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

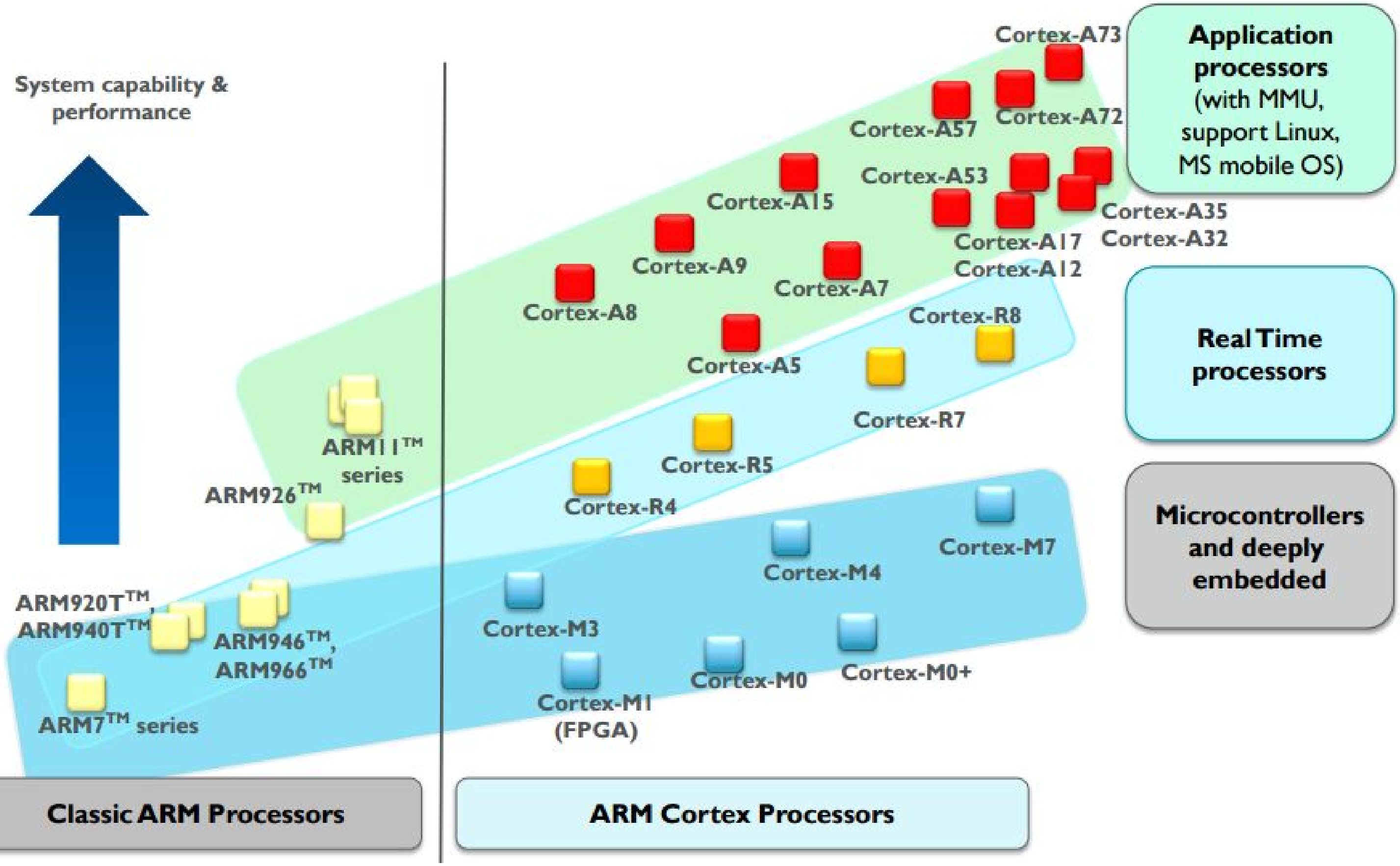
日 **소프트뱅크**, 모바일 반도체 1위 **ARM** 전격 **인수**...로봇·AI·IoT 융합 ...
조선비즈 - 2016. 7. 17.
일본 **소프트뱅크**가 영국 반도체 회사 암(**ARM**) 홀딩스를 234억 파운드(약 35조1800억원)에 전격 **인수**키로 해 배경에 관심이 쏠리고 있다. **ARM**은 ...

소프트뱅크, 영국 **ARM 인수** 의미와 파장은
심층 뉴스 - 전자신문 - 2016. 7. 17.

모두 보기

소프트뱅크, 미 보스턴 다이나믹스와 샤프트 인수! 스마트 로봇업계 지각 ...
www.seminartoday.net/news/articleView.html?idxno=9773 ▾
5일 전 - **소프트뱅크**는 알파벳(Alphabet) 산하의 로봇기업 2개사, 보스턴 ... 보스턴 **다이나**맥스의 개발된 로봇군(사진:보스턴 **다이나**맥스) ...

ARM Processor Family



ARM Architecture 분류

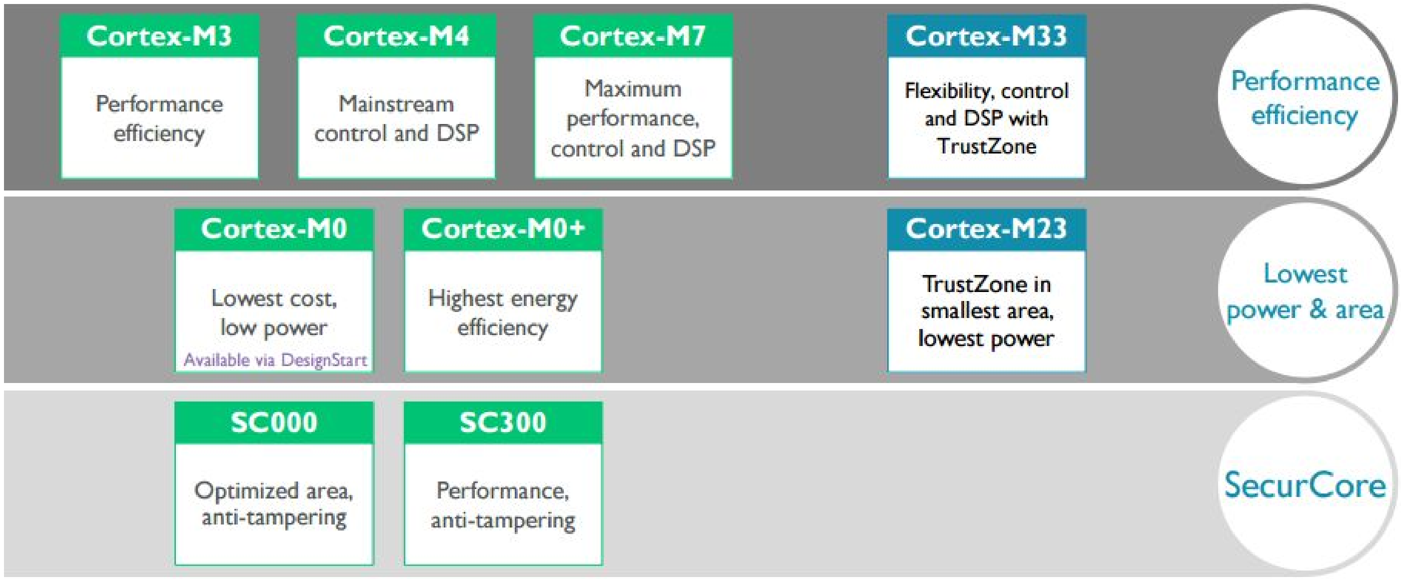
A The *Application* profile defines a VMSA based microprocessor architecture. It is targeted at high performance processors, capable of running full feature operating systems. It supports the ARM and Thumb instruction sets.

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.

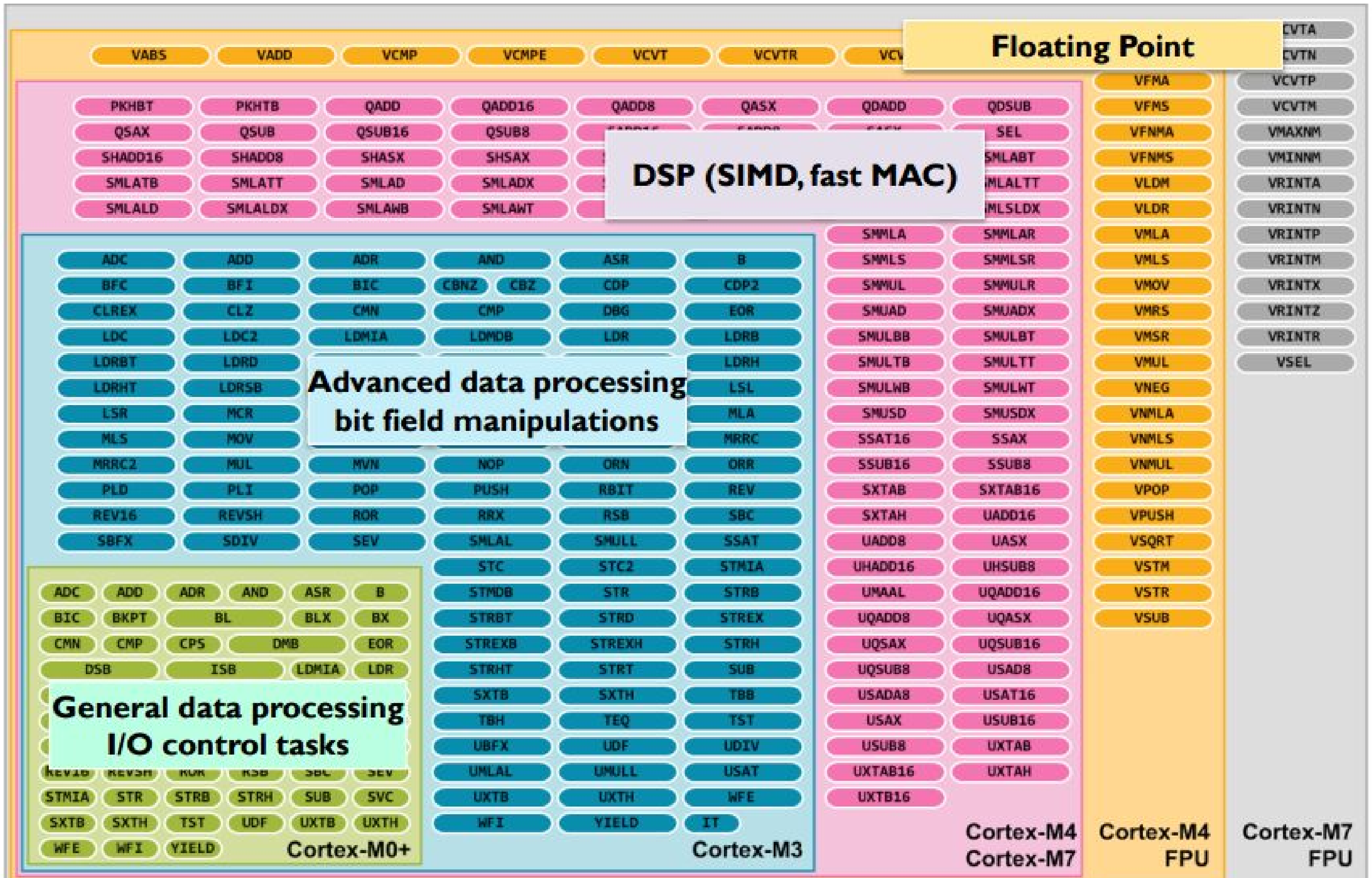
Profile	Architecture	Instruction Set	Processor
A-Profile	ARMv7-A	A32, T32	Cortex-A Series
R-Profile	ARMv7-R	A32, T32	Cortex-R Series
M-Profile	ARMv7-M	T32	Cortex-M Series
	ARMV6-M	T32	Cortex-M0 Series

Cortex-M Processor

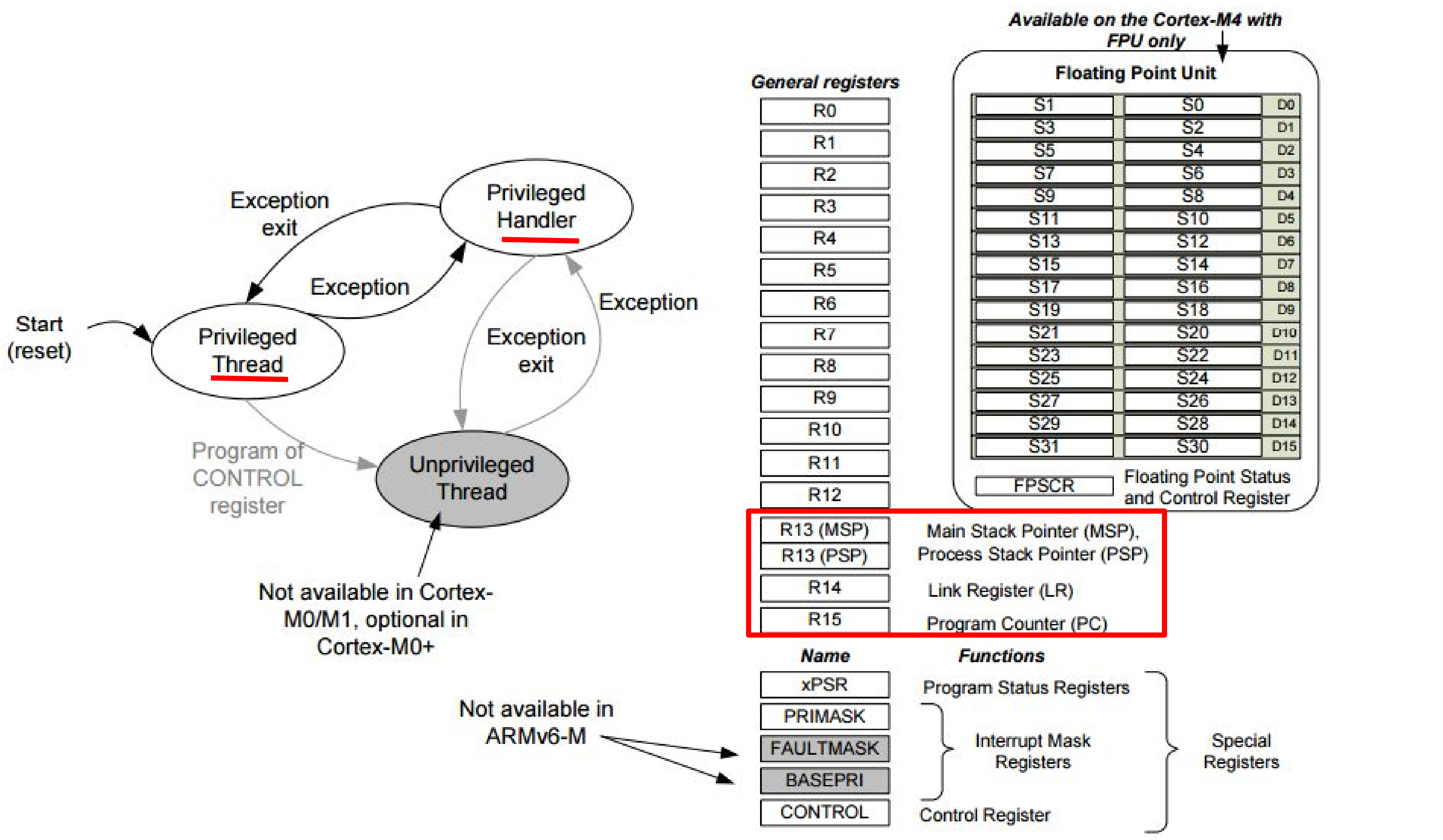


	Cortex-M0	Cortex-M0+	Cortex-M3	Cortex-M4	Cortex-M7
Instruction set architecture	ARMv6-M Thumb, Thumb-2	ARMv6-M Thumb, Thumb-2	ARMv7-M Thumb, Thumb-2	ARMv7-M Thumb, Thumb-2, DSP, FP (SP)	ARMv7-M Thumb, Thumb-2, DSP, FP (1. SP or 2. SP+DP)

Cortex-M Instruction Set support



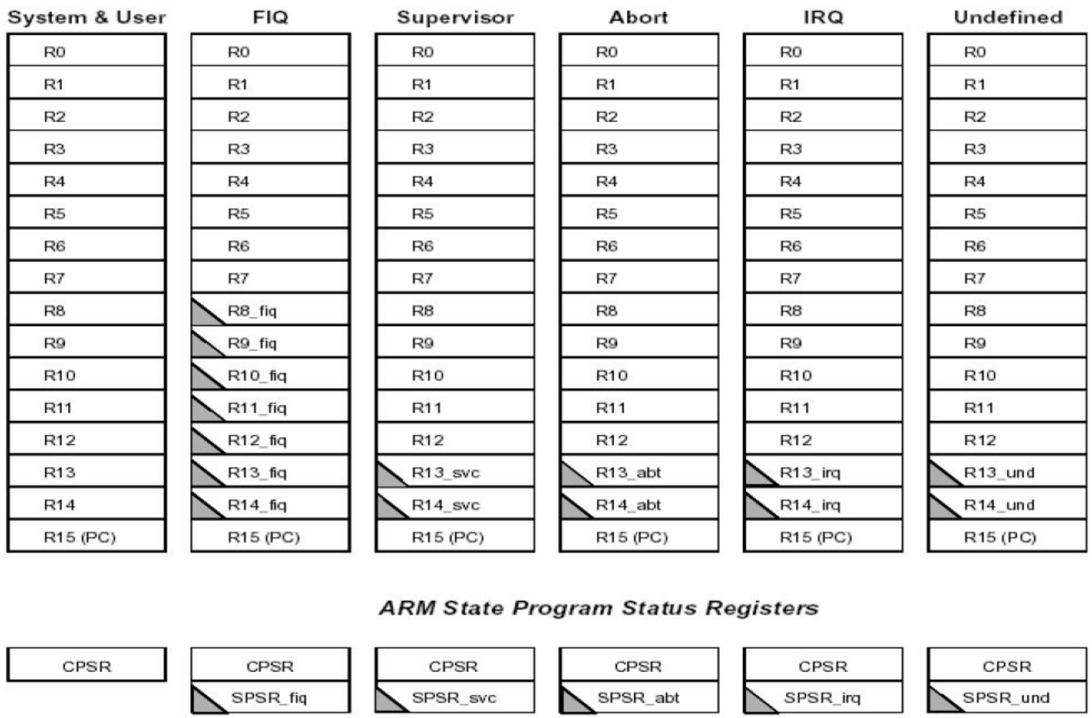
Programmer's model



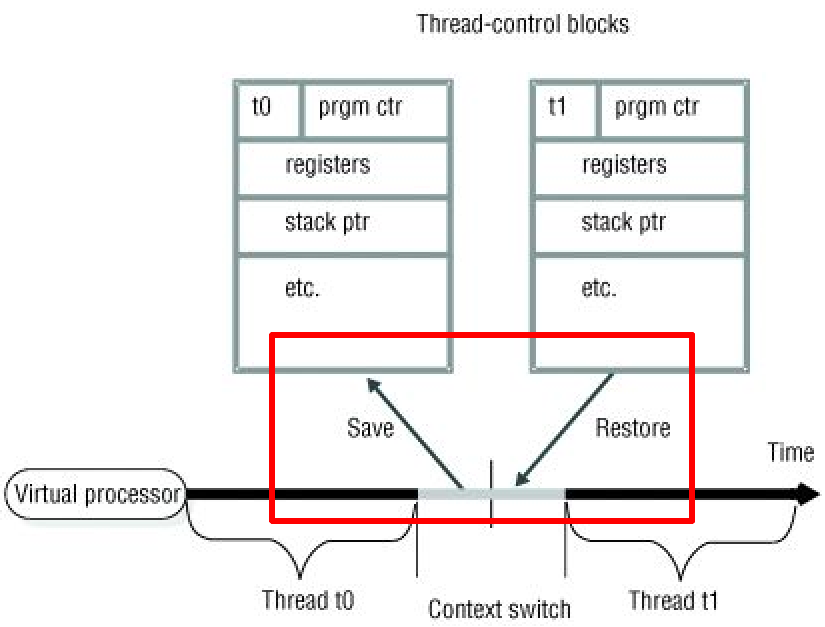
Real Time OS ?

Context

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



Context Switching



출처

스케줄러

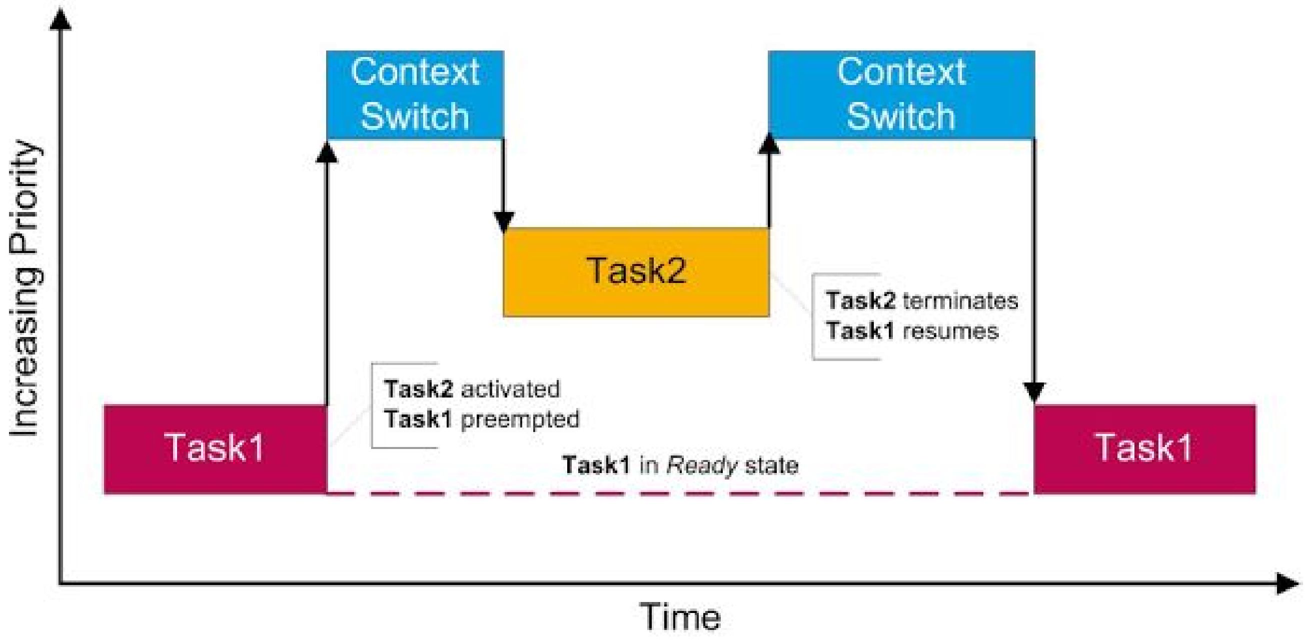
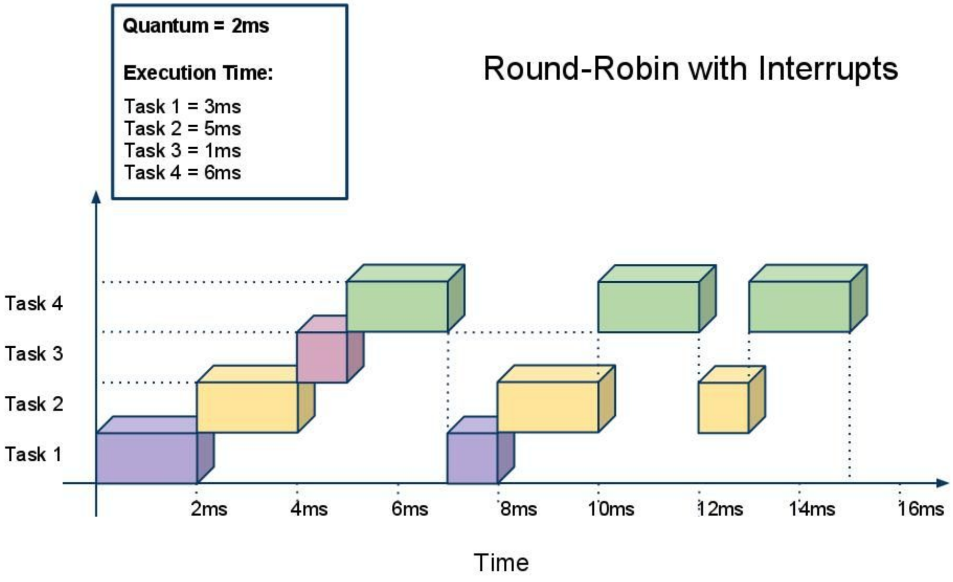


Figure 4.1: Preemptive scheduling of tasks

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

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

성능 비교

- 데이터 처리 속도

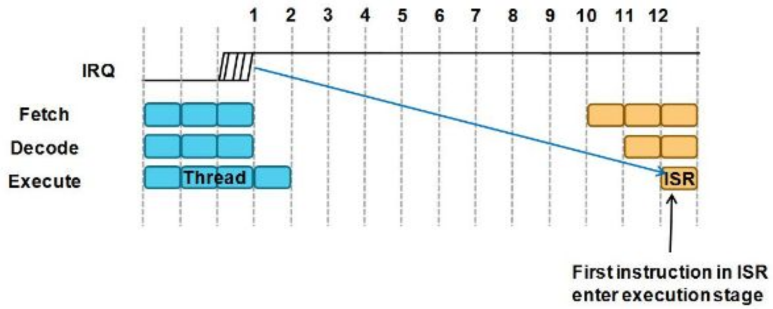
	Dhrystone DMIPS/MHz (v2.1) – official	Dhrystone DMIPS/MHz (v2.1) – full optimization	Coremark/MHz (v1.0)
Cortex-M0	0.84	1.21	2.33
Cortex-M0+	0.94	1.31	2.42
Cortex-M3	1.25	1.89	3.32
Cortex-M4	1.25	1.95	3.40
Cortex-M7	2.14	2.55	5.01

- Interrupt Latency

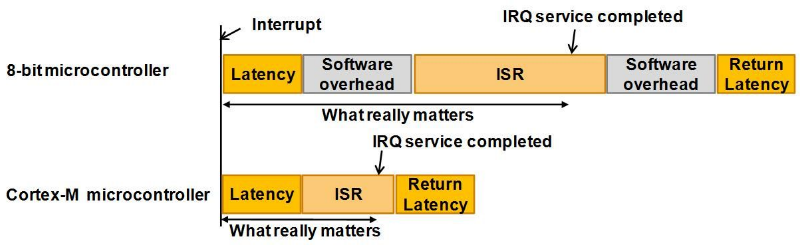
	Interrupt latency (number of clock cycles)
Cortex-M0	16
Cortex-M0+	15
Cortex-M3	12
Cortex-M4	12
Cortex-M7	Typically 12, worst case 14

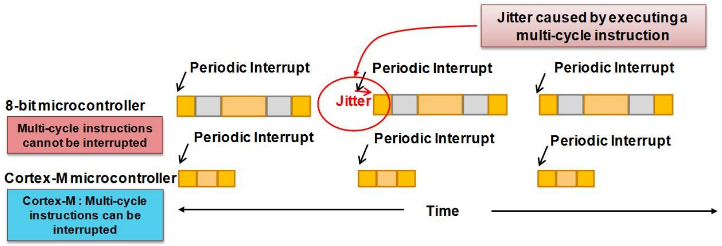
Interrupt Latency

- Interrupt Latency ?



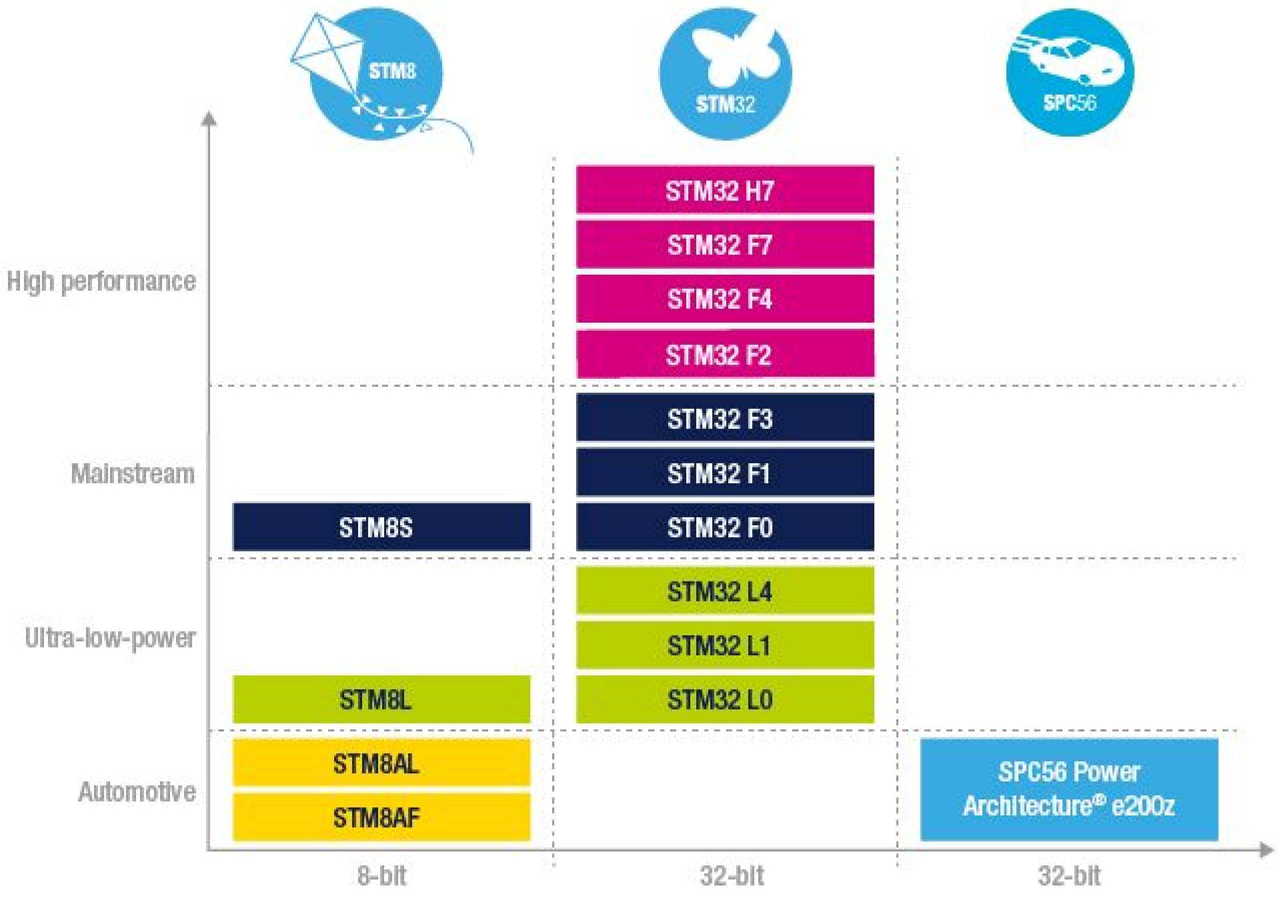
- 기존 마이크로 컨트롤러와 비교





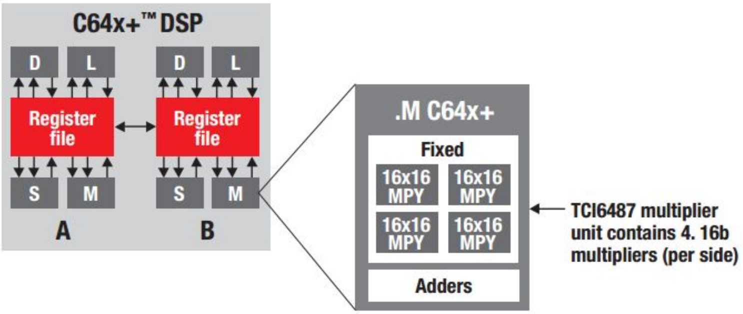
출처 : Link

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

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

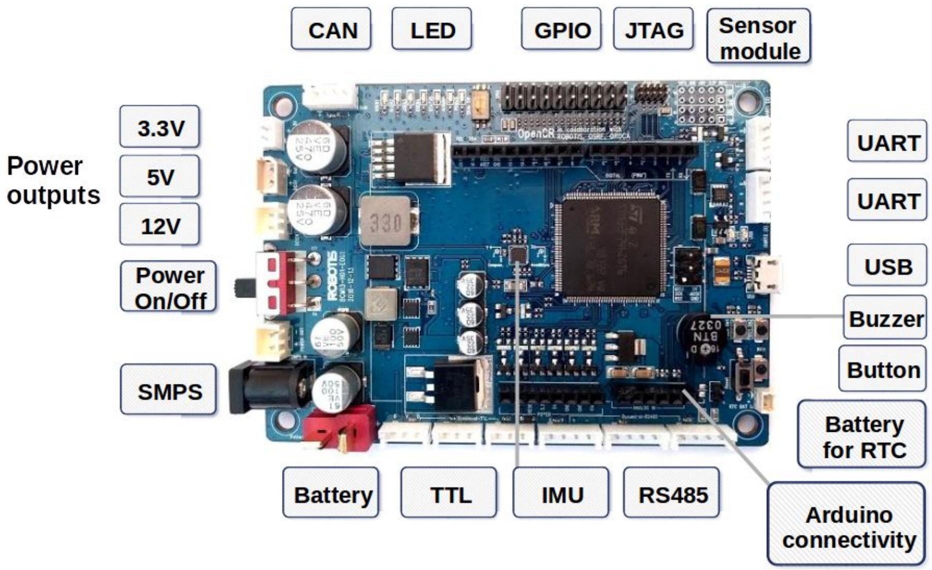
```
   MVKL L1DCC, A0     ; \
|| MVKL L1PCC, B0     ;  |__ Generate L1DCC pointer in A0
   MVKH L1DCC, A0     ;  |   and L1PCC pointer in B0
|| MVKH L1PCC, B0     ; /
|| MVK 1b, A1         ; \___ OPER encoding for 'freeze'
|| MVK 1b, B1         ; /    in both A1 and B1.
   STW A1, *A0        ; Write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

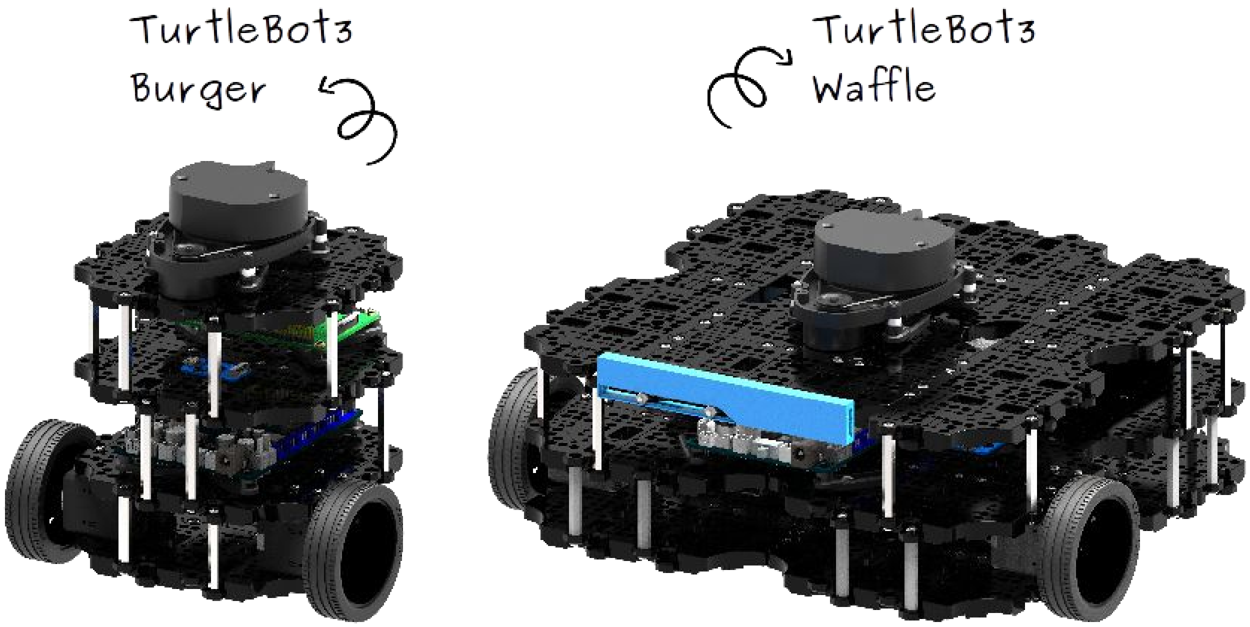
- STM32F746ZGT6 216Mhz, Cortex-M7, 1MB Flash, 320KB SRAM
- 아두이노 우노 핀 헤더
- 아두이노 IDE 개발환경 지원
- 다이나믹셀/올로/UART/CAN 인터페이스
- 배터리 입력 및 전원 출력(12V/5V/3.3V)



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