for ROS)

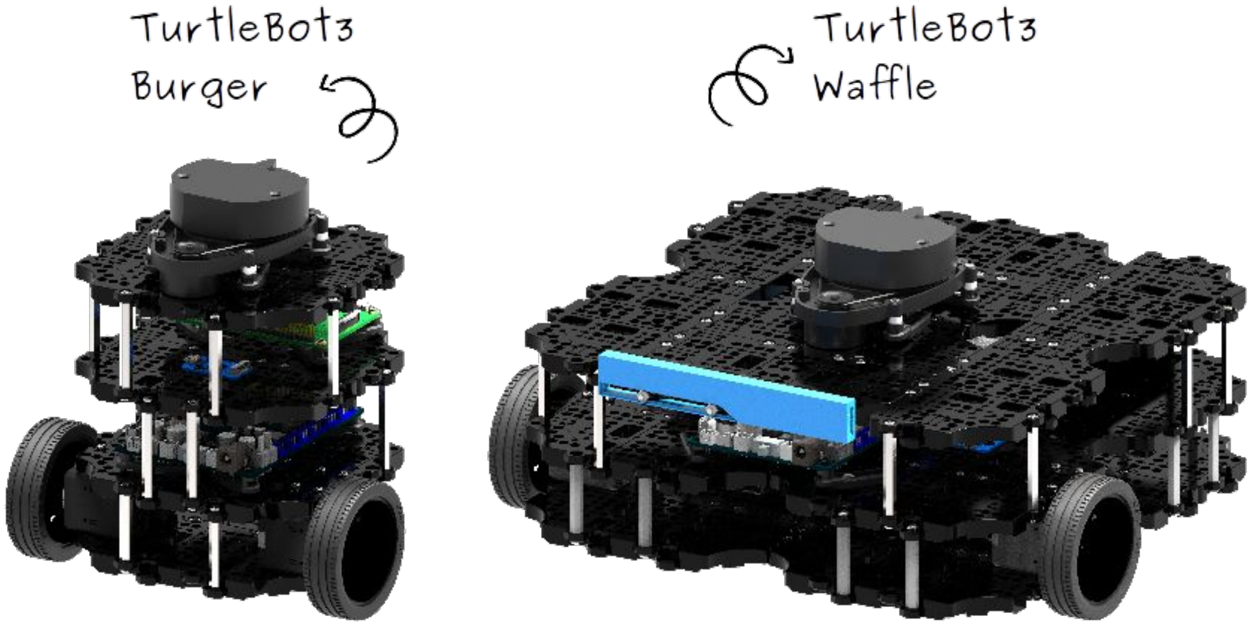
- STM32F746ZGT6 216Mhz, Cortex-M7, 1MB Flash, 320KB SRAM
- 아두이노 우노 핀 헤더
- 아두이노 IDE 개발환경 지원
- 다이나믹셀/올로/UART/CAN 인터페이스
- 배터리 입력 및 전원 출력(12V/5V/3.3V)



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

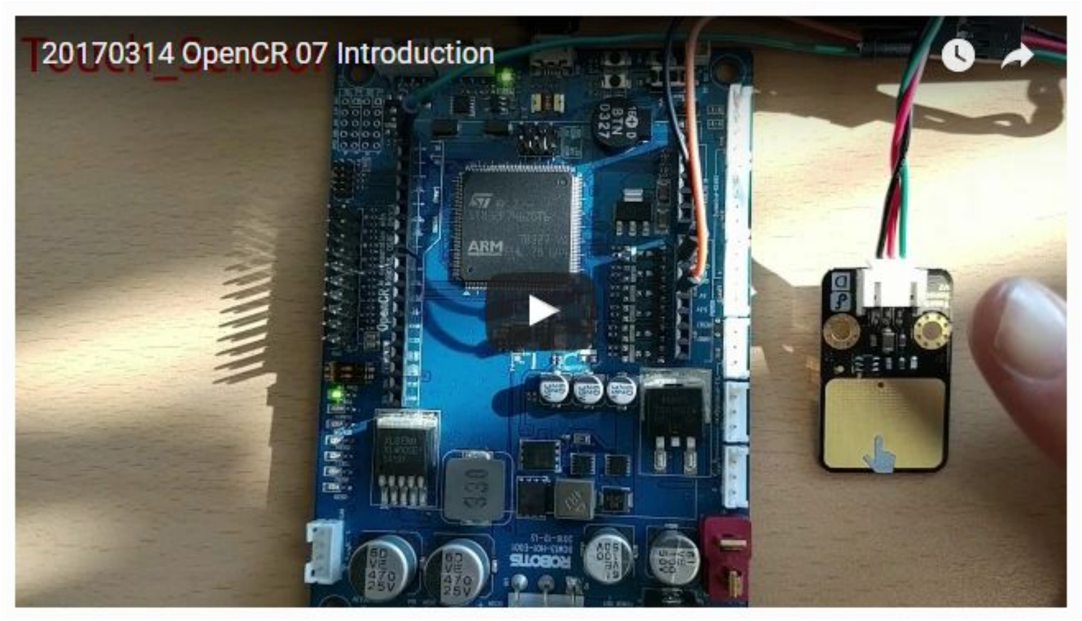
OpenCR (Open-source Control Module for ROS)

- 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

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