Autosar 2.0 on ARM Processor

고려대학교
김세원
swkim@os.korea.ac.kr

목차

- AUTOSAR 등장 배경
- AUTOSAR 소개
- AUTOSAR experience

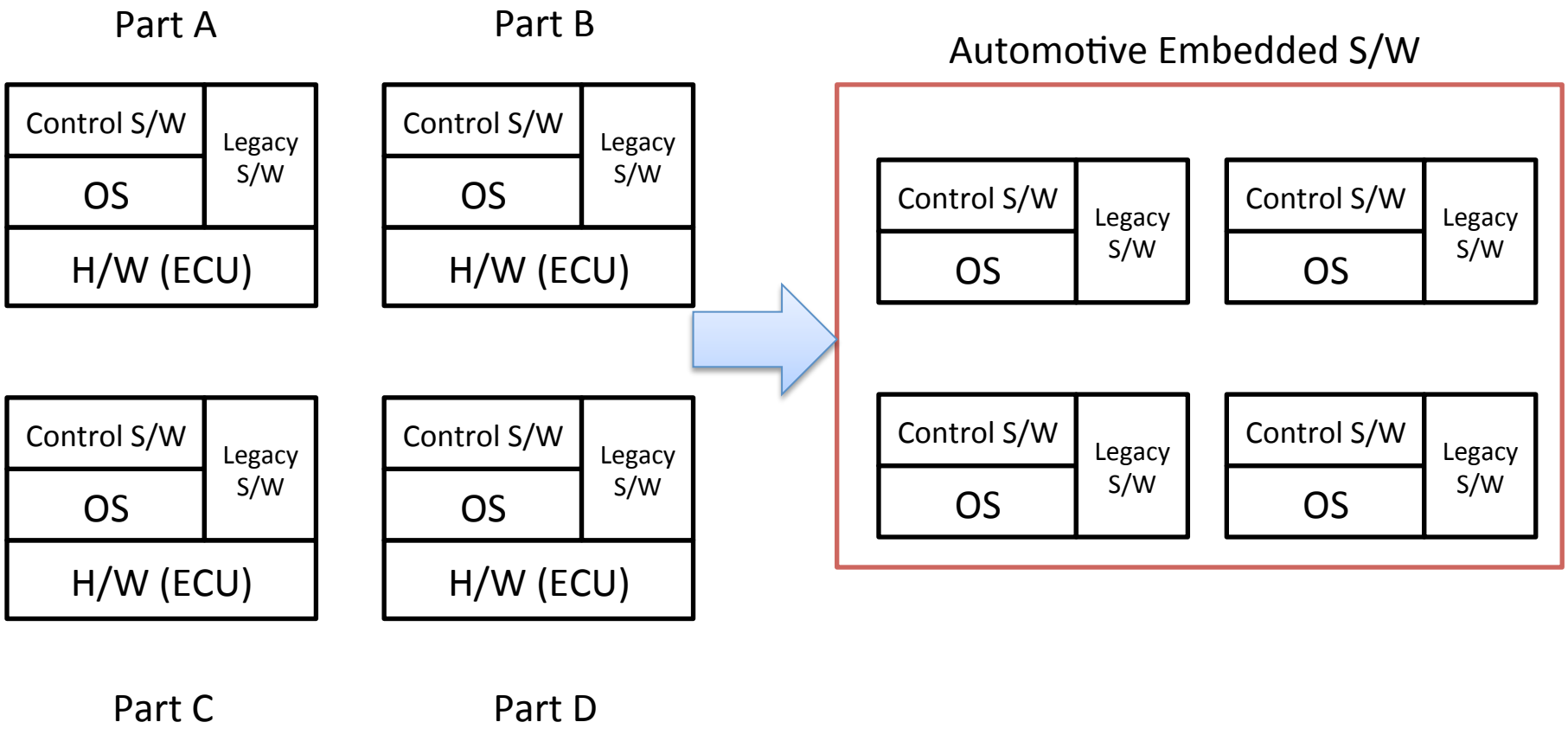
AUTOSAR 등장 배경

자동차 산업의 특징

- 조립산업
  - 부품을 공급받아 생산라인에서 조립하여 부가가치를 창출

- IT산업과 비교
  - 공통점
    - 자동차의 기계적인 부분을 제외 한다면,
      ECU와 마이크로컨트롤러와 같은 H/W와 제어 S/W로 구성
  - 차이점
    - 자동차
      - 부품은 각각의 H/W와 S/W로 구성됨, 자동차 회사는 이들을 CAN과 같은 내부 버스와 연결.
      - 각 부품의 S/W는 다른 S/W와 통신은 하지만 통합되진 않음
    - IT산업
      - H/W 부품과 device driver를 공급받아 하나의 S/W로 통합

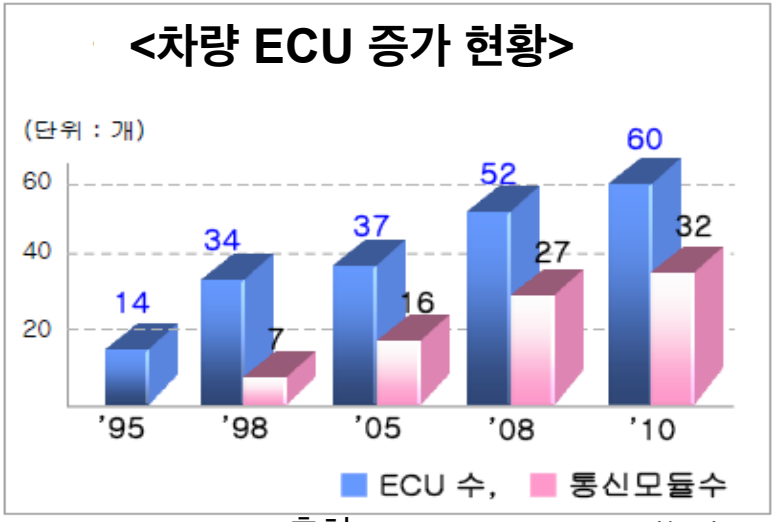
Automotive Embedded S/W



자동차 산업의 추세

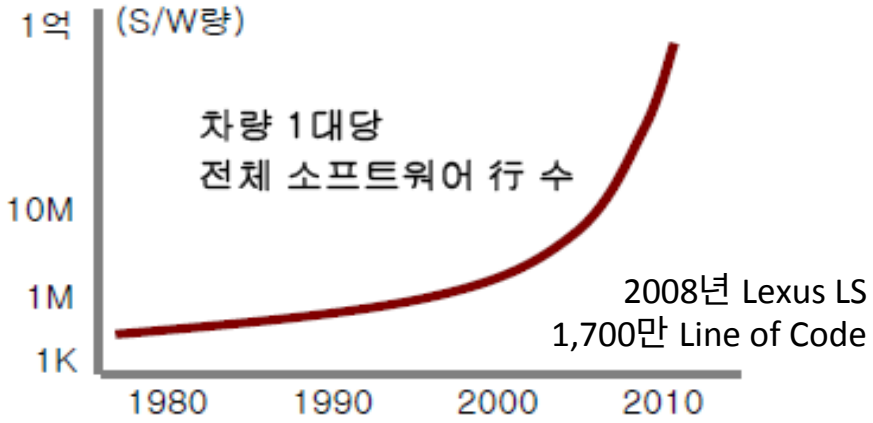
- 3가지 큰 변화
  - 전자장치의 증가, S/W 복잡도 증가, S/W표준화 필요

<차량 ECU 증가 현황>



<출처: AutoEver Systems ('10)>

<차량 소프트웨어 증가 현황>



<출처: Automotive Technology('08)>

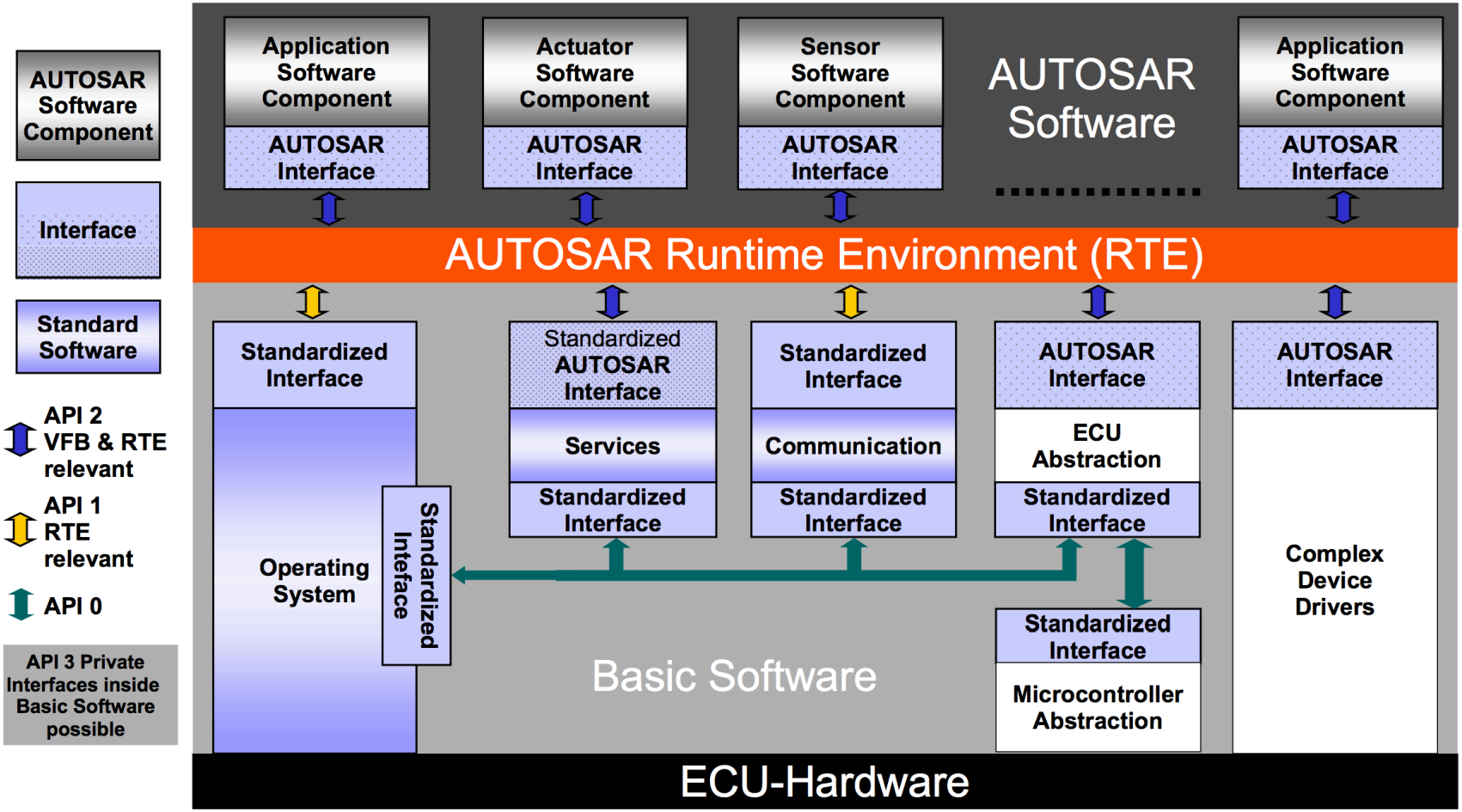
AUTOSAR 소개

AUTOSAR란?

- AUTOSAR
  (AUTomotive Open System Architecture)
  - 자동차용 표준 S/W 플랫폼
    - 자동차 전장용 임베디드 S/W 재사용을 통한 개발의 향상성을 도모
  - BMW를 중심으로 자동차 회사와 부품회사 및 연구소들로 이루어진 partnership 구성
  - 2.0 Release 버전으로 시작하여 현재 4.1 Release 까지 발전됨

AUTOSAR S/W 3 Layer

AUTOSAR Software, RTE, Basic Software로 구성



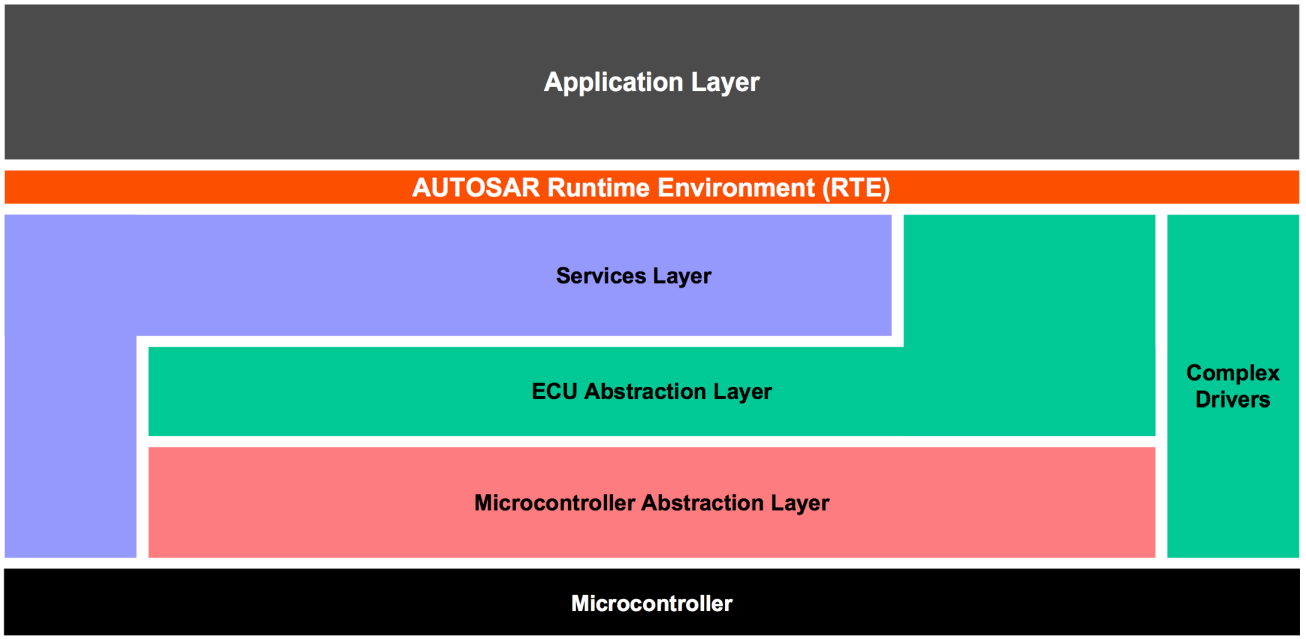
AUTOSAR S/W 컴포넌트

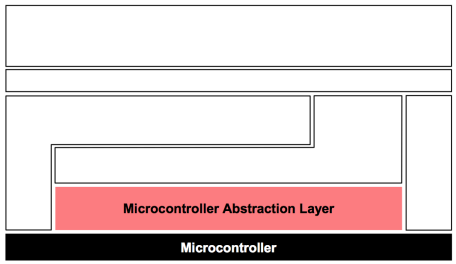
- AUTOSAR S/W
  - 자동차의 실제 기능을 담당하는 응용 S/W
    - 예) 에어백 소프트웨어, 엔진 점화플러그

- RTE (Run-Time Environment)
  - AUTOSAR S/W와 하부의 Basic Software를 분리 시켜주는 미들웨어와 같은 역할
  - Set of Interface

- BSW (Basic Software)
  - AUTOSAR를 다양한 ECU에 적용 가능하도록 만드는 layer
  - OS, EAL (ECU Abstraction Layer), MCAL (Microcontroller Abstraction Layer), CDD(Complex Device Driver)로 구성

Basic Software

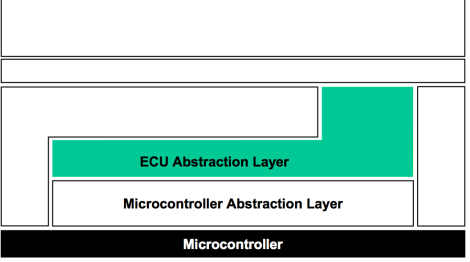




MCAL

- BSW에서 가장 H/W에 밀접한 layer
  - 마이크로컨트롤러의 internal driver를 포함하고, 마이크로컨트롤러에 연결된 내부 장치들와 메모리를 직접 접근가능
    - 내부장치: global timer, DRAM, mmu, cache

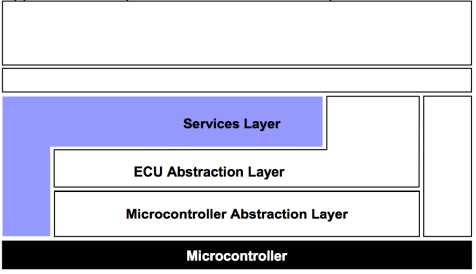
- 목적
  - 상위 layer의 마이크로컨트롤러 의존성을 제거



EAL

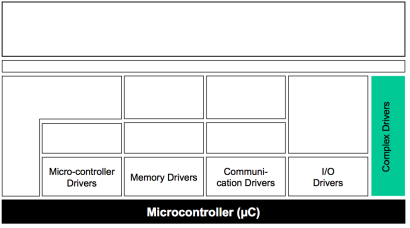
- ECU의 모든 H/W에 대한 API 제공
  - External , Internal H/W 대한 API
  - MCAL이 제공하지 않는 외부 장치에 대한 드라이버 제공
    - 외부장치: flash memory, CAN interface

- 목적
  - 상위 layer의 ECU의존성을 제거
  - 표준 API를 이용하면, ECU 종류에 상관없이 H/W접근



Service Layer

- Application S/W를 위한 서비스를 제공할 수 있는 BSW의 최상위 layer
  - 일부 MCAL과 EAL의 기능을 직접 수행하기도 함
  - 5가지 기능 제공
    - Operating system functionality
    - Vehicle network communication and management services
    - Memory service
    - Diagnostic services
    - ECU state management



Complex Device Driver

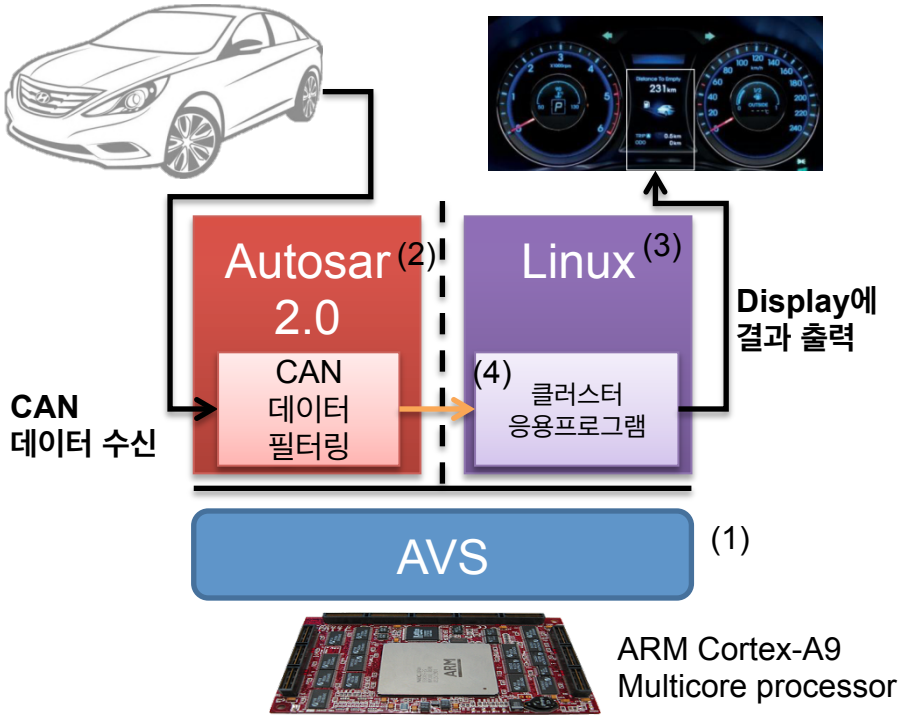
- 마이크로컨트롤러를 이용하여 센서 정보 추출 및 액츄레이터 제어를 위한 드라이버
  - Injection control
  - Electric valve control
  - Incremental position detection

- 목적
  - 특별한 기능이나 시간에 대한 제약사항이 높은 센서와 액츄레이터의 제어를 위한 인터페이스

AUTOSAR EXPERIENCE

연구 개요

AVS (Automotive Virtualization Software) 설계 및 구현
- 게스트 운영체제간 통신 지원
- 2개의 게스트 운영체제 구동
  - Autosar 2.0 : 기존 전장 제어 소프트웨어
  - Linux : 디지털 클러스터 기능이 추가된 소프트웨어



(1) AVS는 가상화 기술을 이용하여 물리적인 하나의 프로세서에서 2개의 운영체제를 동시에 실행

(2) Autosar 2.0은 차량용 실시간 Task를 실행하고, CAN 네트워크와의 통신을 담당

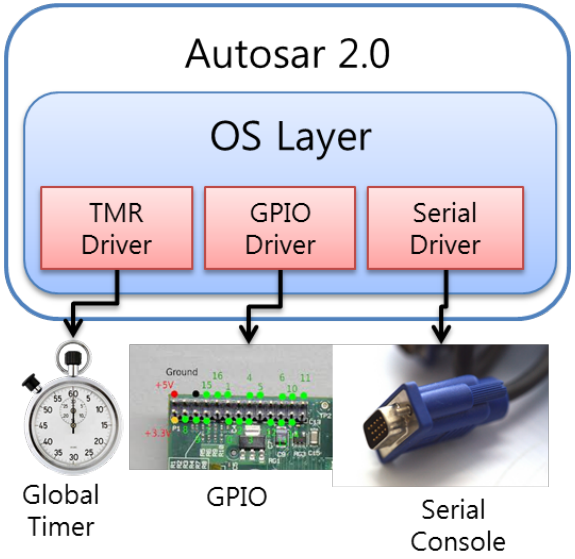
(3) Linux는 Display를 통해 클러스터 응용프로그램 결과를 실시간으로 출력

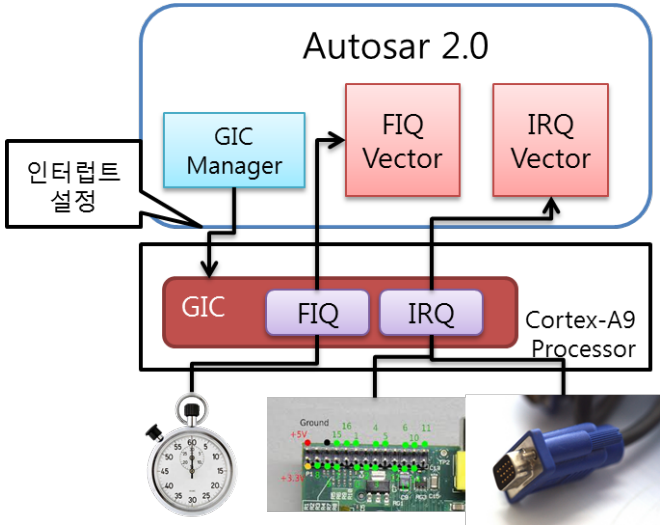
(4) 게스트 운영체제간 통신은 두 운영체제가 데이터를 실시간으로 통신 할 수 있도록 보장

AUTOSAR 수정 내용

게스트 운영체제 – Autosar 2.0 구현

- 타겟 보드의 입출력 장치 활성화(GPIO, 시리얼 콘솔, 글로벌 타이머)
- 실시간 태스크 스케줄링을 위한 FIQ 인터럽트 지원





타겟 보드의 입출력 장치 활성화

- GPIO, 글로벌 타이머, serial console의 장치 드라이버 추가
- 글로벌 타이머 : 태스크 스케줄링에 사용
- GPIO : 자동차 점멸등 태스크의 입출력 장치
- 시리얼 콘솔 : Autosar 2.0의 콘솔 입출력

FIQ 인터럽트 지원

- GIC(Generic Interrupt Controller)를 이용하여 높은 우선순위가 요구되는 장치의 인터럽트는 FIQ로 들어 오도록 설정
- Autosar 2.0에는 기존의 IRQ 뿐만 아니라 FIQ를 지원할 수 있도록 FIQ 인터럽트 벡터 및 핸들러 추가
- 실시간 태스크는 높은 우선순위의 FIQ에 의해 스케줄링 되므로 언제든지 CPU를 선점하여 작업 실행이 가능

AUTOSAR source code tree

- App_temp
  - AUTOSAR application
- CANIF
  - CAN interface driver
- MCAL
- RTE
- OS

```
drwx------  6 sbyoo hmc   4096 2013-01-07 16:28 App_temp
-rw-r--r--  1 sbyoo hmc    585 2013-01-10 14:46 App_temp.d
drwx------  4 sbyoo hmc   4096 2013-01-07 16:28 CANIF
drwx------  4 sbyoo hmc   4096 2013-01-07 16:28 COM
-rw-r--r--  1 sbyoo hmc    385 2013-01-10 14:46 Can_stub.d
-rw-r--r--  1 sbyoo hmc    421 2013-01-10 14:46 Convergence.d
drwx------  4 sbyoo hmc   4096 2013-01-07 16:28 DIO
-rw-r--r--  1 sbyoo hmc    392 2013-01-10 14:46 Dio.d
-rw-r--r--  1 sbyoo hmc    766 2013-01-10 14:46 Dummy_Function.d
drwx------  4 sbyoo hmc   4096 2013-01-07 16:28 ECUM
drwx------  4 sbyoo hmc   4096 2013-01-07 16:28 GENERIC
drwx------  5 sbyoo hmc   4096 2013-01-23 15:04 MCAL
-rwx------  1 sbyoo hmc   8979 2013-01-24 18:07 Makefile
drwx------  4 sbyoo hmc   4096 2013-01-10 17:17 OS
-rw-r--r--  1 sbyoo hmc    142 2013-01-10 14:46 Osal_stub.d
drwx------  4 sbyoo hmc   4096 2013-01-07 16:28 PDUR
drwx------ 10 sbyoo hmc   4096 2013-01-07 16:33 RTE
-rw-r--r--  1 sbyoo hmc    599 2013-01-10 14:46 Rte.d
-rw-r--r--  1 sbyoo hmc     27 2013-01-10 14:46 Rte_Os.d
-rw-r--r--  1 sbyoo hmc     38 2013-01-10 14:46 Stdlib_stub.d
-rwxr-xr-x  1 sbyoo hmc  77536 2013-03-01 22:24 autosar20.bin
-rwxr-xr-x  1 sbyoo hmc  95013 2013-03-01 22:24 autosar20.elf
-rw-r--r--  1 sbyoo hmc 117381 2013-01-07 16:28 build.log
drwx------  4 sbyoo hmc   4096 2013-01-07 16:28 intel
-rw-r--r--  1 sbyoo hmc    161 2013-01-10 14:46 main.d
-rw-r--r--  1 sbyoo hmc    114 2013-01-10 14:46 printf.d
```

MCAL source code

- CPU에 의존적인 코드
  - Serial
  - Multicore
  - Interrupt Controller

```
-rwx------ 1 sbyoo hmc   920 2013-03-01 14:00 Can_stub.c
-rwx------ 1 sbyoo hmc  1229 2013-01-07 16:28 Osal_stub.c
-rwx------ 1 sbyoo hmc   744 2013-01-10 15:24 Stdlib_stub.c
drwxr-xr-x 2 sbyoo hmc  4096 2013-01-08 15:29 printf
-rwxr-xr-x 1 sbyoo hmc 21622 2013-01-23 14:59 printf.c
drwxr-xr-x 2 sbyoo hmc  4096 2013-01-23 14:59 serial
-rwxr-xr-x 1 sbyoo hmc  7044 2013-01-23 15:00 serial.c
-rw-r--r-- 1 sbyoo hmc 11295 2013-01-11 19:10 smp.c
-rwxr-xr-x 1 sbyoo hmc  4262 2013-03-01 22:24 target_gic.c
-rwxr-xr-x 1 sbyoo hmc  2709 2013-01-23 15:03 timer.c
drwxr-xr-x 2 sbyoo hmc  4096 2013-01-24 16:30 trustzone
-rwxr-xr-x 1 sbyoo hmc  6450 2013-01-23 15:02 trustzone.c
-rwxr-xr-x 1 sbyoo hmc  2067 2013-01-25 17:24 tzipc.c
```

OS source code – entry.S

```
-rwxr-xr-x 1 sbyoo hmc 3427 2013-01-25 16:46 entry.S
-rwx------ 1 sbyoo hmc 1215 2013-01-25 17:24 main.c
```

```
        .global start
        .align 0
start:
        /*
         * Stack Setup for User Mode
         * This code will cause abort exception
         */
        cps     0x12 //#CPSR_MODE_IRQ
        ldr     sp, __init_sp_irq
        cps     0x13 //#CPSR_MODE_SVC
        ldr     sp, __init_sp_svc

        cpsid   i

        cpsie   f

        b       main

        .align 5
    .globl vector_table
vector_table:
    b vector_reset       /* ¥ê¥»¥Ã¥È             */
    b vector_und       /* Ì¤ÄêµÁÌ¿Îá           */
    b vector_swi         /* ¥½¥Õ¥È¥¡¥§¥¢³ä¹þ¤ß   */
    b vector_pabt     /* ¥×¥ê¥Õ¥§¥Ã¥Á¥¢¥Ü¡¼¥É */
    b vector_dabt  /* ¥Ç¡¼¥¿¥¢¥Ü¡¼¥É       */
    b vector_addrexcptn
    b vector_irq         /* IRQ                  */
    b vector_fiq         /* FIQ                  */
    b vector_safeg       /* SAFEG                */
```

```
        .align 5
        .global vector_irq
vector_irq:
        sub     lr,lr,#4
        stmfd   sp!,{r0-r12,lr}
        mov32   r0, 0x5004010c
        ldr     r0, [r0]
        mov32   r1, 0x9a
        cmp     r0, r1
        mov32   r1, 0x50040110
        str     r0, [r1]
        bleq    TimerHandler
        blne    TzIPCHandler
        ldmfd   sp!,{r0-r12,pc}^
```

OS source code – main.c

```
void main()
{
        //YSB
        //Initilize Tegra Serial
        QbSerialProbe();

        //Initialize SMP
        cpu_reset_handler_init();

        int i;
        for (i=1; i<4; i++) {
                printf("CPU %d : init!\n", i);
                boot_secondary(i);
        }

        //Initialize Timer & TrustZone
        trustzone_initialize();
        tegra_ipi_init();
        printf("IPI Initialized...\n");
        tegra_timer_init();
        printf("Timer Initialized...\n");

        //Initialize AUTOSAR BSW and Start RTE
        CALL_TASK(EcuMTask);

        //TODO: GPT Initialize and ISR for g_nRtOSCounter should be implemented

        do
        {
                if(g_bRtOSCounterUpdated)
                {
                        g_bRtOSCounterUpdated = FALSE;

                        //Run AdapterTask at every 100ms
                        if(g_nRtOSCounter % 100 == 0)
                                CALL_TASK(AdapterTask);

                        if(g_nRtOSCounter % 1000 == 0)
                                tegra_ipi_send(0x2, 7);

                        g_nRtOSCounter++;
                }
        }while(1);
}
```

```
void TimerHandler() {
        g_bRtOSCounterUpdated = TRUE;
        tegra_timer_clear();
}
```

결론

- AUTOSAR는 자동차용 소프트웨어 플랫폼
  - 계층화를 통해 ECU와 같은 H/W 종속성을 제거
  - AUTOSAR application은 다른 종류의 H/W에서 동작하는 AUTOSAR에서 재사용가능
  - 얻는 효과
    - 개발 시간 및 비용 감소
    - 자동차 S/W의 표준화 및 표준화 플랫폼의 안전성 검증을 통한 전체 S/W 안전성을 도모함