

DK-AVR 기반 RTOS 교육



안녕하세요, 유명환입니다. :-)

Silver

Microsoft



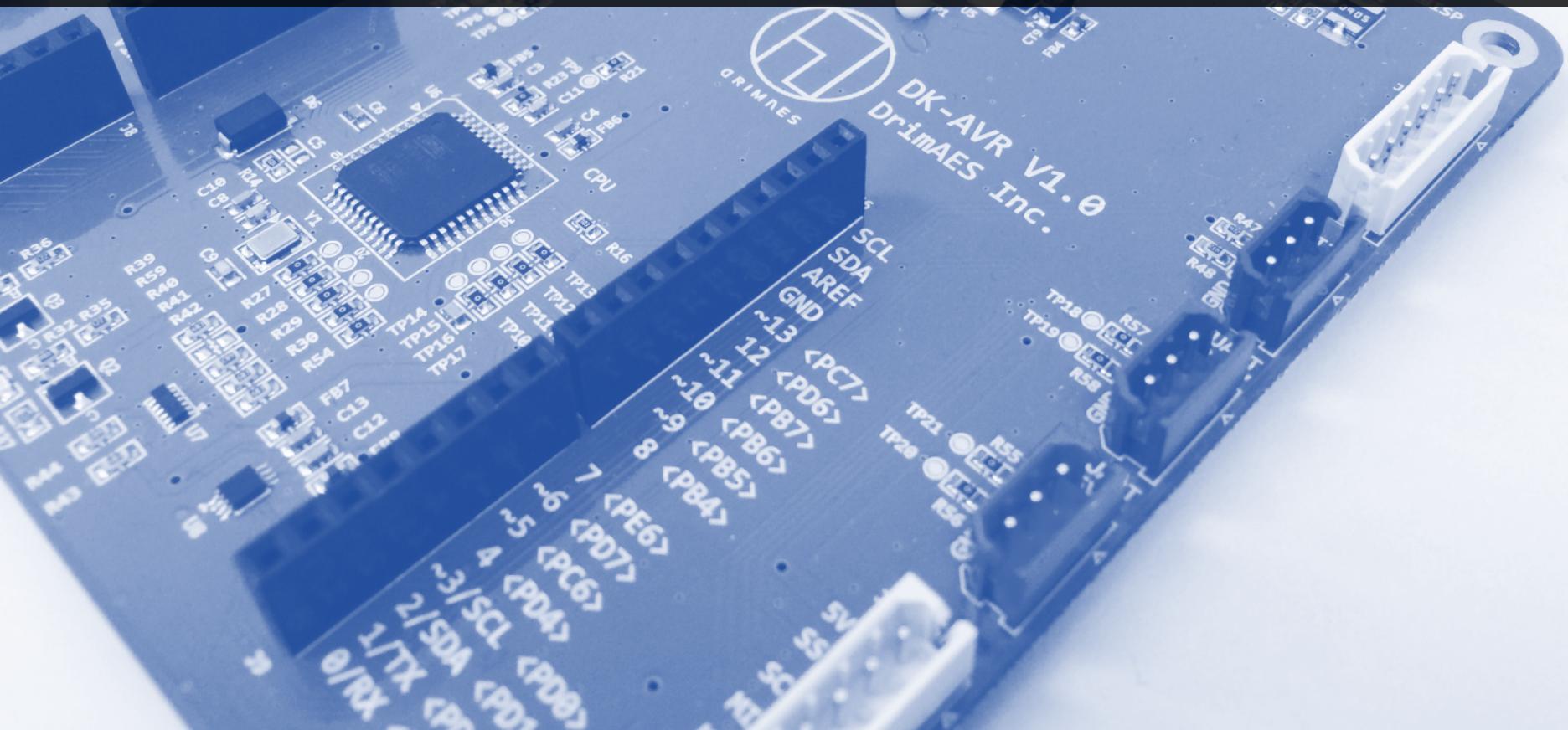
funfun.yoo@gmail.com

- 엑세스 주식회사, 연구소장(CTO)
- 정부통합전산센터, 클라우드 기술위원
- 대구경북과학기술원(DGIST), 산학기술협력 멘토
- 네이버 D2 Startup Factory 기술 파트너
- 오픈스택 한국 커뮤니티, 네트워크 분과장
- 페이스북

[오픈소스포럼], [만물상(IoT)],

[오픈소스 데이터센터 포럼] 운영

교육 내용



교육 내용

왜 AVR 인가?

왜 DK-AVR 인가?

임베디드 플랫폼

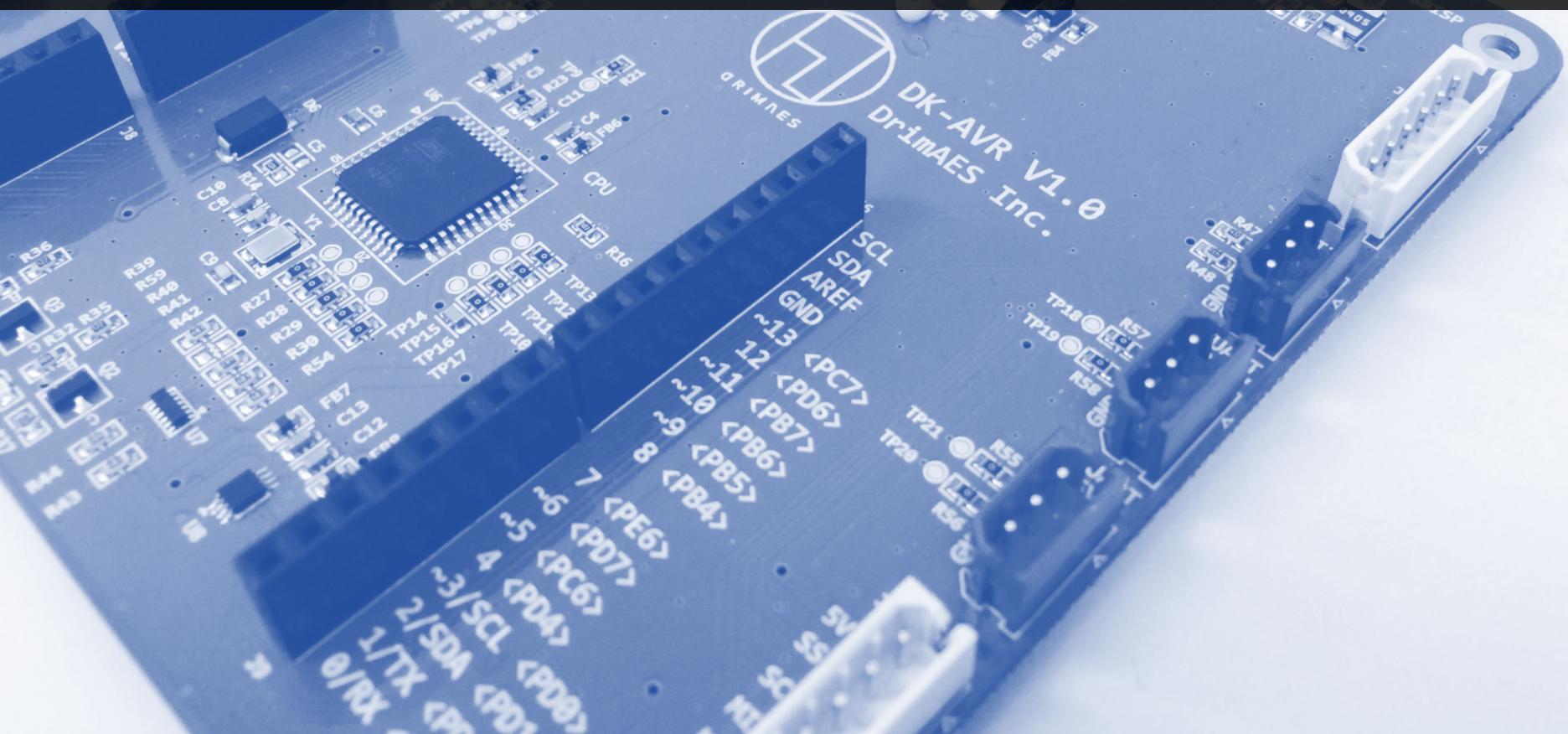
마이크로 프로세서

운영체제

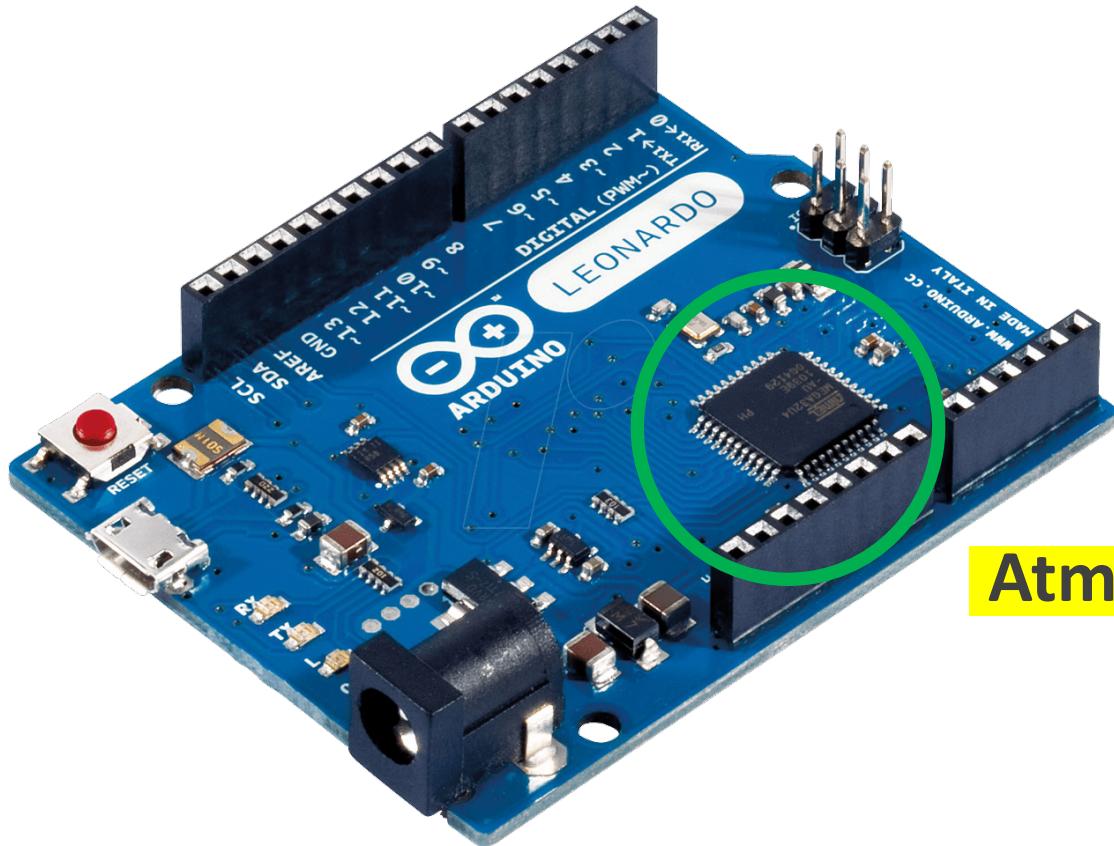
DK-AVR 기반 RTOS 포팅 실습



왜 AVR 인가?



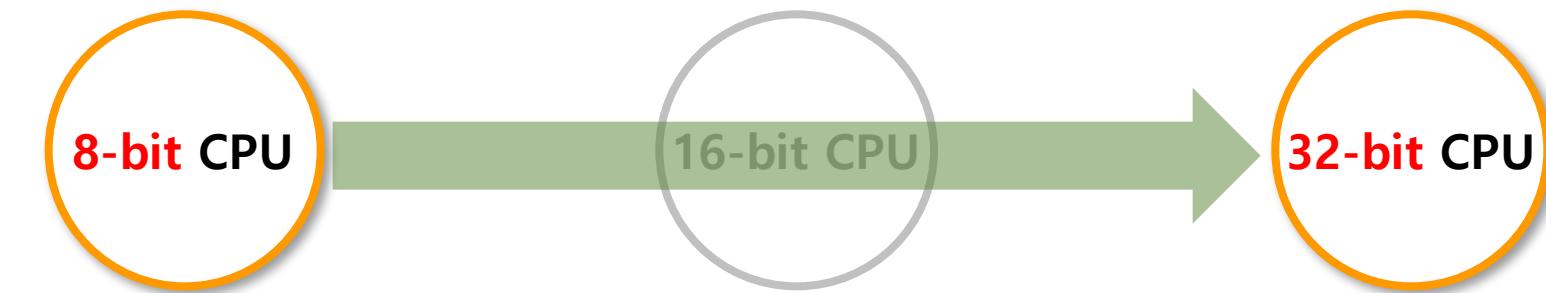
왜 AVR 인가?



Atmel AVR ATmega32U4



왜 AVR 인가?



8051 Core

8비트 기반의 SoC(System on Chip) : ex) CC2430 = 8051 + CC2420

PIC Core

AVR Core

C 언어를 가장 잘 지원해주는 프로세서!

8비트 범용 프로세서 시장 장악

--> ISP(In-System Programming), JTAG support

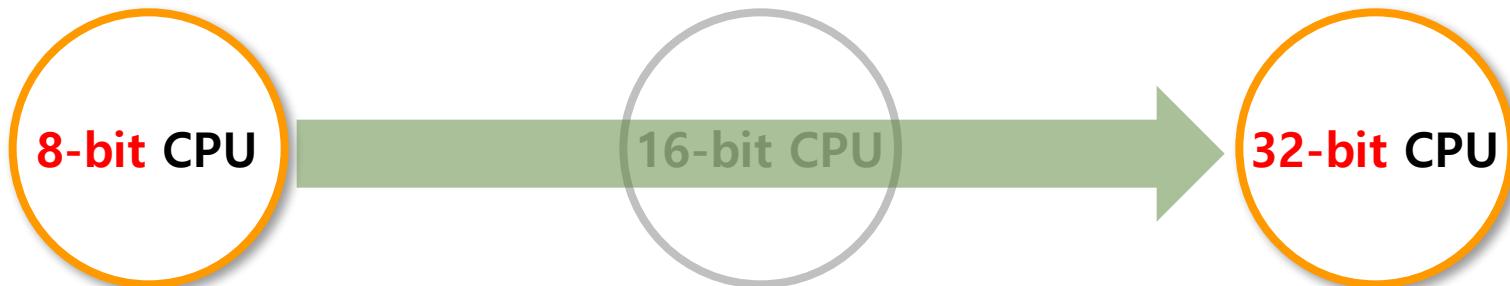
--> Open Source Cross-Compiler(GCC)

--> RTOS(Real-Time OS) support :

ex) uC/OS-II, TinyOS, Nano Qplus, FreeRTOS

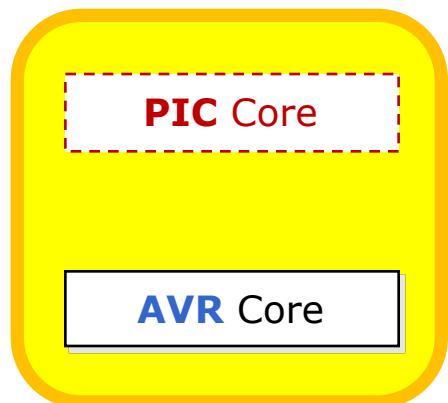


왜 AVR 인가?



8051 Core

8비트 기반의 SoC(System on Chip) : ex) CC2430 = 8051 + CC2420



이제는 한식구!!!



왜 AVR 인가?

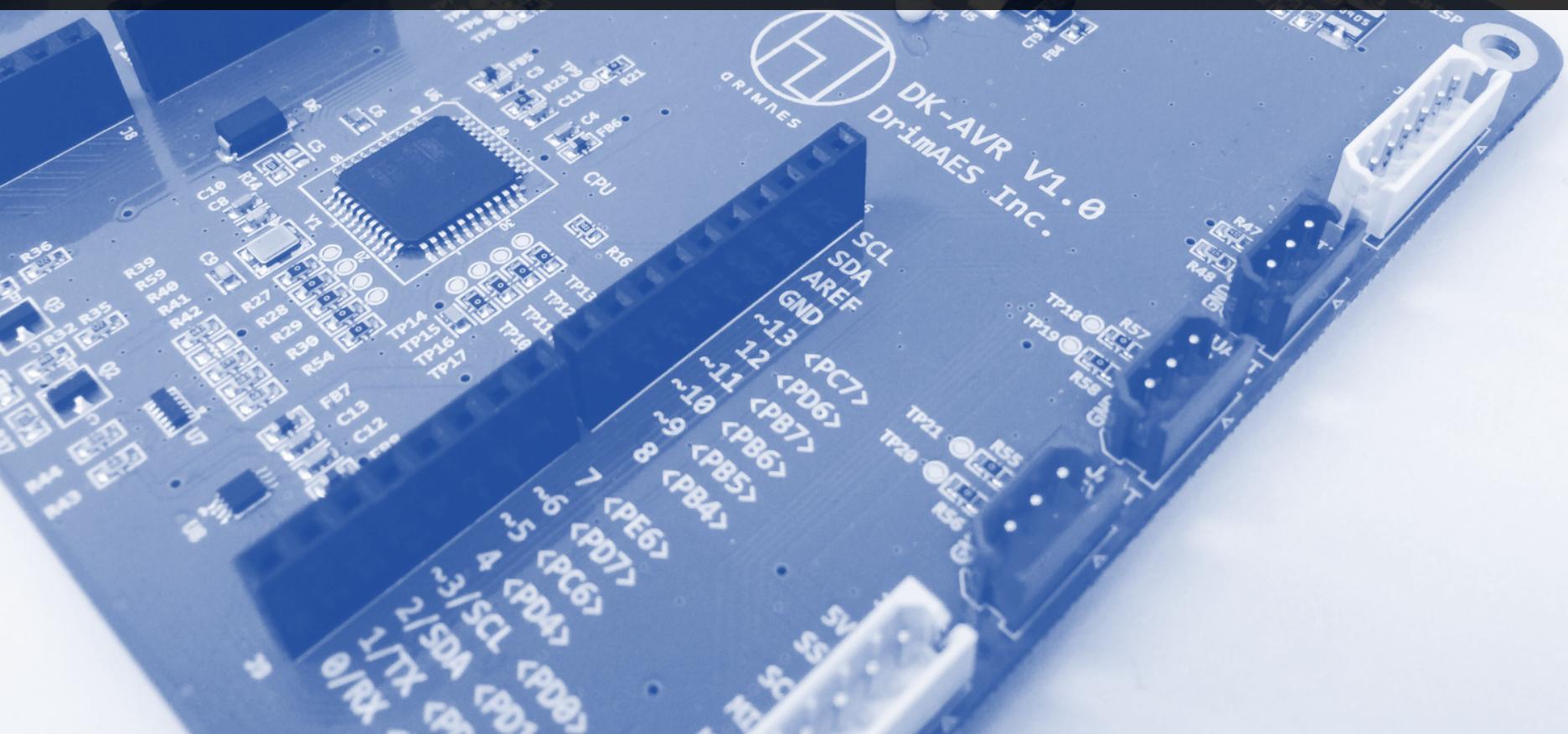
- 개발 환경 구축 비용이 거의 들지 않는다!
- 컴파일러가 매우 다양하다!
Ex) avr-gcc, IAR Systems EW AVR, HP InfoTech CodeVisionAVR, ImageCraft ICCAVR
- 메모리에 대한 부담감이 없다!
- 이미 아두이노(Arduino)로 많이 익숙해져 있다!

※ AVR 개발 시 주의 사항!!!

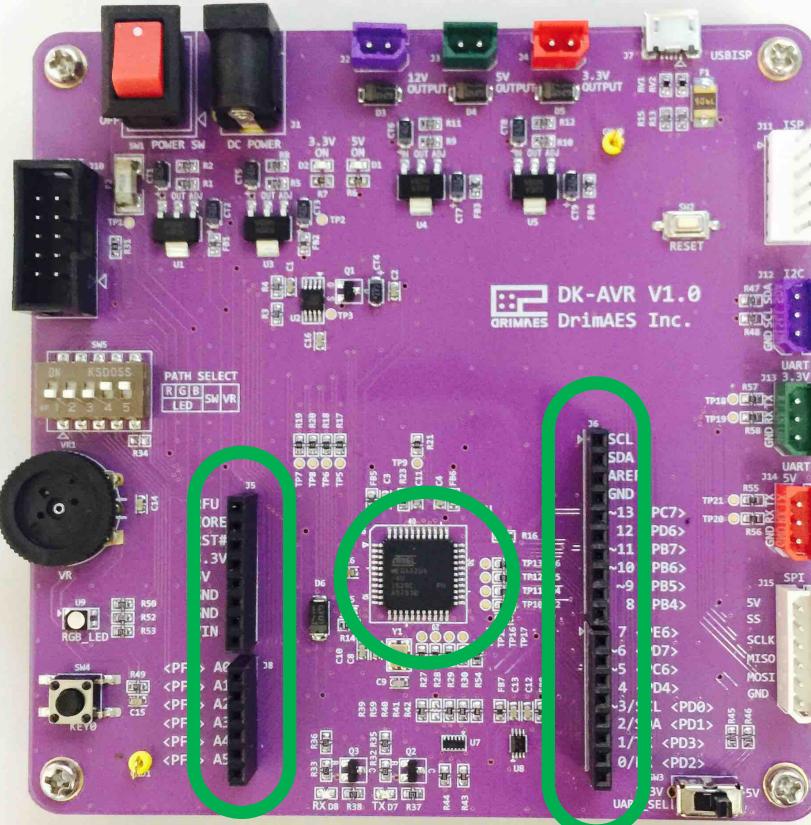
CodeVisionAVR 컴파일러처럼 Only AVR 컴파일러는 사용하지 말 것!!!



왜 DK-AVR 인가?



왜 DK-AVR 인가?

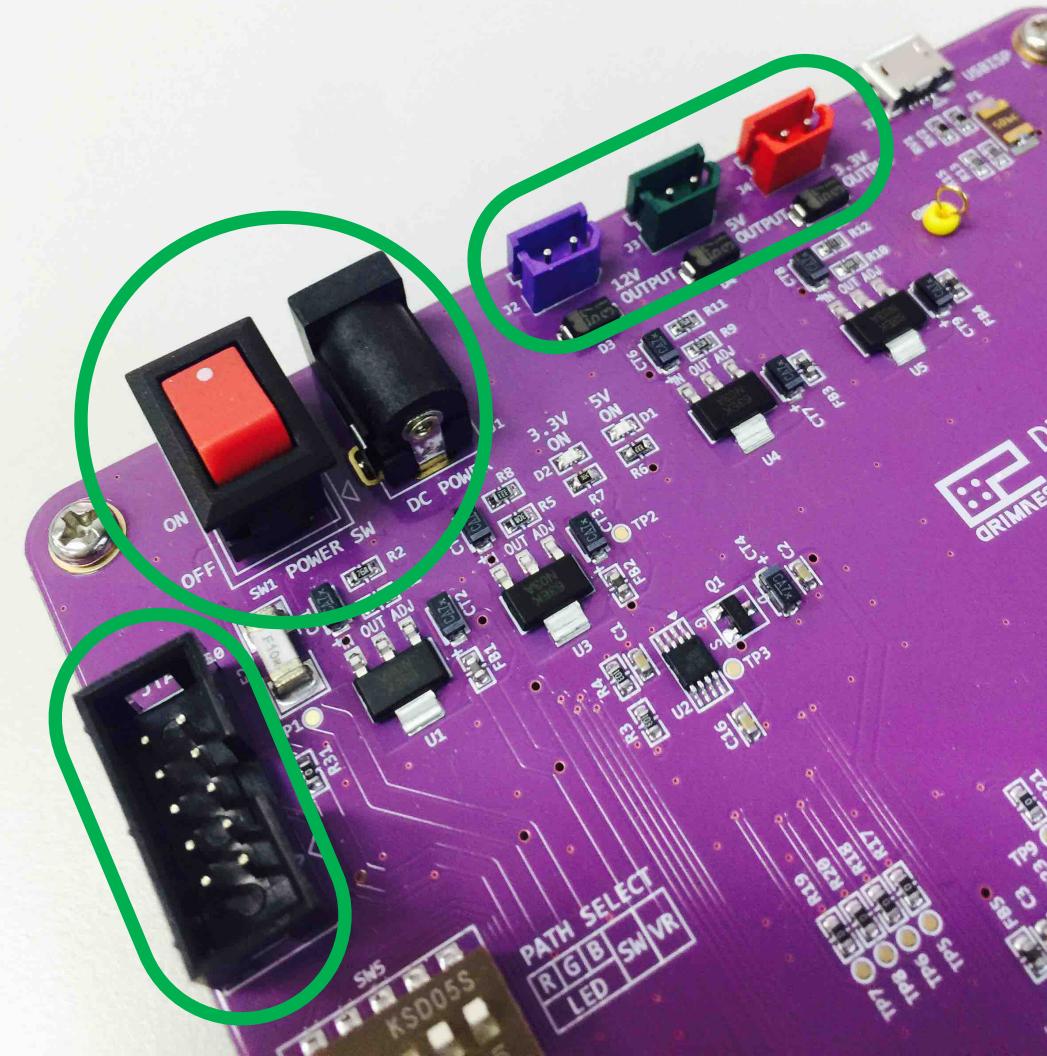


Arduino & AVR Firmware 동시 교육 가능



왜 DK-AVR 인가?

외부 전원 입력 & 외부 전원 출력 가능 (12 / 5 / 3.3V)



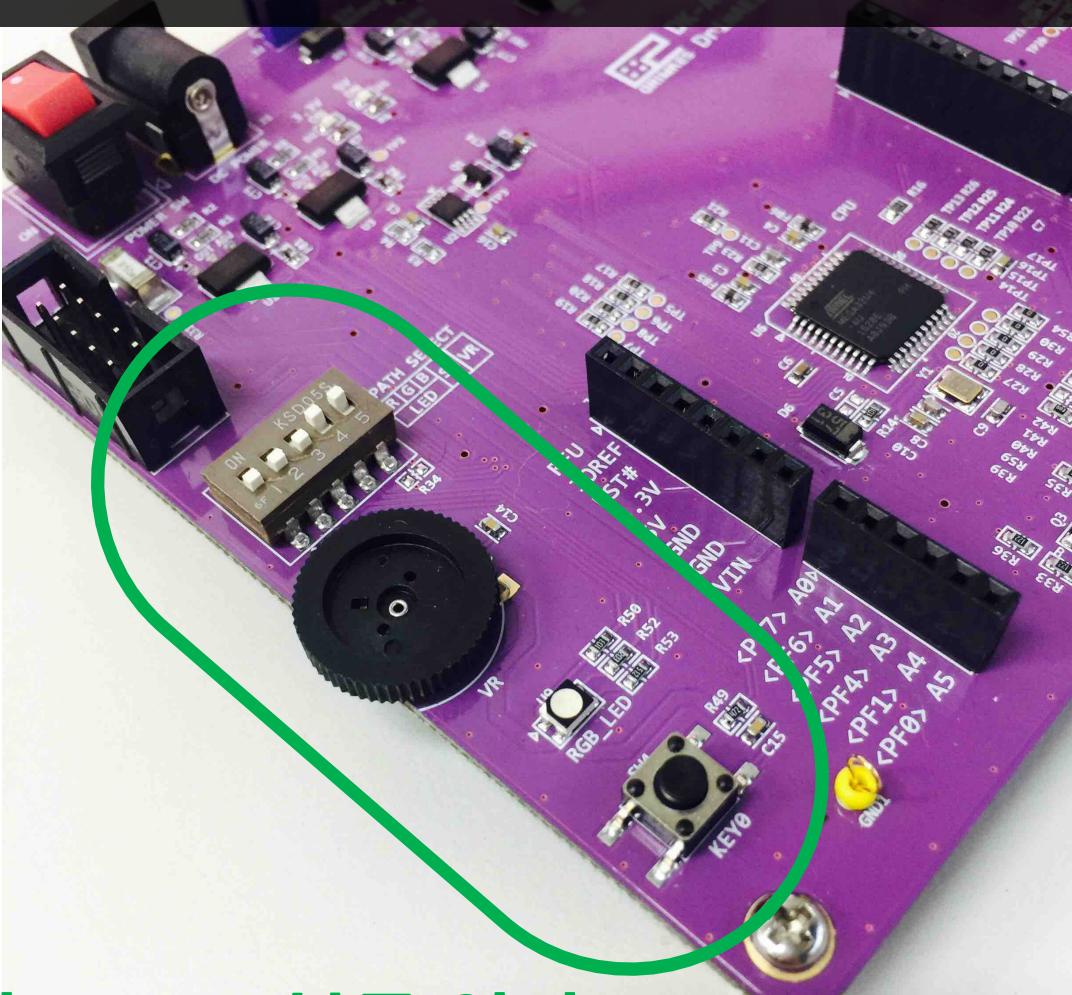
왜 DK-AVR 인가?



다양한 시리얼 통신
교육 가능



왜 DK-AVR 인가?

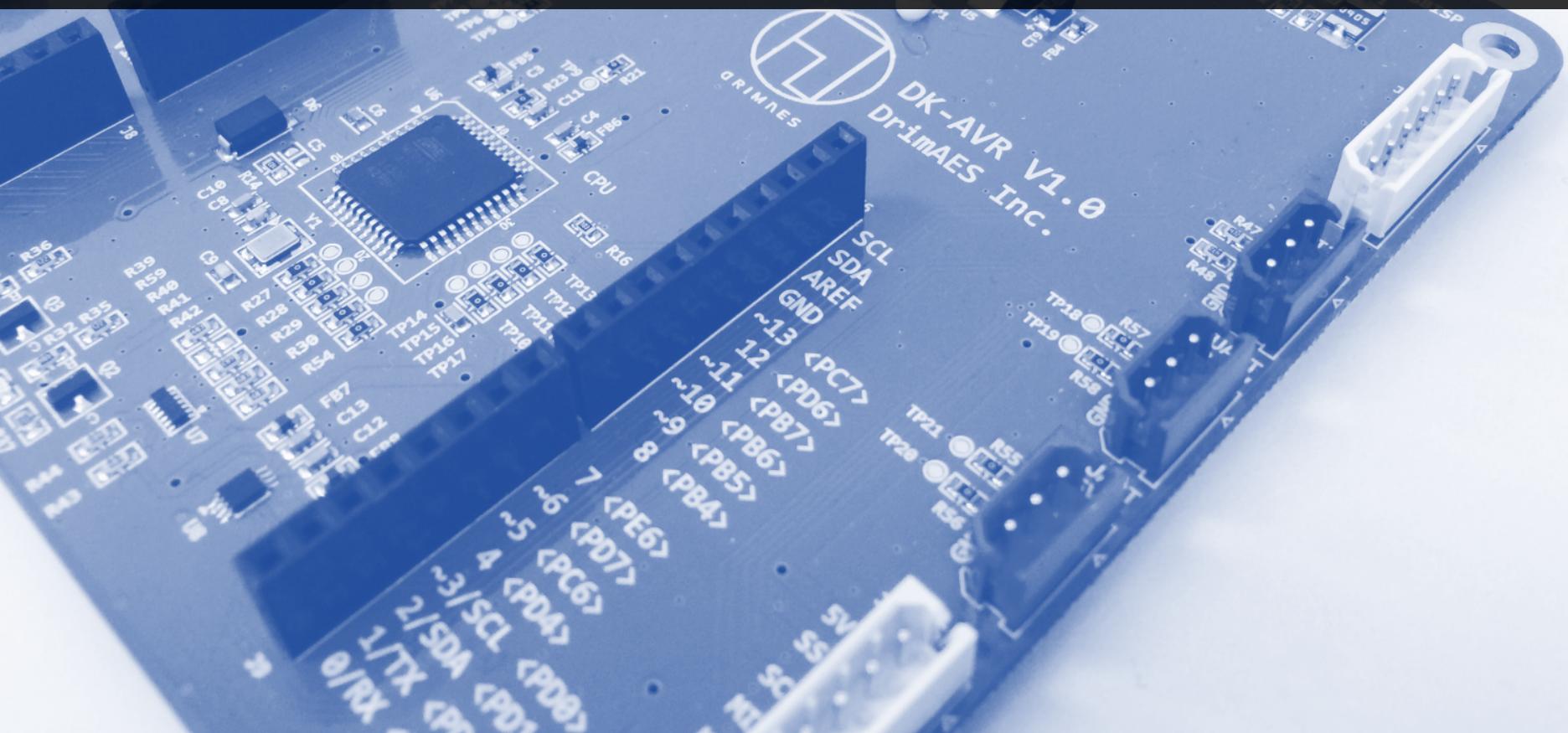


별도의 확장 보드 & 부품 없이도

다양한 임베디드 SW 교육 가능



임베디드 플랫폼



임베디드 플랫폼

Embedded Platform

CPU

OS

Tool



임베디드 플랫폼

Android Platform

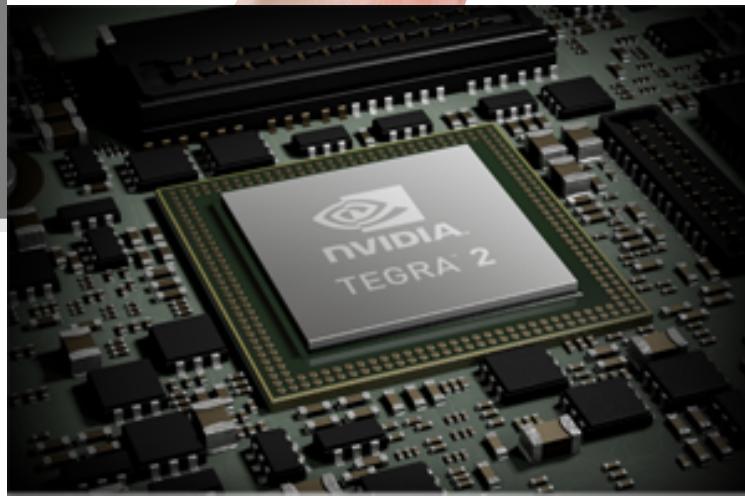
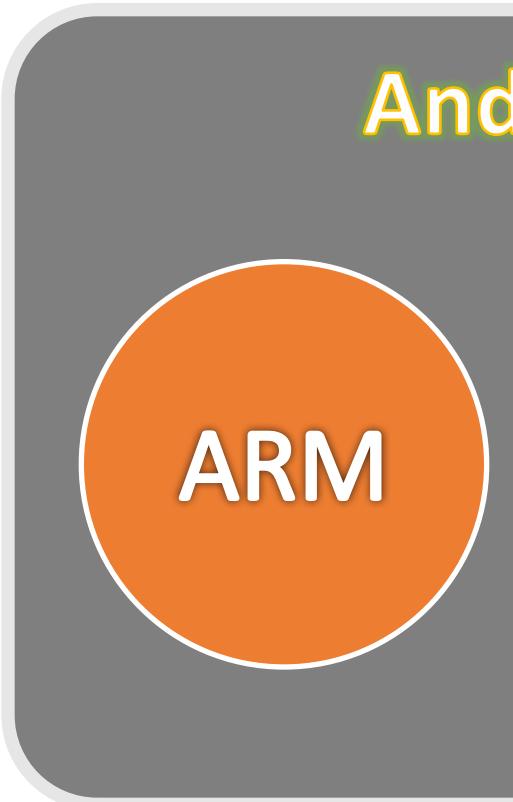
ARM

Linux

Eclipse,
Ubuntu



임베디드 플랫폼



임베디드 플랫폼



인프라웨어 “피처폰이 안드로이드폰으로 변신”



느낌을 그대로 재현하는 기술
다쏘시스템 소프트웨어가
실현할 수 있는 꿈입니다.



DASSAULT SYSTEMES The 3DEXPERIENCE

400MHz짜리 ARM 프로세서로 안드로이드를 돌린다? 1GHz 듀얼코어 프로세서로도 시원스럽게 돌아가지 않는 것이 안드로이드 운영체제다. 인프라웨어는 400~500MHz 수준의 피처폰에서 돌아가는 독자적인 안드로이드를 개발했고 이 운영체제는 이미 7개 회사를 통해 제품으로 만들어지고 있다.

'풀라리스 스마트폰 스위트'라고 부르는 이 플랫폼 기술을 적용하면 구닥다리 취급받는 피처폰에서도 30달러 정도의 비용 투자만으로 안드로이드 스마트폰을 제조할 수 있다. 결과적으로 100달러 내외의 스마트폰이 태어나는 셈이다.

글쓴이



최호

allove@bloter.n



발행일

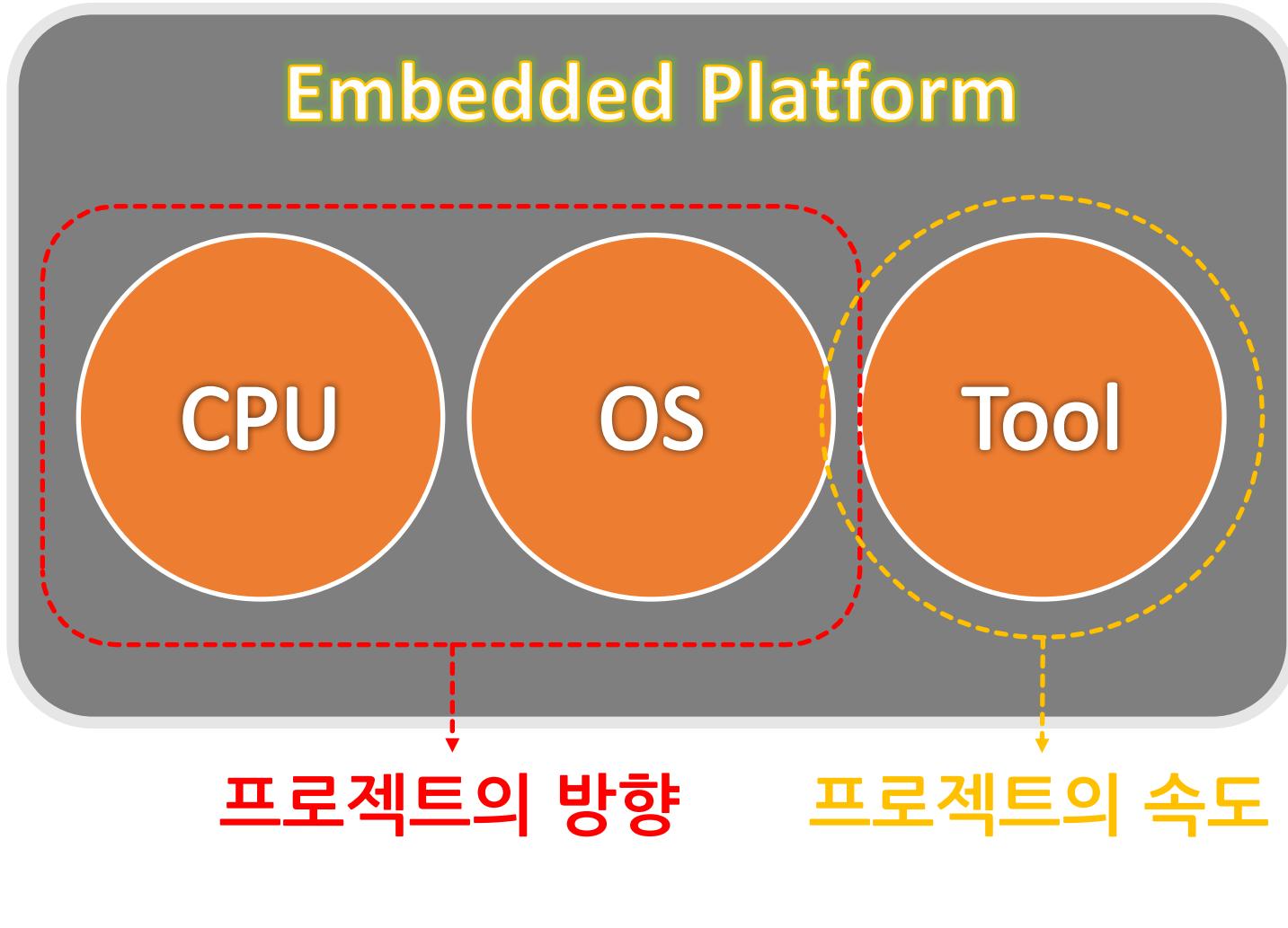
Polaris Smartphone Suite

Reference Device SPEC

Display 3.5 Inch (480x320)
Processor MediaTek MT6257 (ARM11, 520MHz)
Memory SDRAM : 256MB, NAND 512MB



임베디드 플랫폼



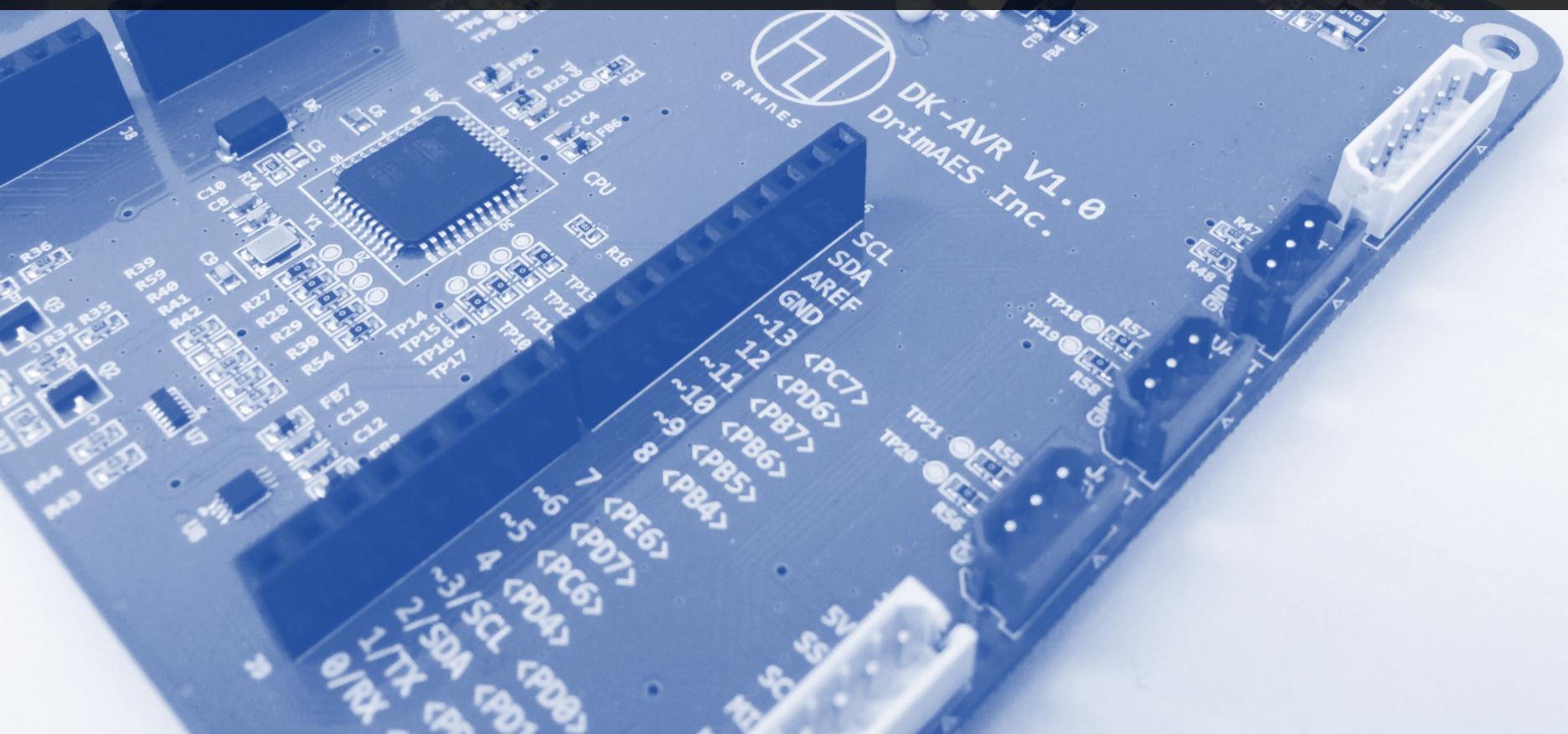
임베디드 플랫폼

임베디드 플랫폼

= 개발에 대한 제약 사항



マイクロ プロセッサー



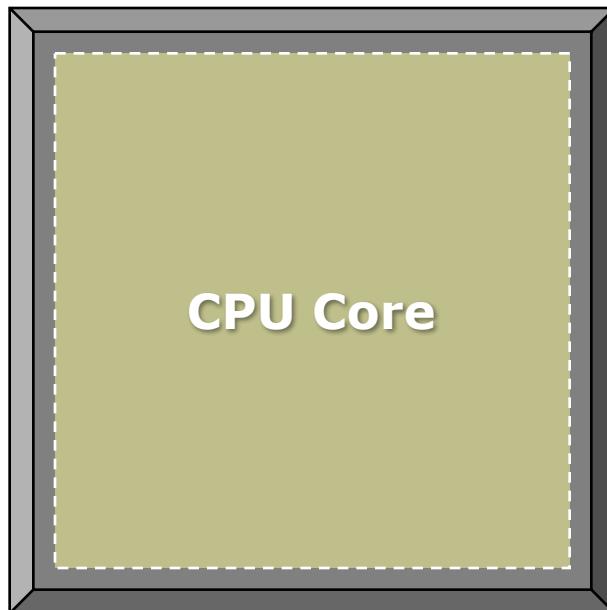
마이크로 프로세서



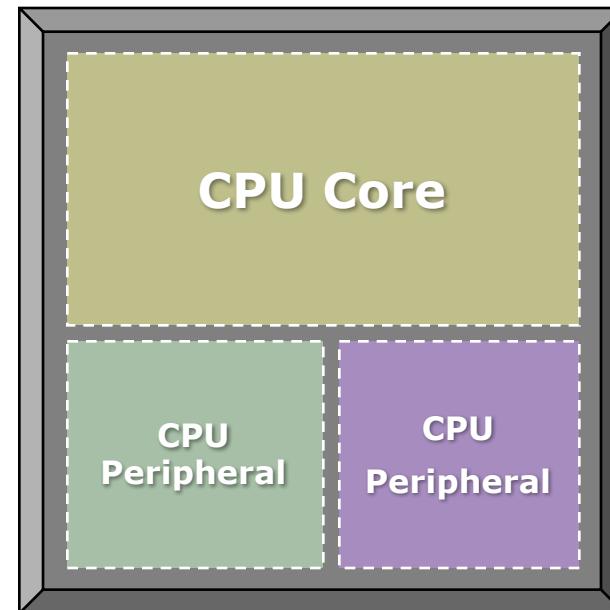
マイクロ プロセ서

CPU = CPU Core + CPU Peripheral (Controller)

- CPU Core : 연산 처리를 담당
- CPU Peripheral (Controller) : (연산 처리 결과에 따른) H/W 제어를 담당



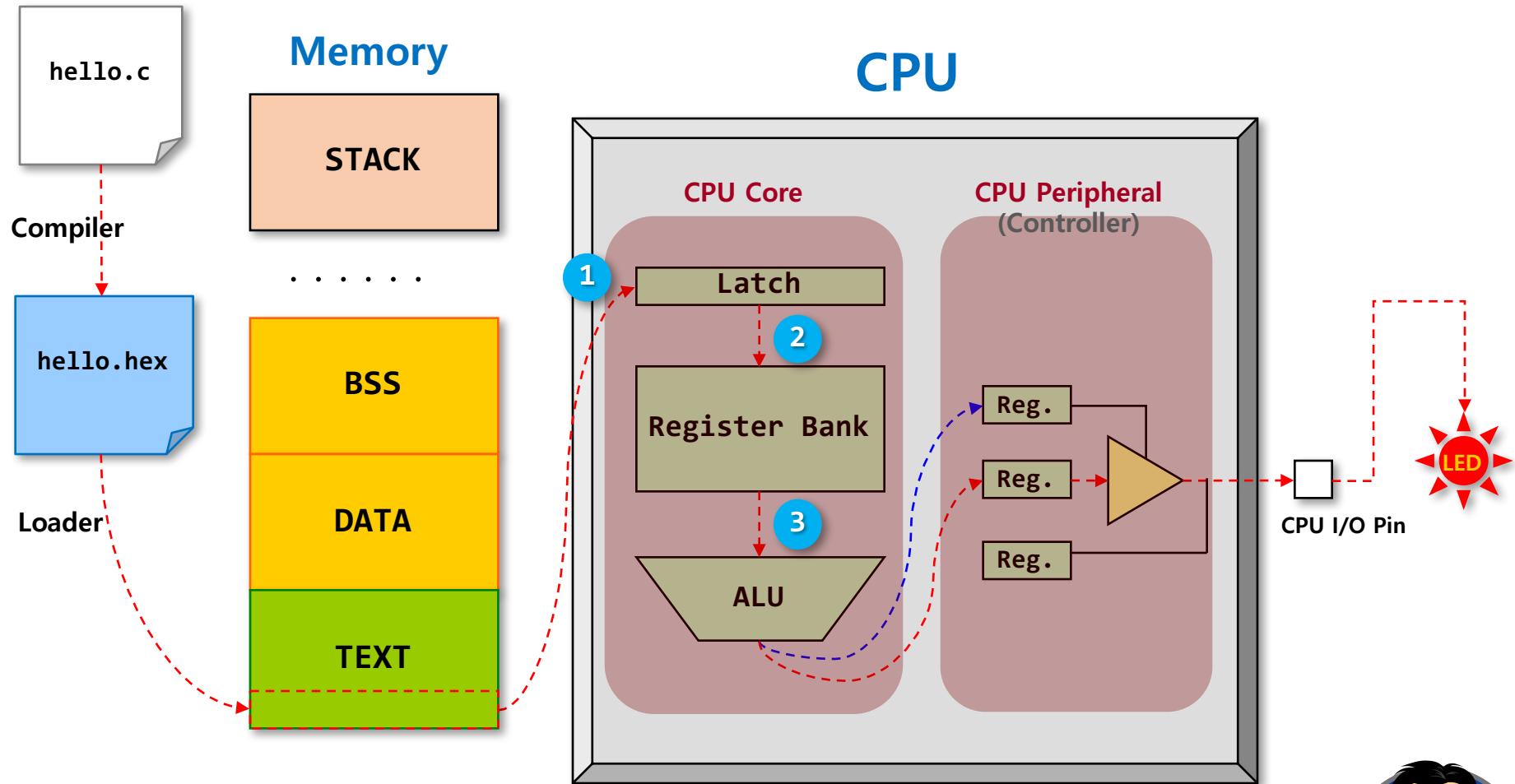
마이크로프로세서(Microprocessor)



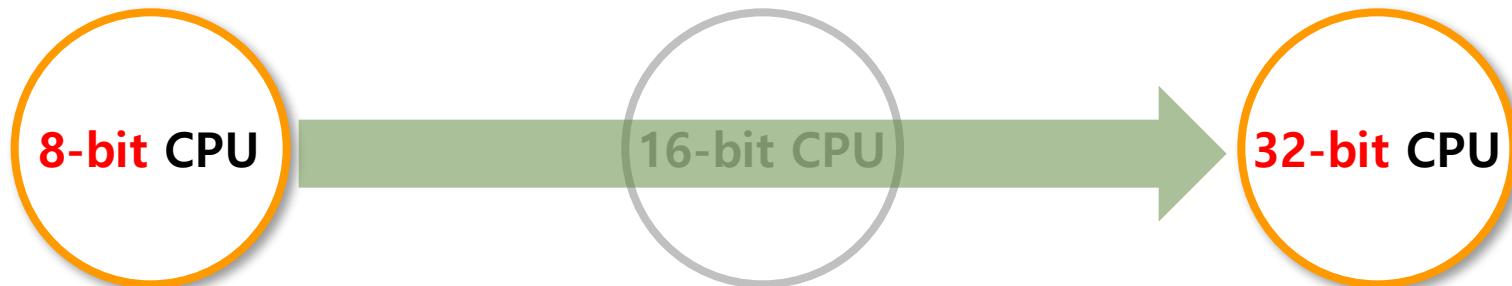
마이크로컨트롤러(Microcontroller)



マイクロ プロセッサー



マイクロ プロセ서



8051 Core

8비트 기반의 SoC(System on Chip) : ex) CC2430 = 8051 + CC2420

PIC Core

AVR Core

C 언어를 가장 잘 지원해주는 프로세서!

8비트 범용 프로세서 시장 장악

--> ISP(In-System Programming), JTAG support

--> Open Source Cross-Compiler(GCC)

--> RTOS(Real-Time OS) support :

ex) uC/OS-II, TinyOS, Nano Qplus, FreeRTOS



マイクロ プロセ서

ARM Core

PPC Core

MIPS Core

x86 Core

ARM7 Core

16비트를 대신할 저가형 32비트 솔루션

Cortex-M3

ARM9 Core

- 저전력, 확장성 등을 앞세워 32비트 시장을 점령
- PC와 마찬가지로 멀티코어(Multi-Core) 등장

ARM11 Core

- PC와 유사한 고 성능 + DSP 기능 추가
- 범용 운영체제(OS) 시장의 견인차 역할 : ex) iPhone, RA

Cortex-A8

AU1200 CPU



AU1250 CPU

- PMP(Portable Multimedia Player)
- STB(Set Top Box)



マイクロ 프로세서

ARM cores

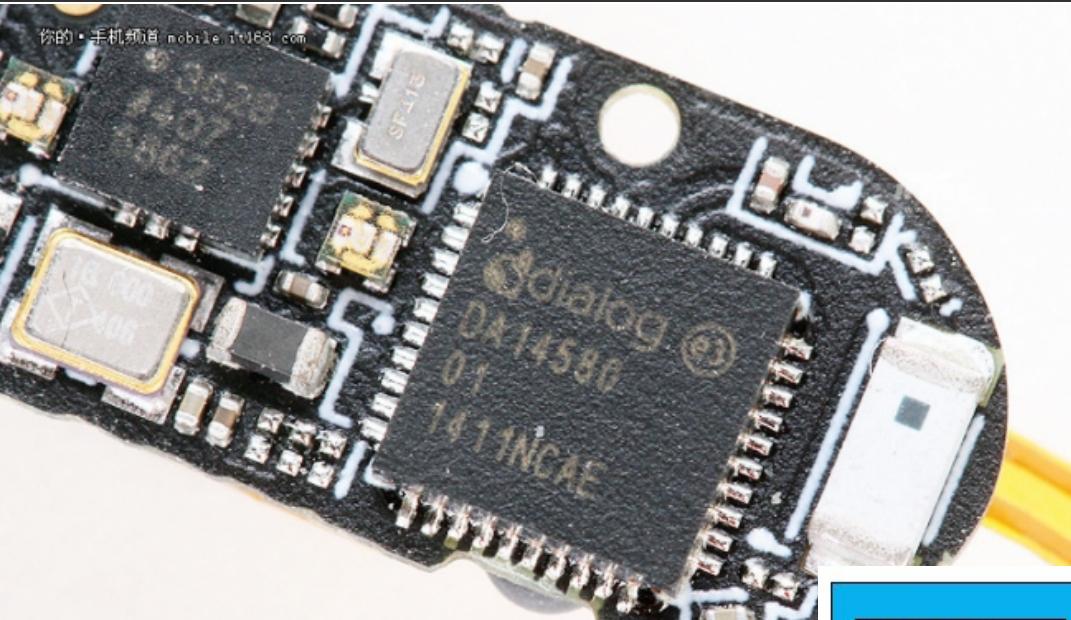
Main article: [List of ARM microprocessor cores](#)

Architecture	Family
ARMv1	ARM1
ARMv2	ARM2 , ARM3 , Amber
ARMv3	ARM6 , ARM7
ARMv4	StrongARM , ARM7TDMI , ARM8 , ARM9TDMI , FA526
ARMv5	ARM7EJ , ARM9E , ARM10E , XScale , FA626TE , Feroceon , PJ1/Mohawk
ARMv6	ARM11
ARMv6-M	ARM Cortex-M0 , ARM Cortex-M0+ , ARM Cortex-M1
ARMv7	ARM Cortex-A5 , ARM Cortex-A7 , ARM Cortex-A8 , ARM Cortex-A9 , ARM Cortex-A15 , ARM Cortex-R4 , ARM Cortex-R5 , ARM Cortex-R7 , Scorpion , Krait , PJ4/Sheeva , Swift
ARMv7-M	ARM Cortex-M3 , ARM Cortex-M4
ARMv8-A	ARM Cortex-A53 , ARM Cortex-A57 <small>[23]</small> , X-Gene , Denver

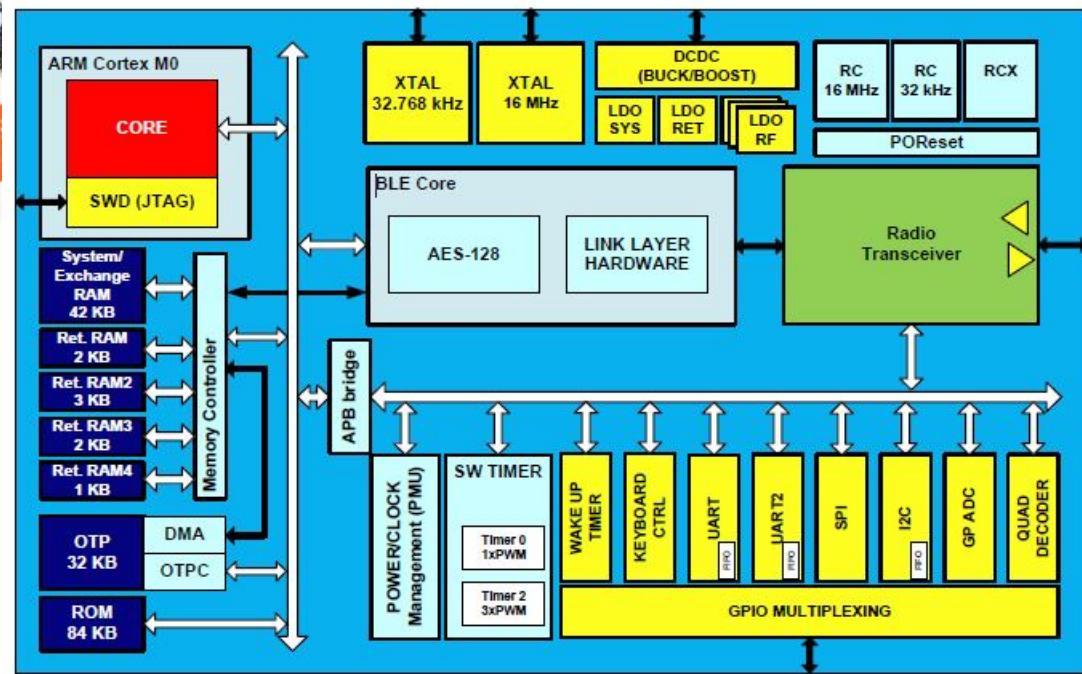
A list of vendors who implement ARM cores in their design is provided by ARM. [24]



マイクロ プロセ서



Dialog DA14580超低功耗处理器，集成蓝牙4.0连接强大计算能力，省电则成为手环选择芯片的第一要素



マイクロ プロセッサー

인텔이 최근 저전력 프로세서인 '인텔 센트리노 아틀 프로세서'를 내놓으면서 모바일인터넷기기(MID)용 칩 시장을 선점한 ARM과 한판 대결을 시작했다. 인텔은 저전력과 속도, 그리고 PC분야에서 구축해 온 강력한 애플리케이션으로 물아붙이겠다는 생각이나 ARM은 아쉽다고 맞받아쳤다.



모바일인터넷기기용 칩 시장

'인텔-ARM의 결투'

인텔 신제품 '아톰' 출시…성능에 자신

ARM "인텔 SW 구현엔 시간 더 필요"

과거 스트롱ARM 계열 솔루션을 갖고 있던 인텔은 PC아키텍처로 무장, 새로운 패러다임을 예고했다.

인텔은 모바일 분야에는 자타가 공인하는 선두 업체이나 인텔의 등장으로 위협을 받게 됐다. 두 회사의 경쟁으로 시장 확대도 예상됐다.

◇인텔 와 MID시장에 가세했

나=MID는 지난달 스페인 바르셀로나에서 열린 WMC 2008에서 기대를 한 품에 뱉었다. 지난 해 인터넷 접속 도구 부문에서 모바일 기기와 PC를 앞질렀다는 조사결과도 나왔다.

MID 같은 모바일 인터넷 기기가 급부상했다. 검색서비스업체인 구글이 MID의 일종인 구글폰을 내놓은 것도 주도권을 끌기 위함이다.

인텔 역시 PC(노트북PC 포함) 전영역에서 구축한 강력한 세력을 바탕으로 MID분야에 진출함으로써 시장을 선도해 나가겠다는 의지가 높다. 노트북PC 분야에서 군림해 온 인텔과 휴대폰 등 모바일 기기에서 힘을 죽

적한 ARM이 MID에서 맞닥뜨린 셈이다.

◇소비전력과 속도, 가격 경쟁 =인텔 아톰이 내세우는 것은 전력소비와 속도, 가격이다. 아톰은 하이-к 메탈게이트 기술과 45 나노 미세 공정을 적용했다. 칩 크기는 '25mm×25mm'이면서 전력소모는 0.6~2.5W이다. 속도는 사용자의 요구에 따라 1.8GHz까지 높일 수 있다.

선 말로니 인텔 수석 부시장은 아톰을 "세계에서 가장 작은 트랜지스터로 만들어진 인텔의 가장 작은 프로세서며 새로 등장할 MID에서 일청난 인터넷 경험을 할 수 있게 해줄 정도로 강력하다"고 강조했다.

ARM 측은 이를 반박했다. 이 회사는 "인텔이 소개하는 전력소모면 0.6~2.5W가 아톰과 애플리케이션 프로세서 전체를 이르는 것인지 확인해 봄야 한다"며 "소비전력 측면에서 ARM 칩이 뒤지지 않는다"고 강조했다.

아톰 칩은 전체 25mm×25mm의 크기 중 코어는 45 나노 공정을 적용해 9mm×9mm를 실현했지만 ARM코어의 크기는 3mm×3mm(65 나노 공정) 수준이다.

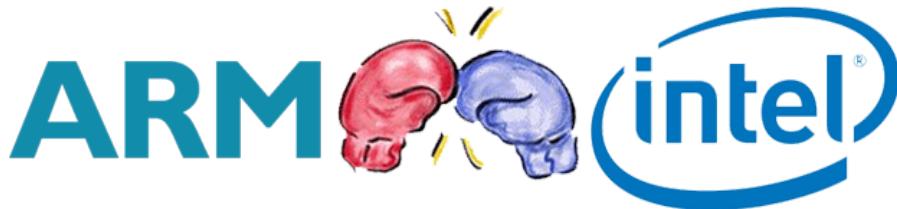
◇운용체계(OS)와 애플리케이션SW 호환=모바일기기에 탑재한 OS의 60~70%는 심비안이다. 심비안은 ARM기반에서만 구현된다. 하지만 인텔의 강점은 x86프로세서에서 돌아가는 애플리케이션SW를 보유했다는 점이다.

김영선 ARM코리아 지사장은 "지금 모바일의 대부분의 애플리케이션은 ARM 위에서 돌아간다"며 "인텔이 애플리케이션 제대로 쓸 수 있을 때까지는 많은 시간이 필요할 것"이라고 밝혔다.

이에 박성민 인텔 삼무는 "인텔이 스트롱ARM에서 PC아키텍처로 선보인 것은 성능향상과 그동안 구축해온 PC애플리케이션을 바탕으로 한 개발기간 단축을 위한 것"이라며 "아톰은 새로운 시장을 창출할 것"이라고 강조했다.

박 삼무는 또 "센트리노 아톰은 인텔이 만든 CPU 중 가장 작으면서 강력한 성능을 갖고 있다"며 "사용자는 PC에서 느꼈던 성능과 만족감을 포기하지 않을 것"이라고 덧붙였다.

주문정기자@전자신문
mjoo@etnews.co.kr



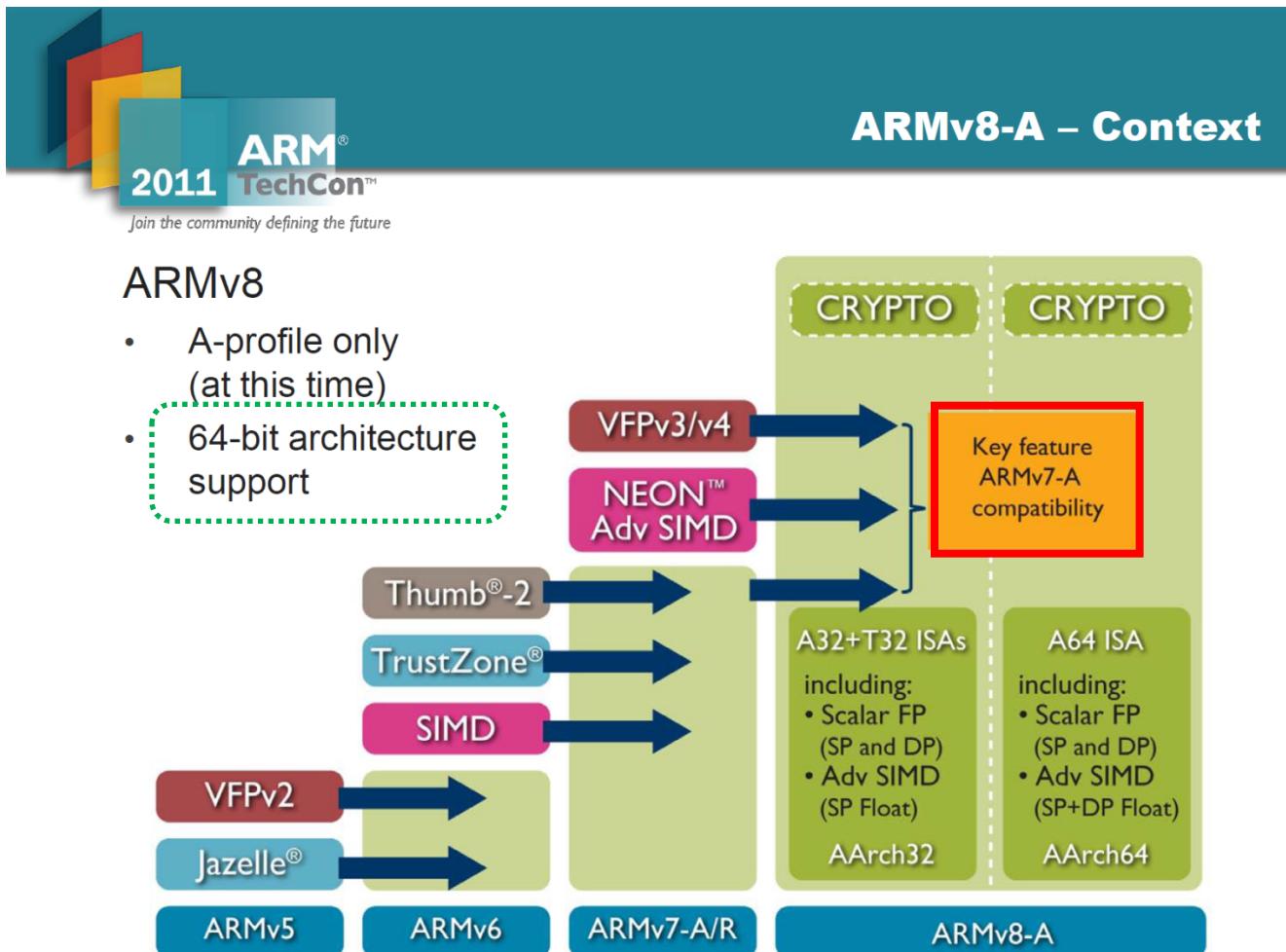
RISC

CISC

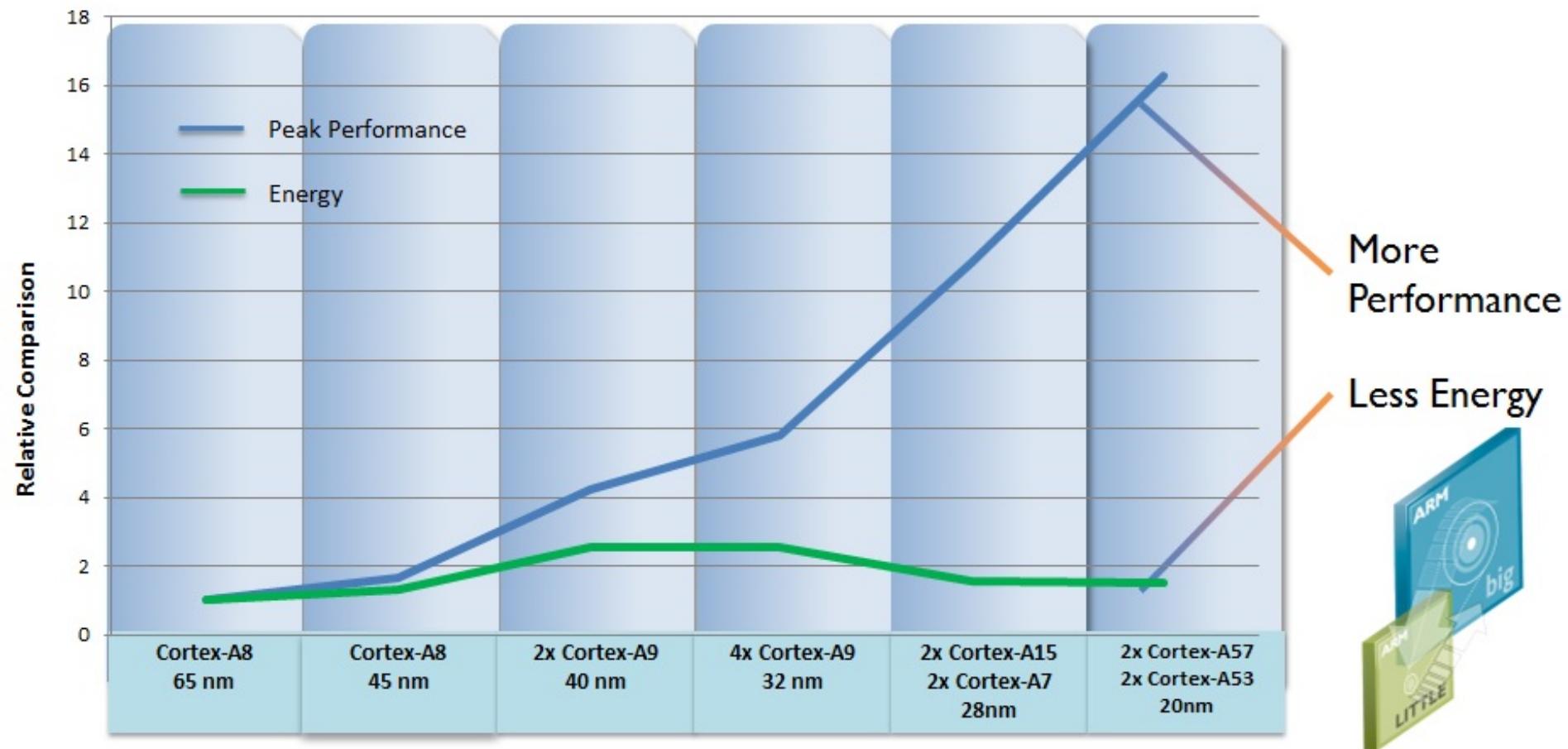
A RISC-based computer design approach means processors require fewer transistors than typical CISC x86 processors



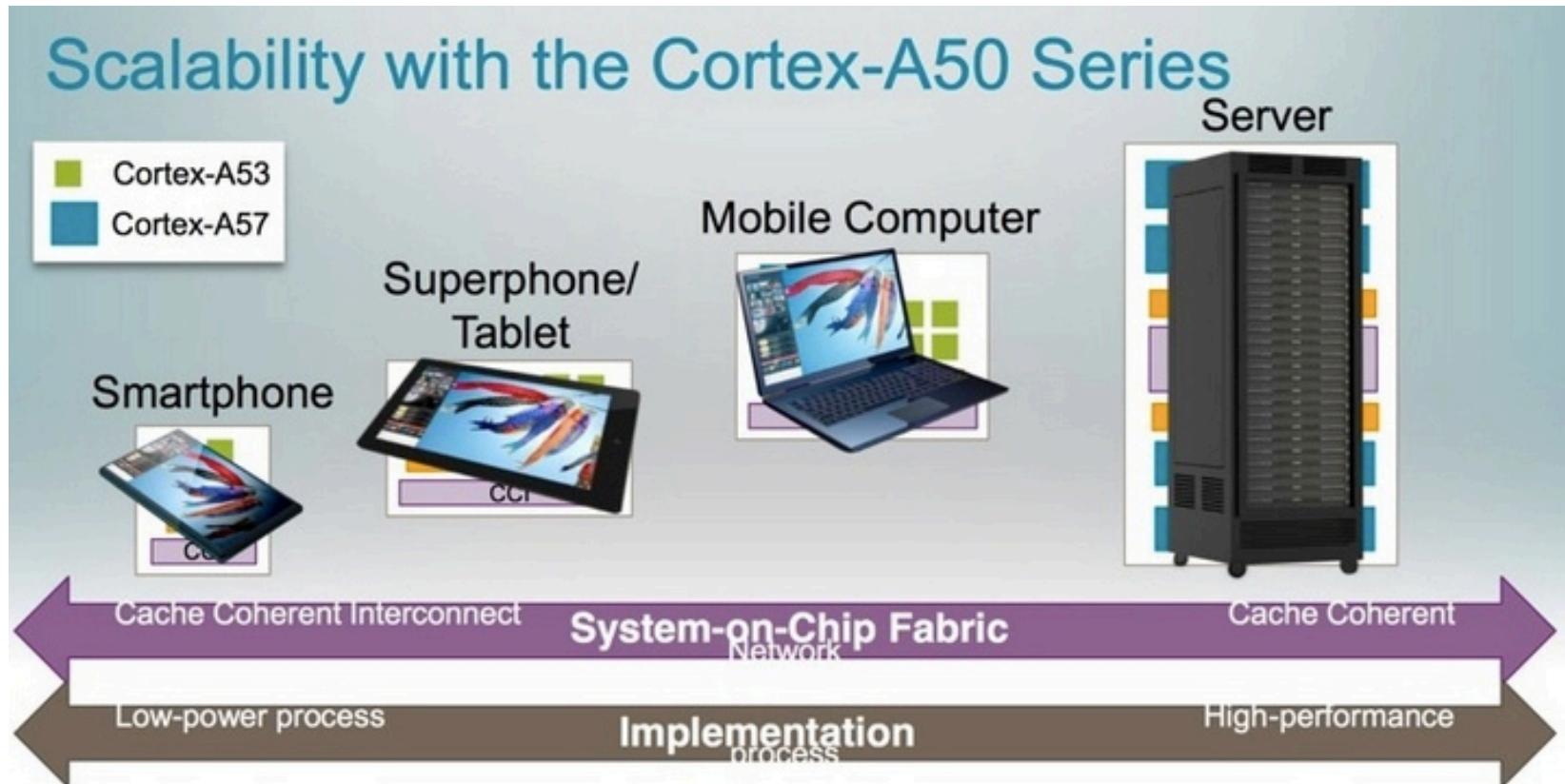
マイクロ 프로세서



マイクロ プロセ서



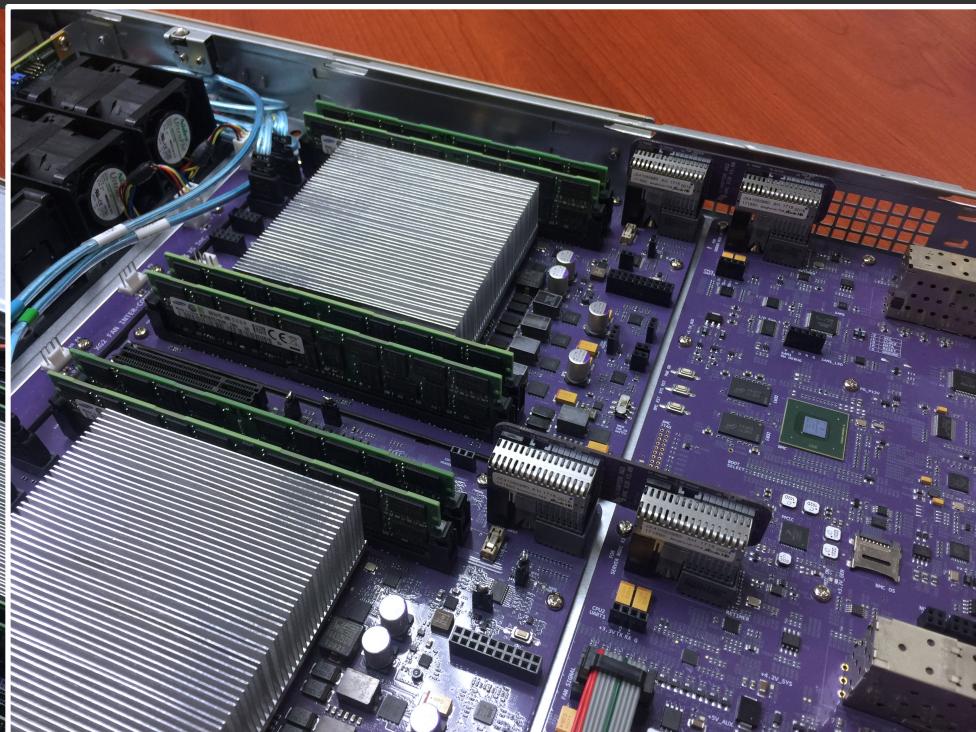
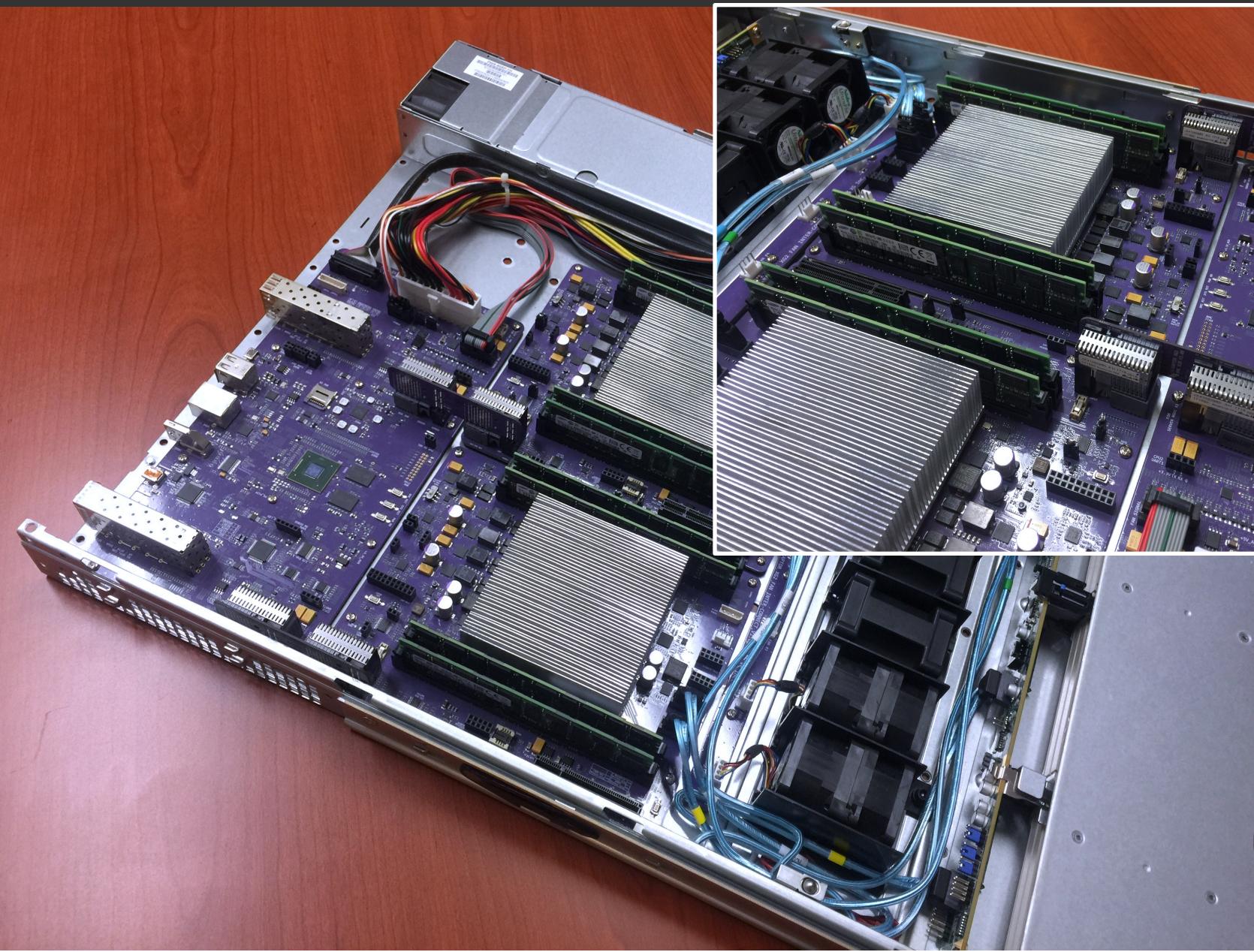
マイクロ プロセ서



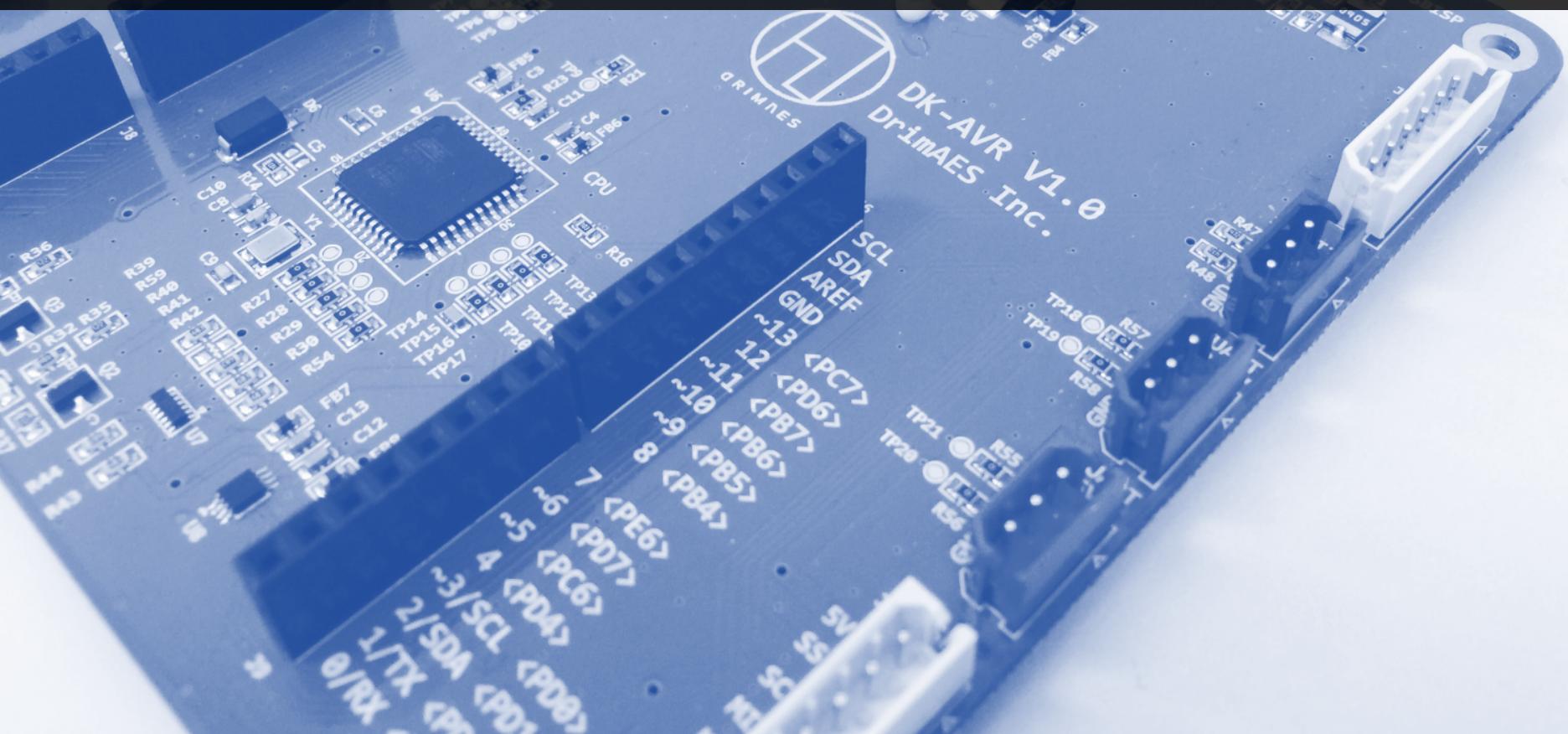
The Architecture for the Digital World® **ARM**



マイクロ プロセ서



운영체제



운영체제



운영체제

OS를 왜 써야만 하는가?

- 첫 번째 이유: 같은 자원에 동시에 접근하려는 태스크가 많을 경우
- 두 번째 이유: 우선순위가 확실하게 보장되어야 하는 경우

운영체제(OS)

非 실시간 운영체제(Non-RTOS)

- Linux, Windows, Windows CE
- Multi-Process
- 커널 + 파일 시스템 + TCP/IP 프로토콜 스택 = Heavyweight
- 주로 응용 프로그램이 많이 필요한 경우에 사용

실시간 운영체제(RTOS)

- VxWorks, pSOS, eCOS, MicroC/OS-II, TinyOS
- Multi-Thread
- 커널 (+ 파일 시스템 + TCP/IP 프로토콜 스택) = Lightweight
- 주로 정확한 시간 제어가 필요한 경우에 사용

“Real-Time” : 즉시, 대기 시간이 없는
⇒ 원하는 시간 내에 원하는 결과를 얻을 수 있다!



운영체제

운영체제(OS)

非 실시간 운영체제(Non-RTOS)

```
int sum(int a, int b)
{
    return (a + b);
}

int main(void)
{
    int a = 10, b = 20;
    int sum;

    sum = sum(a, b);

    printf("sum = %d \n", sum);

    return 0;
}
```

실시간 운영체제(RTOS)

```
void task_start(void *data)
{
    .....
    for (;;)
    {
        .....
    }
}

int main(void)
{
    OSInit();

    OSTaskCreate(task_start, (void *)0,
                (void *)&task_start_stk[OS_TASK_DEF_STK_SIZE
                - 1],
                0);

    OSStart();

    return 0;
}
```



운영체제

```
OS_STK task_start_stk[OS_TASK_DEF_STK_SIZE];
OS_STK task_led_stk[OS_TASK_DEF_STK_SIZE];
OS_STK task_fnd_stk[OS_TASK_DEF_STK_SIZE];

void task_start(void *data);
void task_led(void *data);
void task_fnd(void *data);

int main(void)
{
    OSInit();

    OSTaskCreate(task_start, (void *)0, (void *)&task_start_stk[OS_TASK_DEF_STK_SIZE - 1], 0);

    OSStart();

    return 0;
}
```



운영체제

```
void task_start(void *data)
{
    OS_ENTER_CRITICAL();
    TCCR0 = 0x07;
    TIMSK = (1 << TOIE0);
    TCNT0 = 256 - (CPU_CLOCK_HZ / OS_TICKS_PER_SEC / 1024);
    OS_EXIT_CRITICAL();

    OSTaskCreate(task_led, (void *)0, (void *)&task_led_stk[OS_TASK_DEF_STK_SIZE - 1], 1);
    OSTaskCreate(task_fnd, (void *)0, (void *)&task_fnd_stk[OS_TASK_DEF_STK_SIZE - 1], 2);

    for (;;)
    {
        OSTimeDlyHMSM(0, 0, 1, 0);
    }
}

void task_led(void *data)
{
    INT8U led_status = 0xFF;

    DDR_LED = 0xFF;

    for (;;)
    {
        led_status = ~led_status;
        PORT_LED = led_status;
        OSTimeDlyHMSM(0, 0, 0, 500);
    }
}

void task_fnd(void *data)
{
    INT8U fnd_status = 0x00;

    DDR_FND = 0xFF;

    for (;;)
    {
        fnd_status++;
        if (fnd_status > 99) fnd_status = 0;
        PORT_FND = (fnd_status / 10 << 4) | (fnd_status % 10);
        OSTimeDlyHMSM(0, 0, 1, 0);
    }
}
```

운영체제

```
void task_start(void *data)
{
    OS_ENTER_CRITICAL();
    TCCR0 = 0x07;
    TIMSK = (1 << TOIE0);
    TCNT0 = 256 - (CPU_CLOCK_HZ / OS_TICKS_PER_SEC / 1024);
    OS_EXIT_CRITICAL();

    OSTaskCreate(t1, task_start, data);
    OSTaskCreate(t2, task_led, data);

    for (;;)
    {
        OSTimeDlyHMSM(0, 0, 0, 500);
    }
}

void task_led(void *data,
{
    INT8U led_status = 0xFF;

    DDR_LED = 0xFF;

    for (;;)
    {
        led_status = ~led_status;
        PORT_LED = led_status;
        OSTimeDlyHMSM(0, 0, 0, 500);
    }
}

void task_start(void *data)
{
    OS_ENTER_CRITICAL();
    TCCR0 = 0x07;
    TIMSK = (1 << TOIE0);
    TCNT0 = 256 - (CPU_CLOCK_HZ / OS_TICKS_PER_SEC / 1024);
    OS_EXIT_CRITICAL();

    OSTaskCreate(t1, task_start, data);
    OSTaskCreate(t2, task_led, data);

    for (;;)
    {
        #if OS_CRITICAL_METHOD == 1
        #define OS_ENTER_CRITICAL()     asm volatile ("cli") /* Disable interrupts */
        #define OS_EXIT_CRITICAL()    asm volatile ("sei")  /* Enable  interrupts */
        #endif

        OSTimeDlyHMSM(0, 0, 0, 500);
    }
}

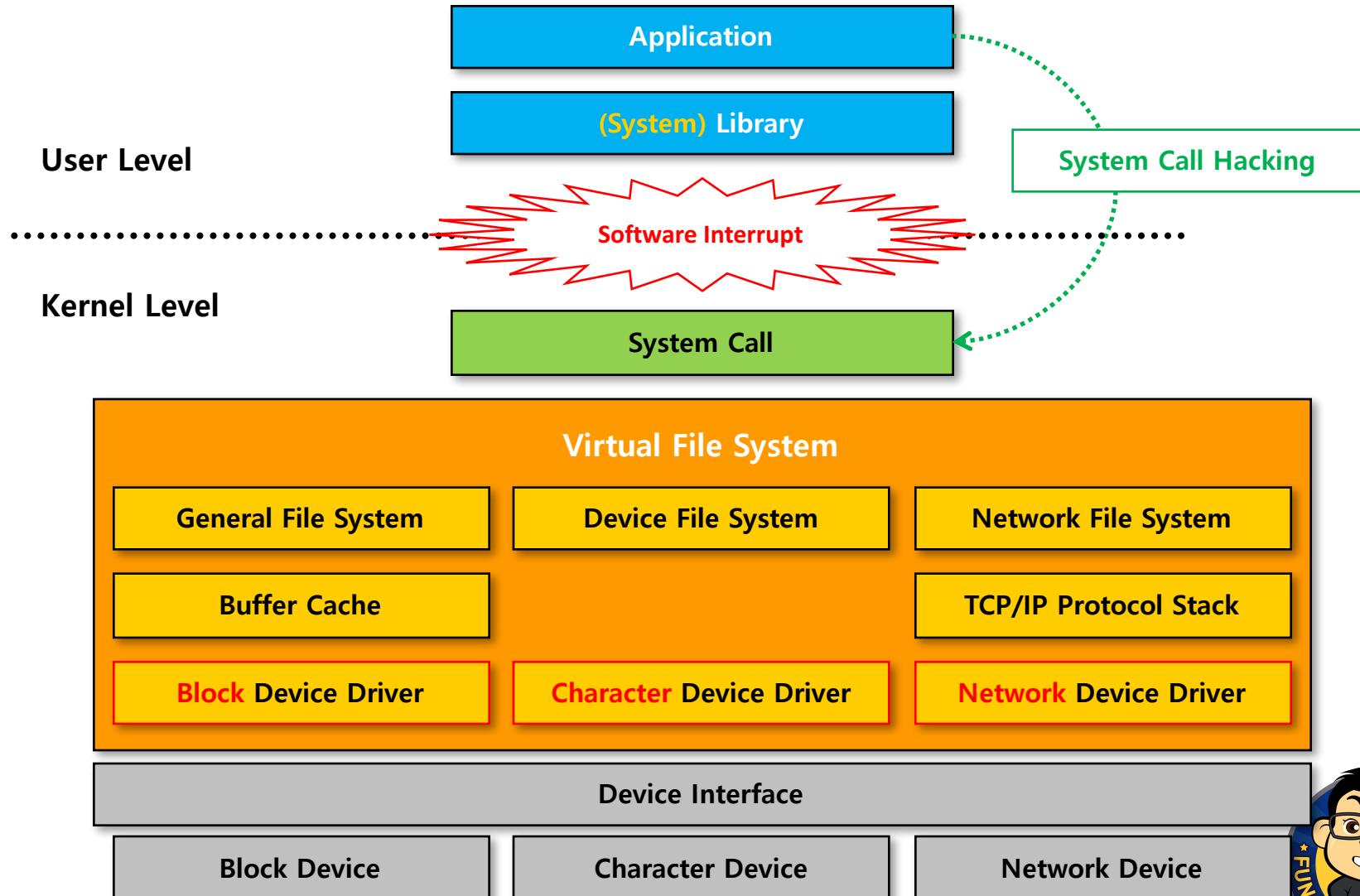
void task_led(void *data,
{
    INT8U led_status = 0xFF;

    DDR_FND = 0xFF;

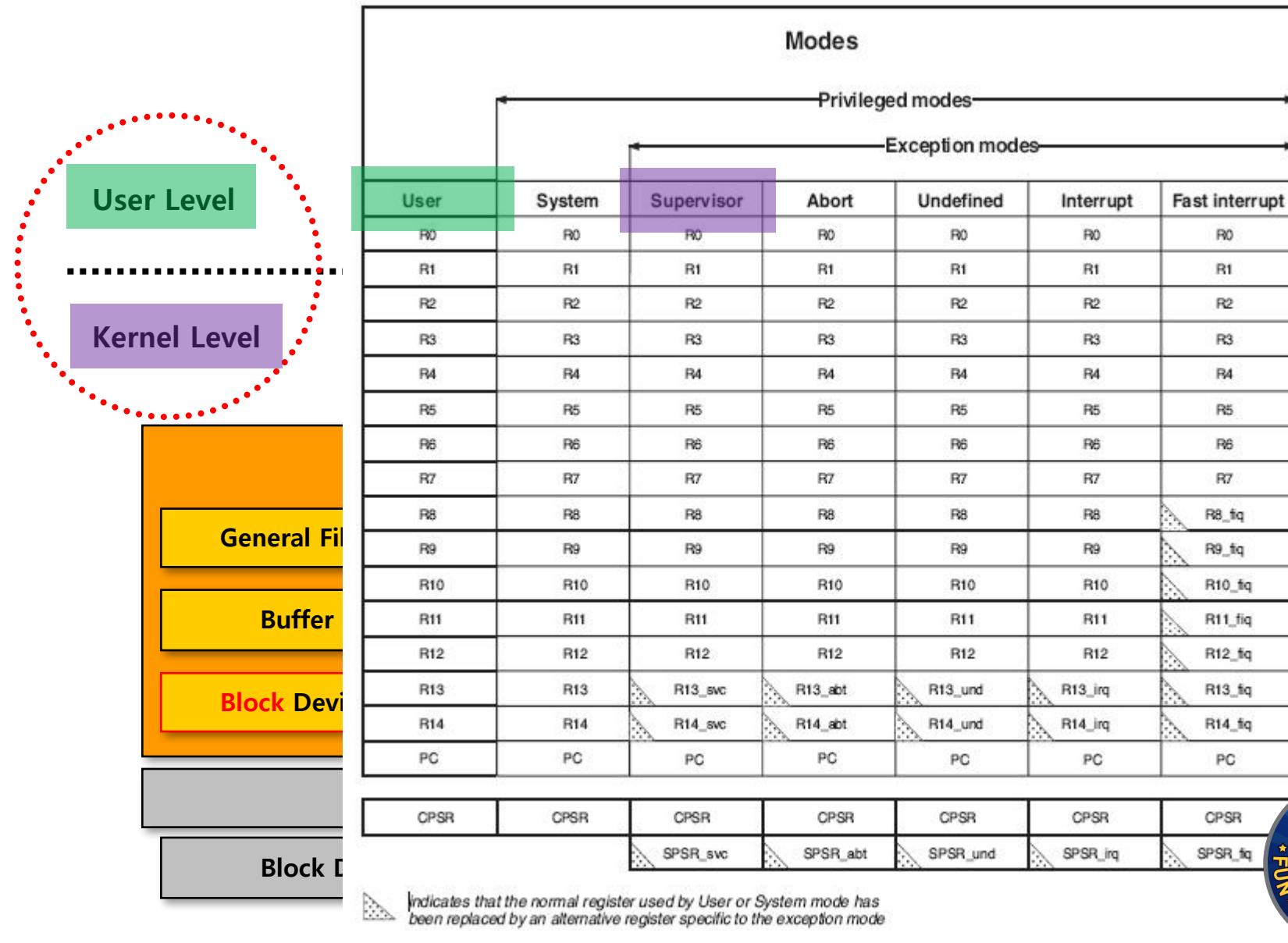
    for (;;)
    {
        fnd_status++;
        if (fnd_status > 99) fnd_status = 0;

        PORT_FND = (fnd_status / 10 << 4) | (fnd_status % 10);
        OSTimeDlyHMSM(0, 0, 1, 0);
    }
}
```

운영체제



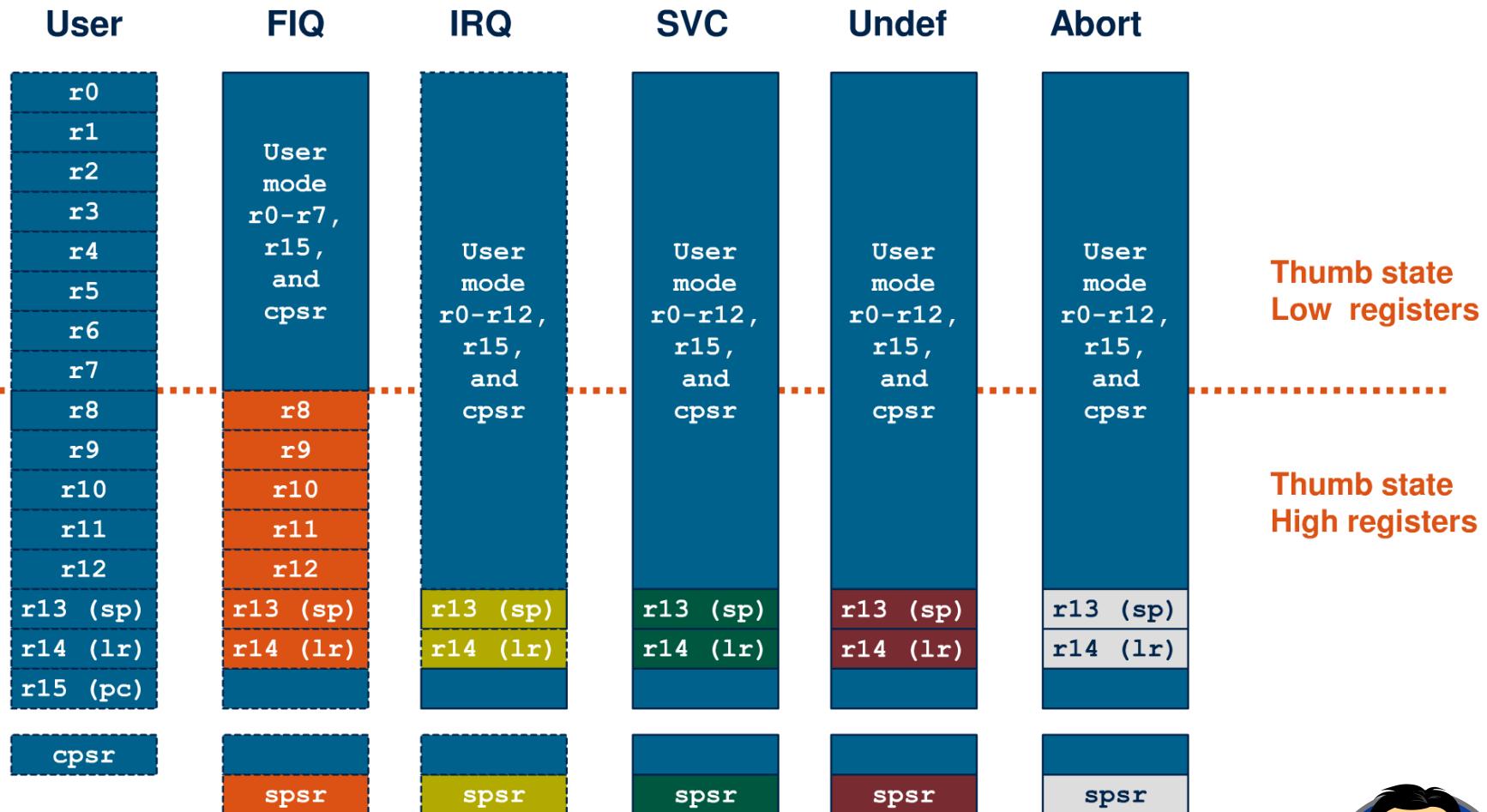
운영체제



Indicates that the normal register used by User or System mode has been replaced by an alternative register specific to the exception mode



운영체제



Note: System mode uses the User mode register set



운영체제

Current Visible Registers

Abort Mode

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 (sp)
r14 (lr)
r15 (pc)

cpsr
spsr

Banked out Registers

User

FIQ

IRQ

SVC

Undef

r8
r9
r10
r11
r12

r13 (sp)
r14 (lr)

r13 (sp)
r14 (lr)

r13 (sp)
r14 (lr)

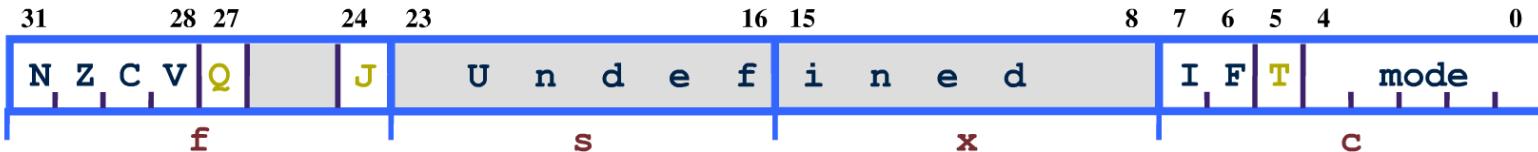
spsr

spsr

spsr

spsr

운영체제



Condition code flags

- N = Negative result from ALU
- Z = Zero result from ALU
- C = ALU operation Carried out
- V = ALU operation oVerflowed

Sticky Overflow flag - Q flag

- Architecture 5TE/J only
- Indicates if saturation has occurred

J bit

- Architecture 5TEJ only
- J = 1: Processor in Jazelle state

Interrupt Disable bits.

- I = 1: Disables the IRQ.
- F = 1: Disables the FIQ.

T Bit

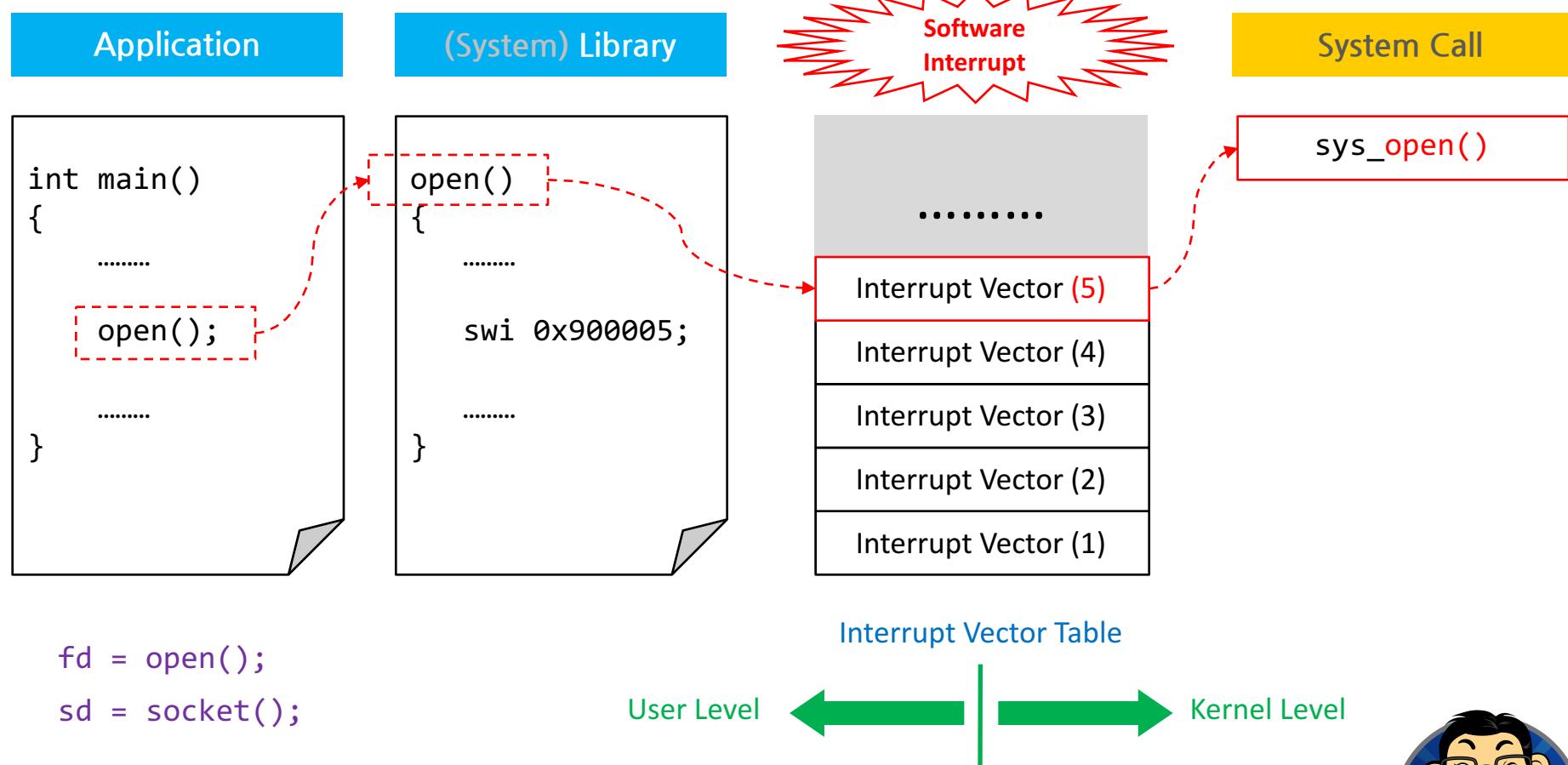
- Architecture xT only
- T = 0: Processor in ARM state
- T = 1: Processor in Thumb state

Mode bits

- Specify the processor mode

CPSR[4:0]	CPSR(ox)	Mode	Registers	상태 및 예외 (Exception)
10000	0x10	User	User	
10001	0x11	FIQ	_fiq	Fast interrupts
10010	0x12	IRQ	_irq	Standard Interrupts
10011	0x13	SVC	_svc	Reset, Power on, SWI
10111	0x17	Abort	_abt	Memory faults
11011	0x18	Undef	_und	Undefined instruction traps
11111	0x1F	System	User	

운영체제



운영체제

Firmware, RTOS System

Firmware

User App.

Hardware (Device)

Non-RTOS System

User Application

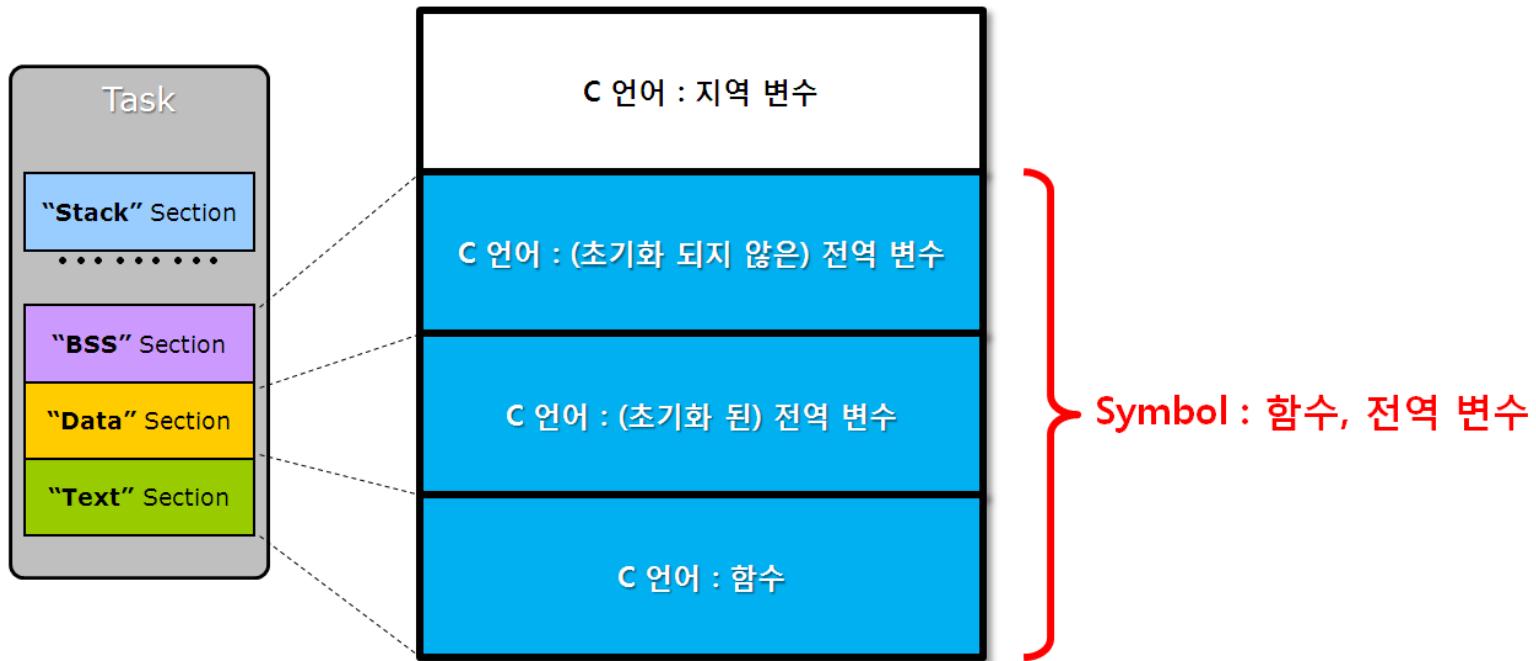
OS (Kernel)

Firmware

Hardware (Device)

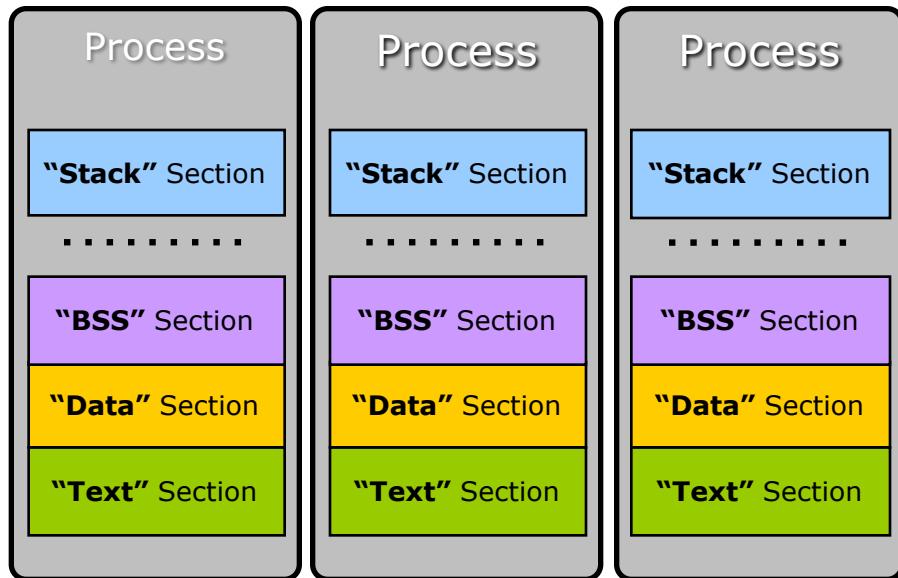


운영체제



운영체제

Multi-Process (Non-RTOS)



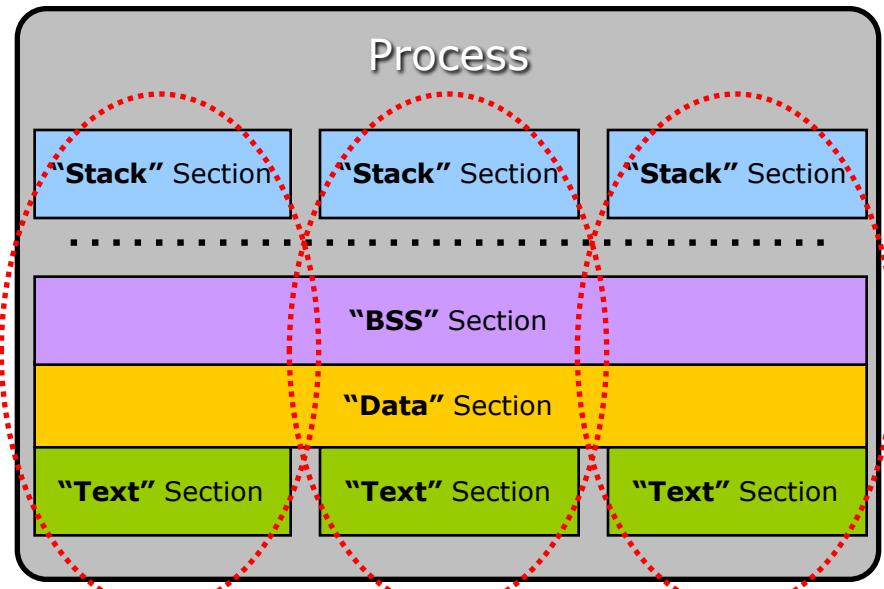
Task1 Process

Task2 Process

Task3 Process

- Linux, Windows
- 프로세스(Process)들마다 독립된 메모리 영역
- 프로세스의 生死 여부가 다른 프로세스에게 영향을 미치지 않는다!
- User Level 영역과 Kernel Level 영역으로 구분된다!

Multi-Thread (RTOS)



Task1 Thread

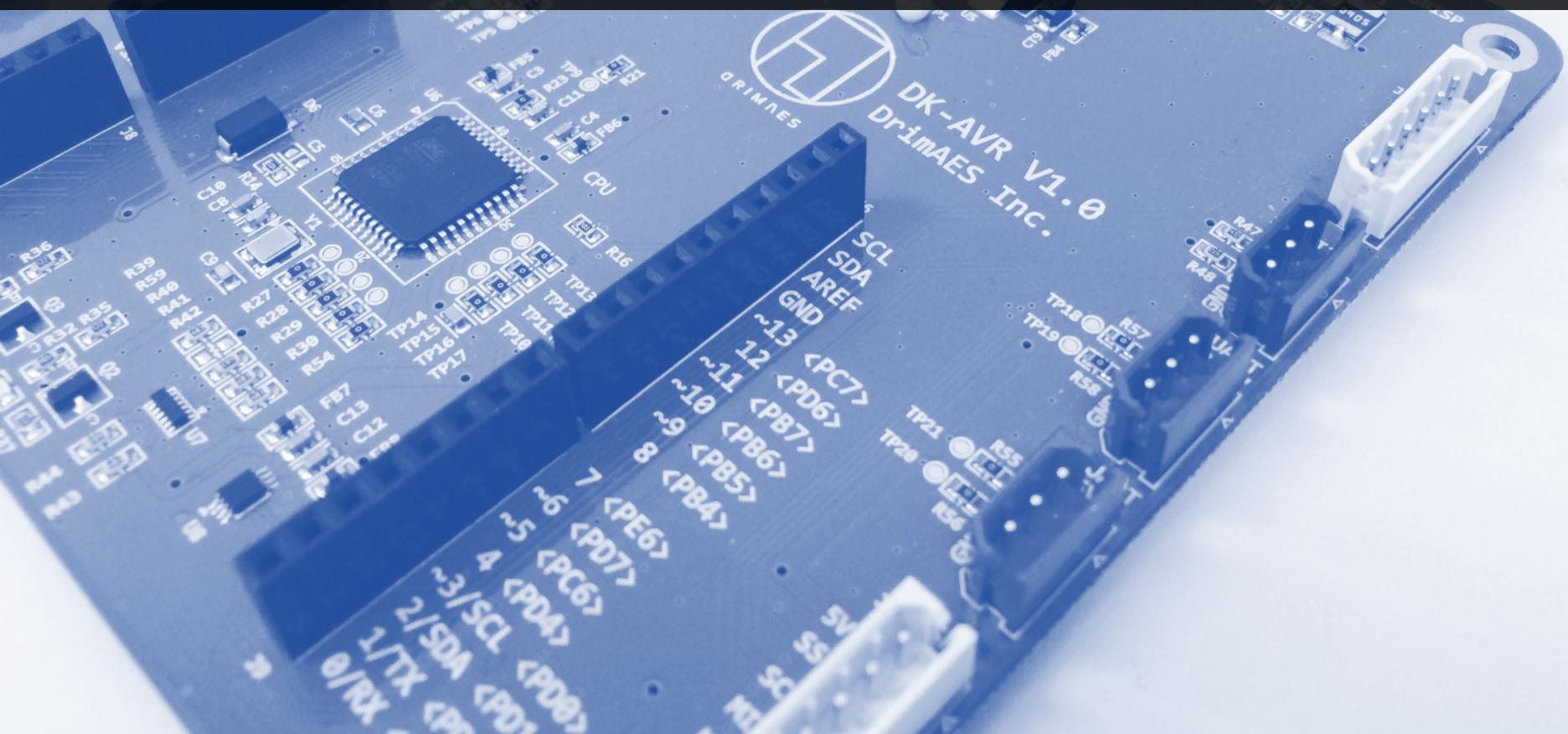
Task2 Thread

Task3 Thread

- VxWorks, pSOS, eCOS, MicroC/OS-II, TinyOS
- 쓰래드(Thread)들이 일부 영역(Data, BSS)을 서로 공유
- 쓰래드의 生死 여부가 다른 쓰래드에게 영향을 미친다!
- User, Kernel Level 영역의 구분이 없다!



DK-AVR 기반 RTOS 포팅 실습



DK-AVR 기반 RTOS 실습

https://github.com/funfunyoo/DK-AVR_RTOs

