AUTOSAR 4.1.1의 최신 기술과 이해

Part 2: AUTOSAR OS

2013년 09월호 지면기사 / 글 | 박 대 영 <ds1620@infobank.net>

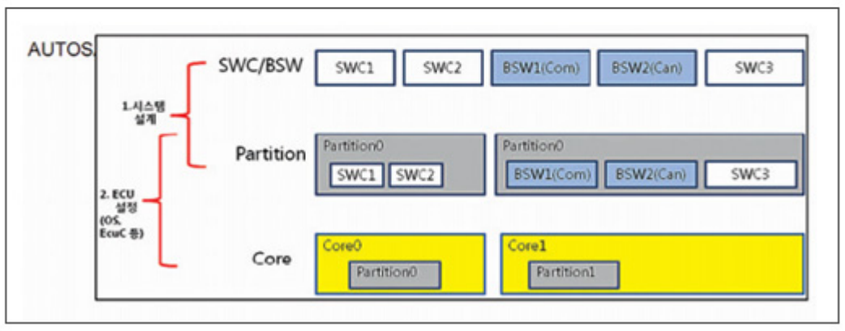
채승엽, 박대영, 이우승, 이헌철 등 인포뱅크(Infobank) AUTOSAR 사업부 멤버가 AUTOSAR 최신 기술에 대한 아티클을 6회에 걸쳐 연재한다. 인포뱅크 사업부는 AUTOSAR 추세와 요구사항에 맞게 AUTOSAR Ethernet Solution, AUTOSAR 플랫폼 교육, SWC 개발 Tool과 같은 AUTOSAR 전반에 걸친 업무를 수행하고 있다. 두번째 회는 'AUTOSAR OS'다.

1. AUTOSAR의 실무 이해
2. AUTOSAR OS
3. AUTOSAR RTE
4. AUTOSAR MCAL
5. AUTOSAR CAN
6. AUTOSAR FlexRay

AUTOSAR OS

지난 호에 이어 이번에는 AUTOSAR에서 사용되는 OSEK OS를 주제로 설명하고자 한다. OSEK OS v2.2.3과 관련해서는 이미 많은 사람들이 알고 있기 때문에, 이번에는 AUTOSAR v4.1.1에서 변경, 추가된 멀티-OS를 주제로 설명하고자 한다. 해외 자동차 회사에서는 앞으로 자동차 산업의 미래에 대해 많은 요구와 변화를 강조하고 있다. 주된 내용은 안전과 관련한 자동차 미래에 대한 것이다. 미래를 표현한 영화에 등장하는 자동차는 공통적으로 사람이 운전하는 것이 아니라 자동차 스스로 운전하는 모습이다. 이것은 먼 미래에 실현될 꿈이 아니라, 현재 프로토 타입으로 운행(테스트)하고 있는 현실이다. 그렇기 때문에 현재 자동차 회사에서 고민하고 있는 가장 큰 문제가 안전에 관한 것이다. 물론, 안전과 관련해서 AUTOSAR를 빼놓을 수 없다. 그 중에서 멀티코어-OS에 대해 많은 관심을 보이고 있다. 왜 안전을 얘기하면서 멀티코어-OS를 얘기하는지 이해가 안 되는 독자도 있을 것이다. 단일 코어에서도 구현 가능한 기능인데, 왜 굳이 멀티코어-OS를 사용해야 하는지에 대해서 설명하겠다. 안전의 중요한 요소 중 하나는 시간인데, 그림 1의 예에서 보듯 각각의 중요한 연산 동작을 여러 개의 ECU에서 처리하고, 그 결과를 전송해 동작하는 단일 동작 방식의 처리시간은 멀티코어에 비해 불리하다. 또, 실행 시간을 줄이는 것 이 외에도 단일 코어에서 모든 알고리즘을 처리하는 것보다 분산해 실행하는 것이 안전성을 높이는데 중요한 열쇠라고 할 수 있다. AUTOSAR에서도 멀티코어-OS에 대해 v4.1.1에서 명시하고 있고, 그 내용에 대해 설명하고자 한다.

[그림 1] 멀티코어를 적용한 예



[그림 2] 멀티코어에서 분할되는 블록 모습

AUTOSAR 4.x의 Partition과 Core 개념

AUTOSAR 멀티코어에서 중요하게 구분되는 것은 Partition 분할에 대한 개념이다.
우리가 하드디스크를 사용하고 싶은 용량으로 분할하듯이, AUTOSAR 멀티코어에서도 분리하고 싶은 SWC를 Partition 별로 각각 나눠야 한다. 사용하고 싶은 블록을 소프트 설계 시 미리 선정해 디자인하고, 그 구분된 소프트웨어를 코어별로 구분하는 작업이 필요하다.
ARAMIS의 멀티코어 사용 예를 보면 쉽게 이해되리라 생각된다.

News & Analysis

2륜차 리어램프에 최적! 4ch 리니어 LED 드라이버 (/article/articleView.asp?idx=3473)

다쏘시스템, 조메트리와 파트너십 체결... (/article/articleView.asp?idx=3472)

포레스터 리서치, 산업용 IoT SW 플랫폼 리더로 마인드스피어 선정 (/article/articleView.asp?idx=3471)

[보도자료]ST, IIoT와 자동차 애플리케이션을 위한 안전한 셀룰러 연결 제공 (/article/articleView.asp?idx=3470)

뤼즈 시스템 솔루션즈, 자동차 이더넷 테스터(ATE) 발표 (/article/articleView.asp?idx=3469)

TOP

Feature	Scalability Class 1	Scalability Class 2	Scalability Class 3	Scalability Class 4	Hardware Requirements
OSEK OS (all conformance classes)					
COUNTER Interface	8*	8*	8*	8*	
SWFRT Interface					
SCHEDULE Tables	2*	8*	2*	8*	
STACK Monitoring					
PROTECTION Hook					
TIMING Protection					Timer(s) with high priority interrupt
Global TIME / Synchronization Support					Global time source
MEMORY Protection					MPU
OS-APPLICATION			2*	2*	
SERVICE Protection					
CallTrustedFunction					(Non-)privileged Modes

* 확장클래스에서 요구한 최소 사양

[그림 3] AUTOSAR에서 확장된 클래스

	Scalability Class				Function definitions	Hook Function	Configuration Parameters	Scalability Class			
GetApplicationID			3	4	√		OsTrusted			3	4
GetISRID	1	2	3	4	√		OsAppAlarmRef			3	4
CallTrustedFunction			3	4	√		OsAppCounterRef			3	4
CheckISRMemoryAccess			3	4	√		OsAppIsrRef			3	4
CheckTaskMemoryAccess			3	4	√		OsAppScheduleTableRef			3	4
CheckObjectAccess			3	4	√		OsAppTaskRef			3	4
CheckObjectOwnership			3	4	√		OsRestartTask			3	4
StartScheduleTableRel	1	2	3	4	√		OsAppErrorHook			3	4
StartScheduleTableAbs	1	2	3	4	√		OsAppShutdownHook			3	4
StopScheduleTable	1	2	3	4	√		OsAppStartupHook			3	4
NextScheduleTable	1	2	3	4	√		OsTrustedFunctionName			3	4
StartScheduleTableSynchron		2		4	√		OsProtectionHook		2	3	4
SyncScheduleTable		2		4	√		OsIsrResourceLockBudget		2		4
SetScheduletableAsync		2		4	√		OsIsrResourceLockResourceRef		2		4
GetScheduleTableStatus	1	2	3	4	√		OsIsrAllInterruptLockBudget		2		4
IncrementCounter	1	2	3	4	√		OsIsrExecutionBudget		2		4
GetCounterValue	1	2	3	4	√		OsIsrOsInterruptLockBudget		2		4
GetElapsedValue	1	2	3	4	√		OsIsrTimeFrame		2		4
TerminateApplication			3	4	√		OsScalabilityClass	1	2	3	4
AllowAccess			3	4	√		OsTaskResourceLockBudget		2		4
GetApplicationState			3	4	√		OsTaskResourceLockResourceRef		2		4
ProtectionHook		2	3	4		√	OsTaskAllInterruptLockBudget		2		4
StartupHook_<App>			3	4		√	OsTaskExecutionBudget		2		4
ErrorHook_<App>			3	4		√	OsTaskOsInterruptLockBudget		2		4
ShutdownHook_<App>			3	4		√	OsTaskTimeFrame		2		4

[그림 4] SC 정의 API 함수

Service	Task	Cat1 ISR	Cat2 ISR	Error Hook	PreTask Hook	PostTask Hook	Startup Hook	Shutdown Hook	Alarm Callback	Protection Hook	Extended	Adapted	Unchanged	New
GetNumberOfActivatedCores	√		√											√
GetCoreID	√	√	√	√	√	√	√	√	√	√				√
StartCore														√
StartNonAutosarCore														√
GetSpinlock	√		√											√
ReleaseSpinlock	√		√											√
TryToGetSpinlock	√		√											√
ShutdownAllCores	√		√	√			√							√

[그림 5] AUTOSAR 4.1.1에서 새로 추가된 API

확장 클래스(Scalability class)

AUTOSAR OS는 OSEK/VDX와 HIS(Hersteller Initiative Software)에서 요구하는 사양을 포함하는 클래스를 확장했다. 그림 2에서 보면 SC1 ~ SC4까지 단계별로 포함하고 있는 요구 사양을 정리했다. 그 중에서 SC 단계별로 요구하는 최소 단위의 사양을 정의하고 있다.

AUTOSAR 4.1.1 상호배제(mutual excusion)

AUTOSAR 4.x MultiCore-OS에서 중요한 사양의 핵심은 상호배제(mutual excusion)이다.
흔히, 테스크 프로그램에서 자주 사용되는 MUTEX를 말한다. 상호배제가 중요한 이유는 싱글코어에서 사용했던 GetResource를 멀티코어에서 사용할 경우, 내가 점유한 리소스를 다른 코어의 Task에서도 사용이 가능하기 때문에 멀티코어 OS-Application에서는 Spinlock을 사용해 상호배제를 구현한다.

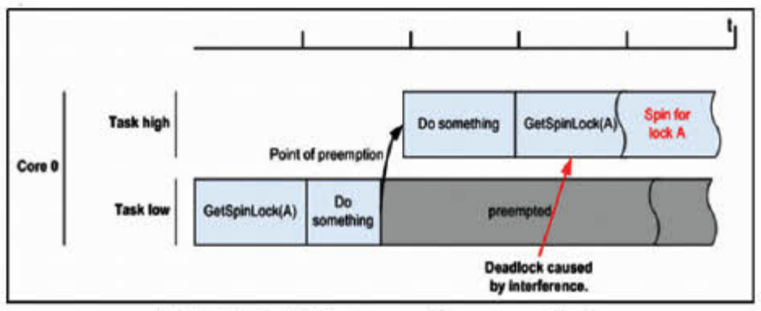
The spinlock mechanism

OSEK의 "ResourceType"과 같은 형식으로 "SpinlockType"을 사용한다.
SPinLock이란, 사용하고자 하는 자원이 사용 가능한 상태가 될 때까지 무한정 대기하는 순환 대기 개념이다.

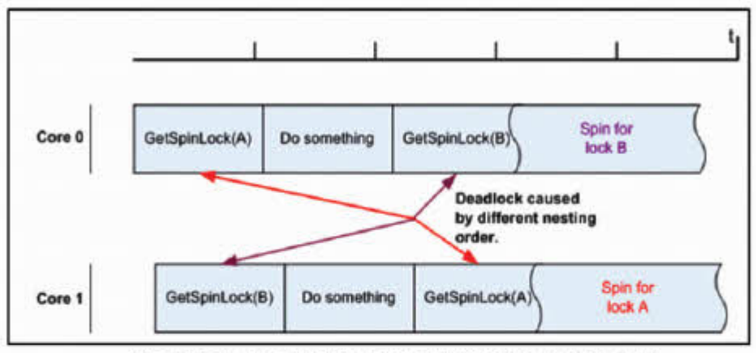
자원 획득 순서: 사용 자원 쭺 사용중 쭺 대기 쭺 사용가능 쭺 점유

낮은 우선순위의 Task가 'A' 자원을 먼저 점유한다. 높은 우선순위의 Task에서 동일한 자원을 점유하려고 시도하는데, 이 때, 높은 순위의 Task는 DeathLock이 발생한다[그림 6].
그림 7의 경우처럼 서로 다른 코어에서도 미리 점유한 자원을 다른 코어에서 사용하려할 때, DeathLock이 발생할 수 있다.
이러한, Deathlock을 막기 위해 순환형 대기(circular wait) 방식을 AUTOSAR에서 사용한다. 그림 8에서 보듯이 Task 2가 자원 S1, 2, 4, 6을 획득하기 위해서는 꼭 순서대로 1 쭺 2 쭺 4 쭺 6의 순서로 자원을 획득해야 한다. Task 1은 S2 자원을 사용 후 반환하고, 그 자원(S2)을 Task 2가 순서대로 사용하는 순환형 대기 개념이다. 이 때, Task 2는 S3의 자원을 점유할 수 없다. 또한, 그림에서 보듯이 S6 자원 사용 후 S5의 자원을 사용할 수 없다.

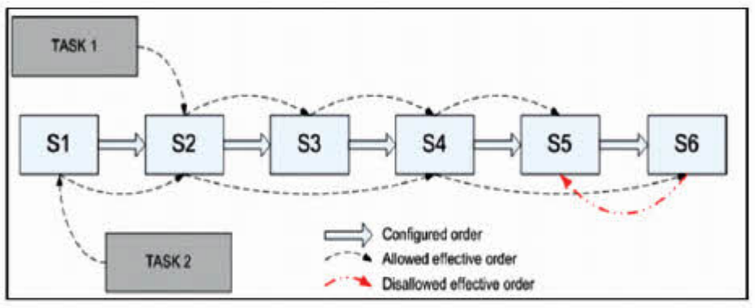
TOP



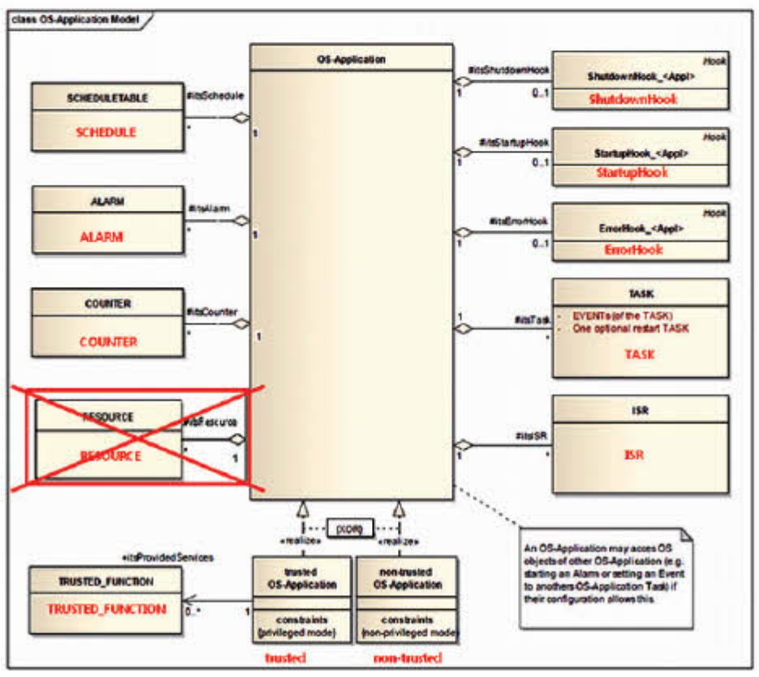
[그림 6] 높은 순위의 SpinLock이 DeathLock된다



[그림 7] 서로 다른 코어에서 점유한 자원을 서로 동시에 획득



[그림 8] 자원 점유를 위한 순서를 설정



[그림 9] 멀티코어 OS-Application 설계 시에는 RESOURCE를 사용하지 않는다

OS-Application

멀티코어 설계에서는 먼저 Os-Application을 구성해야 한다. 이 Os-Application에는 그림 9에서 보듯이 all Tasks, ISRs, Counter, Alarms, Schedule 등이 속해 있어야 한다. 또, 멀티코어의 설계에서는 반드시 Spinlock을 사용해야 한다.
같은 OS-Application에 속해 있는 Objs들은 서로 Access할 수 있다. 다른 Os-Application에서 Object에 Access할 수 있는 권한은 Configuration해 권한을 부여할 수 있다. Os-Application은 2가지의 경우가 있다.

1. Trusted Os-Application

(1) 런타임 시에 보호 기능을 실행하거나, Monitoring을 실행할 수 있다.
(2) 메모리에 제한 없이 Access할 수 있다.
(3) OS Module API의 동작은 런타임에 적용할 필요가 없다.
(4) 프로세서에서 지원하는 경우 특별한 권한으로 동작될 수 있다.

2. Non-Trusted Os-Application

(1) Monitoring해 실행될 수 없다.
(2) 메모리 Access에 제한을 받다.
(3) OS Module API의 런타임 동작에 대한 Access는 제한을 받는다.
(4) 프로세서에서 지원하는 경우라도 특별한 권한으로 동작될수 없다.

States Os-Applications

1. Active and accessible(APPLICA TION_ACCESSIBLE):
다른 Os-Application에서 OS Objs를 Access할 수 있다.
2. Currently in restart phase(APPLICA TION_RESTART):
다른 Os-Applica tion에서 OS Objs를 Access할 수 없다.
3. Terminated and not accessible(APPLICATION_TERMINATED): 다른 Os-Application에서 OS Objs를 Access할 수 없다. 상태는 변경되지 않는다.

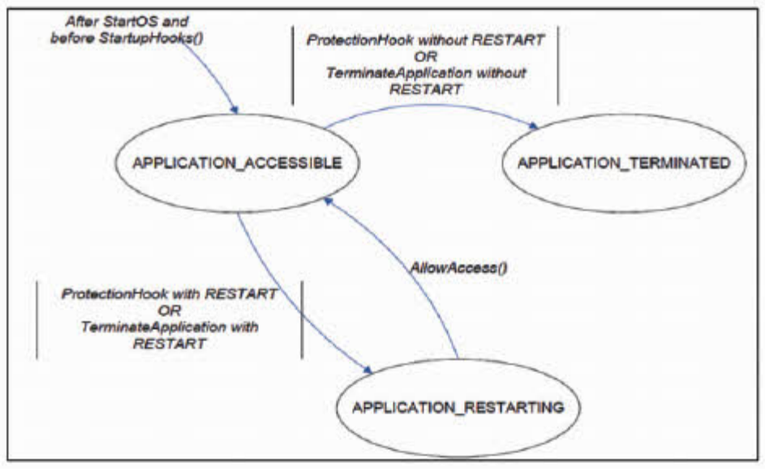
Inter-OS-Application Communication (IOC)

IOC는 Os-Application과 코어 간 의사소통과 메모리 보호에 직접 관여한다. 특히, 운영체제와 깊은 관계가 있다. 그림 9/10에서 보듯이 다른 코어 상호간의 통신을 IOC를 통해서 수행한다. Core0의 SWC는 다른 코어1의 SWC와 Rte_Send/Receiver 통신을 수행하려면, 반드시 IOC를 거쳐서 수행해야 한다.

Core0 SWC → RTE → IOC → RTE → Core1 SWC

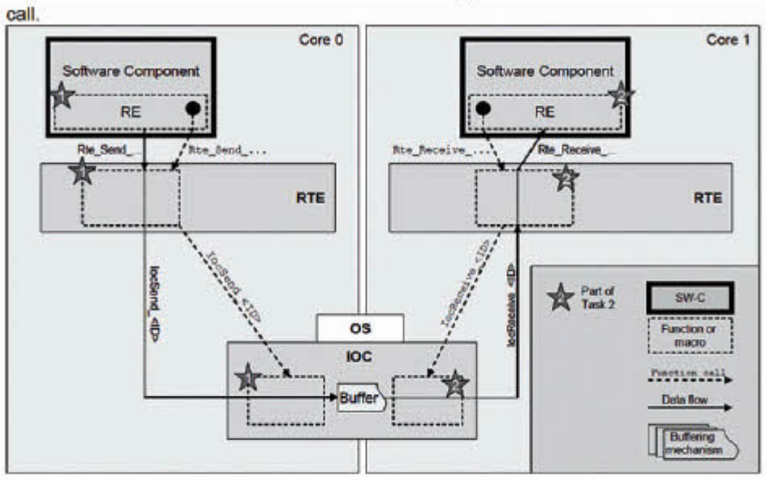
1개의 코어에서 사용되는 IOC는 생략할 수 있지만, 멀티코어에서 IOC는 매우 중요한 기능을 수행한다. IOC는 동일한 ECU에서 Os-Application 경계를 넘어 의사소통을 해야 하는 Client가 Access할 수 있는 통신 서비스를 제공한다.
IOC는 Sender-Receiver 통신만 제공한다. Server의 runnable을 호출할 경우, IOC를 통한 매개변수(notify)를 통해 호출한다. 이 때, 통신 사용자가 구성한 callback function config로 수신측에 access 할 수 있다(notifies).

TOP

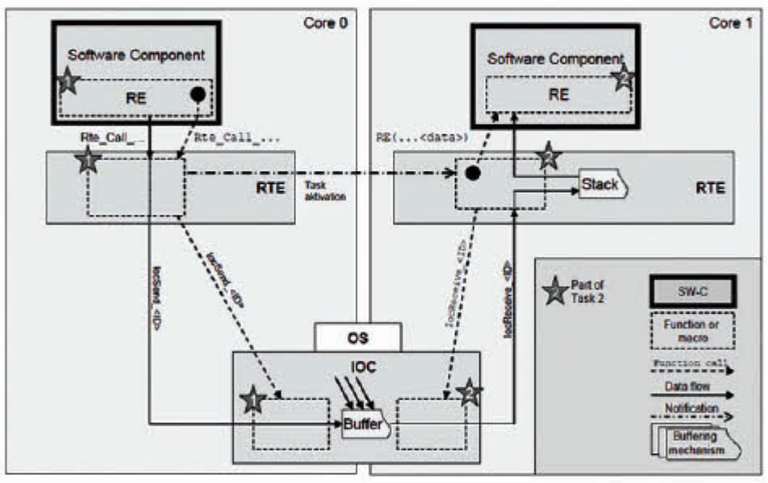


[그림 10] States of OS-Applications

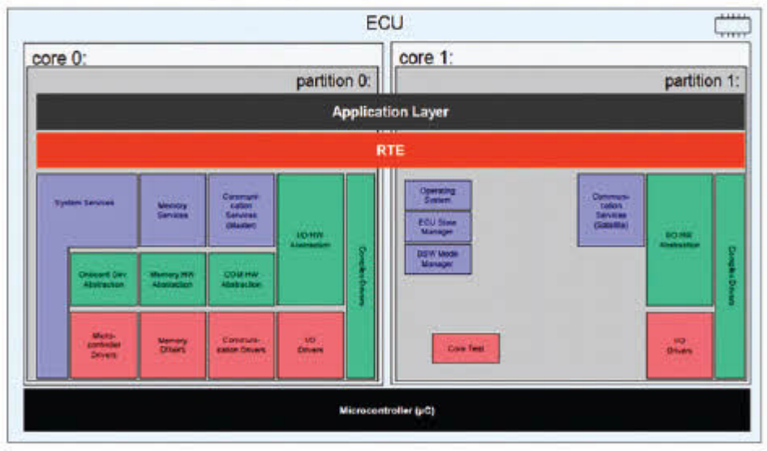
call.



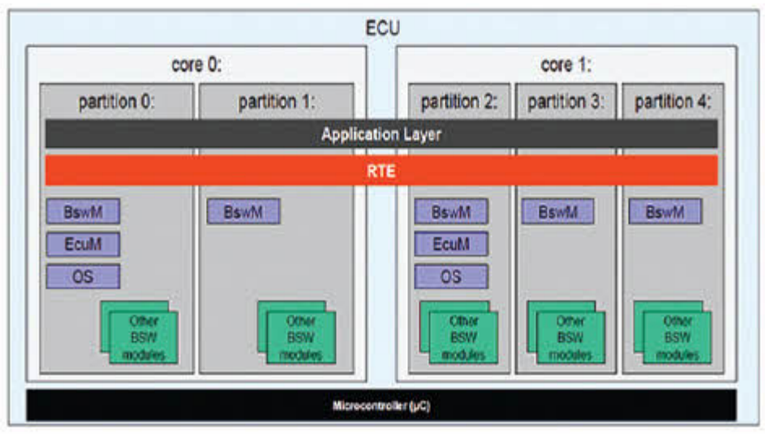
[그림 11] Send/ Receiver Interface



[그림 12] Client/ Server Interface



[그림 13] 1개의 코어에 1개의 partition이 있는 예



[그림 14] 1개의 코어에 2개의 partition으로 설계한 예

Partition 분할 구조

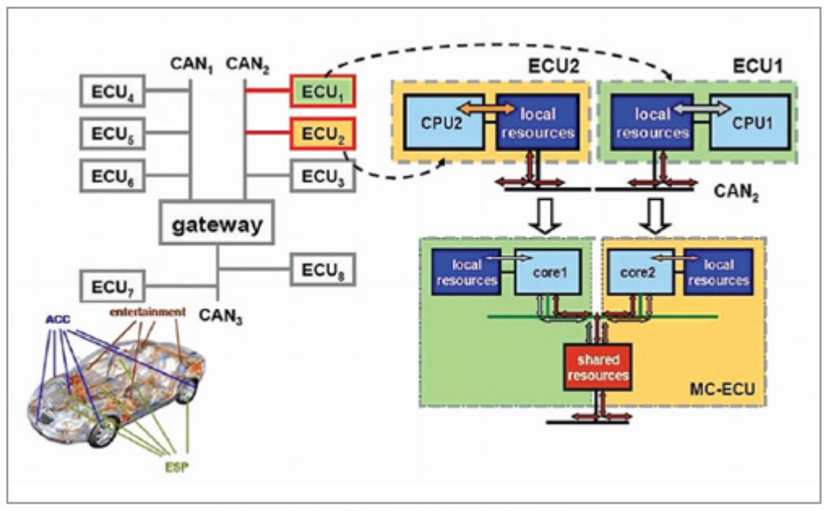
멀티코어 설계 시에 목적에 따라 분할할 수 있는 Partition이 제공된다. 그림 13은 1개의 Core에 1개의 partition으로 설계한 모습이다. 2개의 Core 중에 OS는 Core1에만 존재한다.
그림 14는 1개의 Core에 2개의 partition으로 구현된 그림이다. Core0과 Core1은 각각의 OS가 존재한다.

멀티코어 AUTOSAR로 구현한 여러 아키텍처들

AUTOSAR에서 멀티코어를 구현하는 방법은 여러 가지 방법이 있는데, 1개의 Core에만 AUTOSAR Basic Software를 두는 방법과 모든 Core에 각각 포함시키는 방법이 있다.

(1) 기능 안전을 고려한 방법의 설계는 상위 애플리케이션을 ASIL 안전등급으로 나누고 그 안전 기능에 대한 동작을 중심으로 구현된다.
(2) MCU의 동작을 빠르고 안정되게 최적화하는 방법의 설계는 MCU 동작과 직접적으로 관련된 제어/입출력/연산 등의 소프트웨어를 중심으로 구현한다. 이런 설계의 장점은 자원의 할당과 반환을 한 곳에 모아서 처리하기 때문에 빠른 처리를 할 수 있는 장점이 있다.
(3) 제품의 기능을 중심으로 설계하는 방법은 예를 들어, 오디오/비디오/통신/외부장치의 기능을 가진 제품을 각각 기능별로 분리해 설계를 할 수 있다. 이 때의 장점은 특화된 기능을 독립적으로 신속히 처리할 수 있는 장점이 있다.

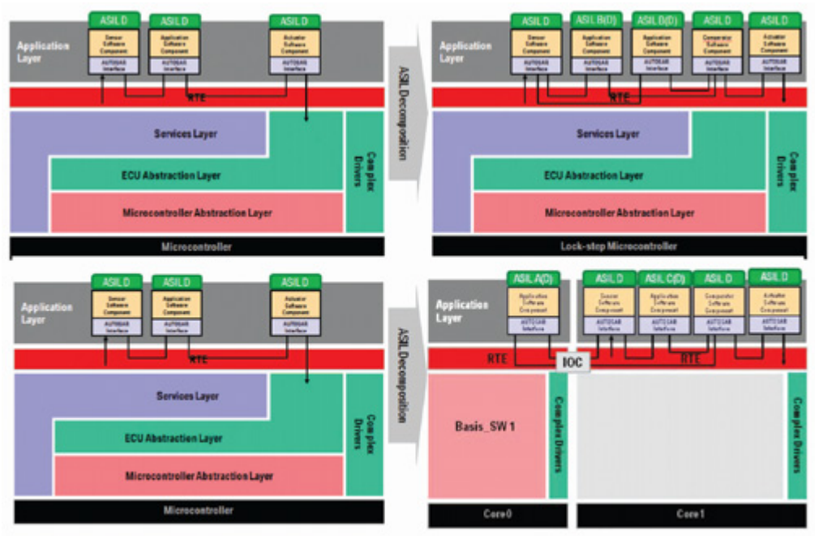
TOP



[그림 15] 실차에 적용하는 ECU 통합 제안

1. Recomp-Project motivation

CISTER(Research Centre in Real-Time and Embedded Computing Systems)의 Recomp Project(Reduced Certification Costs for Trusted Multi-core Platforms: 인증비용 절감 멀티코어 플랫폼)에서 실차의 2개 ECU를 한 개로 통합했을 경우를 그림 12로 표현했다. ECU 간 통신이 물리 CAN 통신에서 내부 Core 간 통신으로 변경됐다. ECU 상호간의 통신을 위해서는 물리 통신 라인을 거쳐서 통신을 하게 되면 당연히 통신에 대한 제약이 따르게 된다. 하지만, Core 간 통신 구조로 변경하게 되면 물리 통신으로 기능 구현이 어려웠거나, 구현하지 못했던 안전과 관련한 기능을 구현할 수 있다. 왜 멀티코어를 사용해야 하는지를 잘 보여주는 예이다.



[그림 16] ASIL 분리를 통한 구현 아키텍처 예

2. aramis Project

fortiss는 비영리 단체이고, 독일 뮌헨 기술대학의 학술 연구기관이다. 이곳에서 진행하는 Project인 aramis를 소개한다. Aramis 과제는 모바일, 자동차, 항공, 및 추가자원, 교통의 효율성과 편안함의 향상을 위한 기술 기반을 만들 레일의 영역에서 멀티코어 기술의 사용을 목표로 하는 과제이다. 그림 16은 안전등급으로 분리한 구현 아키텍처 예제의 모습이다. 이렇게 구현될 경우의 효과는 아래를 참고한다.

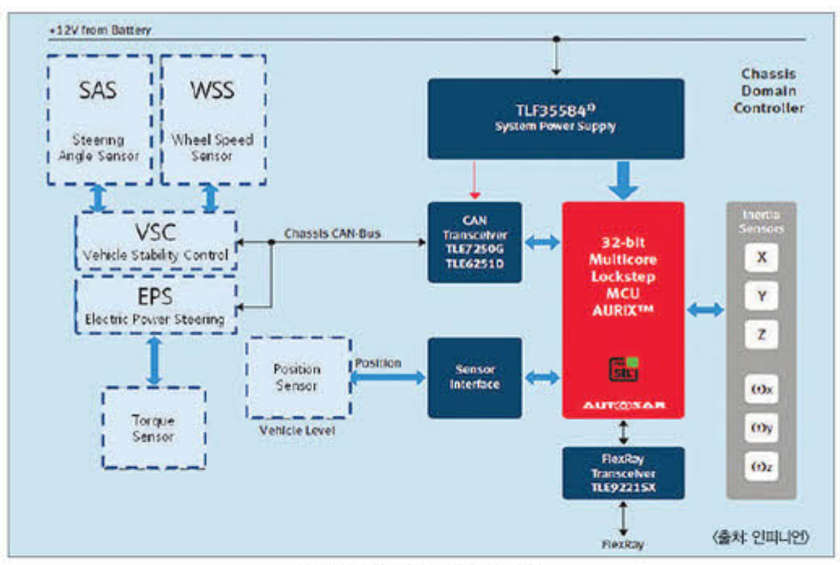
1. 개발비용 절감
2. 지능형 응용 프로그램 분배와 병렬이중화(parallel Redundancy)를 사용해 ASIL 분류에 더 나은 옵션을 제공
3. 핵심 분리를 통한 안전 증가

3. 3Core Infineon-AURIX MCU

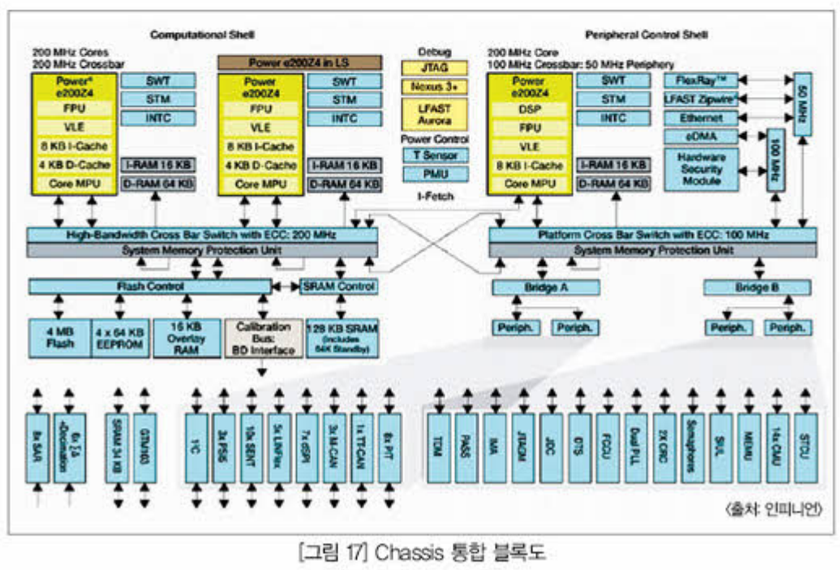
그림 17은 반도체 업체인 인피니언의 3 Core TC27x를 Chassis의 여러 ECU를 하나로 통합한 예를 보여준다.

4. Freescale MPC57xx 개발 키트

그림 18은 반도체 회사인 프리스케일의 MPC57xx에 3 Core가 적용된 아키텍처이다. HSM(하드웨어 보안)블록에 별도의 100 Mhz 클록을 삽입한 모습이 독특하다.



[그림 17] Chassis 통합 블록도



[그림 17] Chassis 통합 블록도

글을 마치며

필자가 지금까지 자동차 부품 개발을 해오면서 가장 많이 들었던 말이 "안전"이다. 똑같은 소프트웨어를 개발해도 차량용이란 것만 붙으면 정말 많은 것을 설계자에게 원한다. 설계자에게는 힘들고, 고단한 일이지만, 1% 이내의 안전이라도 더 확보할 수 있다면 당연히 대응해야 하는 것이 자동차 업무이다.

TOP

AUTOSAR 멀티코어는 "자동주행 시스템"으로 가기위한 바탕이 되는 기본 단계이다. "자율주행"은 필수적으로 여러 개의 ECU를 1개의 ECU로 통합해야만, 자율적으로 안전하게 제어할 수 있다. 이것은 선택이 아닌 필수이다. 이번 연재는 멀티코어를 구현하기 위한 AUTOSAR OS에 대해 설명했다. 필자가 경험한 AUTOSAR를 한마디로 말한다면, 역시 "안전"이라고 말하고 싶다. AE

100자평 쓰기 로그인

로그인후 입력하세요

등록

스마트앤컴퍼니(주) 대표이사 : 박성규 사업자등록번호 : 108-81-64739 통신판매업신고 : 2019-서울구로-2138호
서울특별시 구로구 디지털로34길 43, 607호(구로동, 코오롱싸이언스밸리1차) P: (Phone) 02-841-0017 F: (Fax) 02-841-0584 webmaster@smartn.co.kr