AUTOSAR 기반 차량용 소프트웨어를 위한 NVRAM Manager 및 시스템 서비스 구조 설계

류현기° 조성래 정우영
대구경북과학기술연구원
{hkryu, srcho, wyjung}@dgist.org

NVRAM Manager and System Service Architecture Design for AUTOSAR-based Automotive Software

Ryu HyunKi Cho SungRae Jung WooYoung
Division of Advanced Industrial Science & Technology, DGIST

요 약

자동차의 전기/전자 기술 발전과 함께 관련 소프트웨어의 구조 및 복잡성이 날로 증가되고 있으며 이러한 복잡한 구조를 간략화 하여 산업용으로 폭넓게 표준화된 소프트웨어의 인프라를 구축하는 방법이 필요하게 되었다. 이러한 목적에서 AUTOSAR라는 표준 단체가 탄생하게 되었으며 차량용 소프트웨어, 사용자 인터페이스 등의 개발을 위한 차량용 소프트웨어 구조를 정의한다. 본 논문은 AUTOSAR 구조에 기반하여 AUTOSAR 기반 시스템 서비스 및 NVRAM 관리자를 개발하기 위한 표준 구조를 설계 하고자 한다.

1. 서 론

좀 더 안전하고 편안한 운전 환경에 대한 요구가 증가함에 따라 이를 충족시키기 위해 자동차 개발 단계에서 많은 노력이 이루어지고 있다. 이와 같은 노력은 많은 부분 차량의 전기 전자 장치에서 진행되고 있으며 결과적으로 자동차의 전기 전자적 구조는 급격하게 복잡해지고 있다. 자동차의 전기 전자적 구조가 복잡하게 됨에 따라 이를 제어하기 위한 소프트웨어의 크기와 복잡도가 증가하게 되고 이는 곧 차량 개발에 소모되는 비용과 기간의 증가를 일으키고 있으며 뿐만 아니라 개발된 차량을 보수하거나 기능을 추가할 경우 많은 비용과 기간이 소모되게 된다. 예를 들어, 차량 개발 시 제조업체는 ECU의 기능과 속성을 정의하고 이를 제품 공급업체에 보내면 제품 공급업체는 이에 해당하는 제품을 개발해서 납품하고 제조업체에서는 최종적으로 납품된 부분들을 통합해서 하나의 완성된 차량을 제조하였다. 그러나 이와 같은 개발 방법의 경우 많은 문제점을 내포하고 있었다. 예를 들어 새로운 차량을 개발하거나 제품 성능을 향상시키고자 할 경우 기존에 개발된 소프트웨어나 제품들을 재사용할 수 없고 다시 처음부터 새로 개발해야 했다.

AUTOSAR (AUTomotive Open System ARchitecture)는 이와 같은 문제점들을 해결하기 위해 자동차 생산 업체와 제품 공급 업체들에 의해 만들어진 개방형 표준을 제정하는 단체로서, 차량용 소프트웨어, 사용자 인터페이스 등의 개발을 위한 차량용 소프트웨어 구조를 정의한다. AUTOSAR 소프트웨어 구조는 소프트웨어 컴포넌트와 하위의 하드웨어를 분리함으로써 소프트웨어 컴포넌트의 독립성을 확보함으로써 소프트웨어의 재사용성을 보장하였으며 표준화된 소프트웨어 컴포넌트 인터페이스를 정의함으로써 ECU 간에 데이터 교환이 원활이 이루어질 수 있도록 하였다. 이에 본 논문에서는 AUTOSAR의 기본 개념에 대해 설명하며 아울러 AUTOSAR 기반 차량용 소프트웨어를 위한 NVRAM 관리 서비스 설계 방안에 대해 기술한다.

본 논문의 구성은 다음과 같다. 2장에서는 AUTOSAR의 기본 개념에 대해서 살펴보며 AUTOSAR 시스템 서비스의 구조와 역할에 대해 상세히 살펴본다. 그리고 3장에서는 AUTOSAR 기반의 차량용 소프트웨어를 개발 과정에서 AUTOSAR기반 시스템 서비스와 NVRAM 관리 서비스 설계 방안에 대해 기술하며 4장에서는 결론을 도출한다.

2. 본 론

2.1 AUTOSAR

AUTOSAR의 핵심 개념은 소프트웨어 컴포넌트와 하단의 하드웨어를 분리함으로써 소프트웨어 컴포넌트의 독립성을 보장하는데 있다. 즉, ECU에 탑재된 마이크로 컨트롤러의 종류나 ECU 종류 혹은 상호 연결되는 소프트웨어 컴포넌트에 상관없이 소프트웨어 컴포넌트 개발에 독립성을 보장하며 이는 곧 개발된 소프트웨어의 재사용을 가능하게 한다. 아래 그림 1은 이와 같은 AUTOSAR의 기본 개념을 함축적으로 보여준다. 그림 1에서 VFB(Virtual Functional Bus)는 차량내의 소프트웨어 컴포넌트 간에 상호 연결을 추상화 한 가상적 통신 버스를 의미한다. VFB를 통해서 서로 다른 소프트웨어 컴포넌트간의 연결, 소프트웨어 컴포넌트와 다른 환경(hardware driver, OS, services 등)간의 연결이 가능하며 결과적으로 소프트웨어 컴포넌트와 하드웨어를 분리하는 기능을 제공한다. 가상적으로 VFB의 관점에서 설계된 결과는 ECU Description과 System Constraint Description 등과 함께

소프트웨어 개발 툴을 통해 구체화 된다. 여기서 ECU Description은 ECU의 기능과 소프트웨어 컴포넌트의 인터페이스와 컴포넌트간의 연결 구조 등에 대한 세부 정의를 기술하는 문서이며 System Constraint Description은 시스템의 전반적인 제약 사항을 기술하는 문서이다. 이 단계에서 소프트웨어 컴포넌트를 적절한 ECU에 매핑하고 이와 관련된 세부 정보들이 앞에서 언급한 여러 문서에서 기술된다.

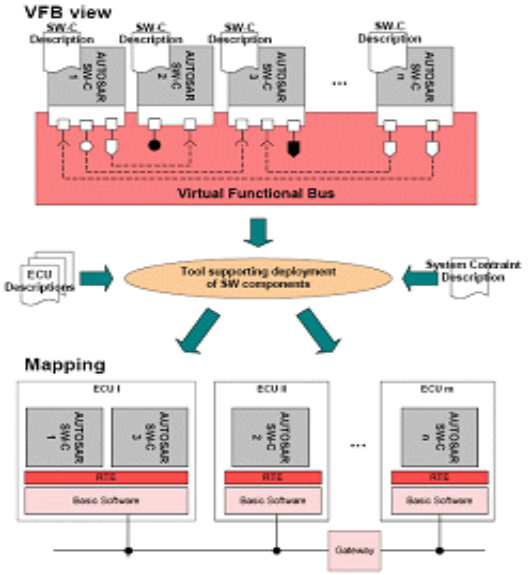

그림 1 AUTOSAR 기본 개념

이 과정에서 소프트웨어 컴포넌트 간의 연결이나 소프트웨어 컴포넌트와 다른 기능 블록간의 연결 기능을 수행했던 VFB는 RTE(Run Time Environment)와 Basic Software로 구체화 된다.

AUTOSAR 소프트웨어 구조에서 핵심적인 요소로 컴포넌트, 포트 그리고 인터페이스가 있다. 컴포넌트는 소프트웨어의 내부 기능을 캡슐화 하며 소프트웨어 재사용의 단위가 된다. 이와 같은 컴포넌트는 잘 정의된 포트를 가지고 있으며 이를 통해 컴포넌트는 다른 컴포넌트들과 상호 연결될 수 있다. 컴포넌트의 포트에 의해 제공되거나 요구되는 서비스나 데이터를 정의하기 위해 인터페이스 개념이 도입되었으며 AUTOSAR 인터페이스는 Client-Server 인터페이스이거나 Sender-Receiver 인터페이스일 수 있다[3].

2.2 시스템 서비스

그림 2는 AUTOSAR ECU 소프트웨어 구조를 보여준다. AUTOSAR 소프트웨어는 ECU에 매핑된 AUTOSAR 소프트웨어 컴포넌트로 구성되며 소프트웨어 컴퍼넌트 간에 연결은 RTE를 통해 이루어진다. Basic Software는 소프트웨어 컴포넌트에 다양한 서비스를 제공하는 표준화된 소프트웨어 계층으로 서비스 (Service), 통신(Communication), OS(Operating Systme), Microcontroller Abstraction, ECU Abstraction 그리고 Complex Device Driver(CDD) 등으로 구성된다. 본 논문에서는 서비스 컴포넌트 중에 시스템 서비스 중 NVRAM 관리자에 대해 자세히 다루도록 한다.

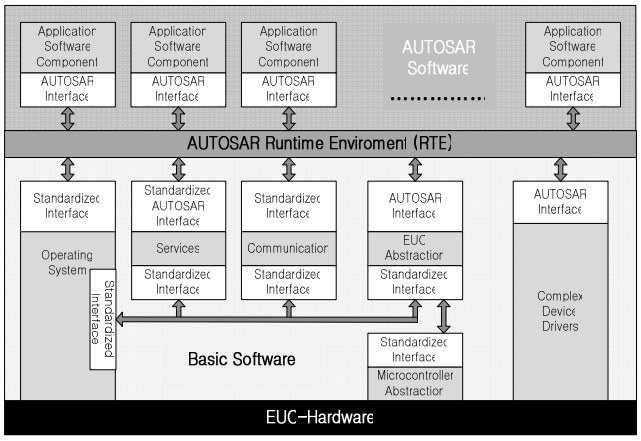

그림 2 AUTOSAR ECU 소프트웨어 구조

시스템 서비스는 모든 계층의 모듈에서 사용되는 기능과 모듈의 그룹이다. 대표적인 예로서, ECU 상태 관리자, Watchdog 관리자, NVRAM 관리자, 실시간 OS와 에러 관리자 그리고 라이버러리 기능 (CRC, interpolation 등) 등이 있다. 시스템 서비스의 일부 기능은 마이크로프로세서 종류, ECU의 하드웨어 혹은 응용 프로그램 등에 영향을 받아 개발되어야 한다.

2.2.1 Mode 관리[4]

모드 관리는 AUTOSAR의 모드를 관리하는 모듈로 ECU State Manager, Watchdog Manager 그리고 Communication Manager 등으로 구성된다. ECU State Manager는 ECU의 상태(OFF, RUN, SHUTDOWN)를 관리하는 Basic Software 모듈이다. 뿐만 아니라 OS와 RTE 등을 포함한 모든 Basic Software 모듈을 초기화, 종료화하는 기능을 수행하며 또한 전력 소모를 고려해서 일시적으로 동작을 멈출 필요가 있는 경우 SLEEP 상태로 천이하기도 하며 wake-up 이벤트에 따라 ECU 상태를 SLEEP 상태에서 다시 RUN 상태로 변경하는 역할을 한다. 그림 3은 ECU State Manager의 상태 천이도를 보여준다[5].

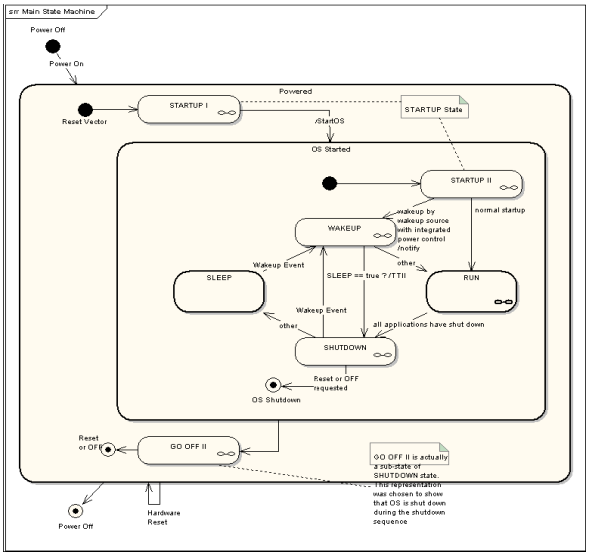

그림 3 ECU State Manager 상태 천이

Watchdog Manager는 응용 프로그램의 시간 제약과 관련된 신뢰성 확보하기 위해 응용 프로그램을 감독하는 역할을 수행한다. 소프트웨어 컴포넌트와 하드웨어의 분리 개념에 따라 Watchdog Manager는 watchdog hardware를 트리거링 시키면서 여러 개의 독립된 응용 프로그램을 감독하는 것이 가능하다. 세부적인 동작으로는 ECU 상위의 개별 응용 프로그램을 감독하며 각각 감독되는 개체에서 발생하는 문제를 처리하고 ECU 상태에 따라 watchdog mode(Off Mode, Slow Mode, Fast Mode)를 변경, 선택할 수 있다[6].

Communication Manager는 통신 사용자로부터 통신 접속 요청을 받아서 이를 처리하는 역할을 수행한다. 세부적인 기능으로는 사용자를 위한 통신 프로토콜 스택 사용을 단순화 하며 여러 개의 독립된 소프트웨어 컴포넌트가 통신 스택을 이용하는 경우 이를 조정하는 역할을 한다[7].

2.2.2 NVRAM 관리자(NVRAM Manager)

NVRAM (Nonvolatile RAM)은 기존 RAM과 동일한 성질을 보이면서 디스크와 같이 비휘발성인 메모리이다.
지금까지 컴퓨터의 저장 장치로는 RAM과 같은 휘발성 메모리와 디스크와 같은 휘발성 저장 장치가 널리 사용되고 있다. 휘발성 저장 장치를 효율적으로 사용하기 위해 버디(buddy)나 슬랩(slab)과 같은 기법들이 개발 되었으며, 비휘발성 저장 장치에 정보를 효과적으로 저장하기 위해서 다양한 파일 시스템들이 개발 되었다. NVRAM은 비휘발성 저장 장치와 휘발성 RAM의 특성을 모두 가지고 있다. 그렇기 때문에 휘발성 저장 장치인 RAM처럼 작은 객체들을 효율적으로 관리할 수 있을 뿐만 아니라 비휘발성 저장 장치인 디스크처럼 큰 파일 데이타도 함께 NVRAM에 효율적으로 저장할 수 있어야 할 것이다. NVRAM 관리자는 자동차 환경 내에서 개별적 요구에 따라 데이터가 저장되고 NV데이터의 유지를 보장하는 서비스를 제공한다. NVRAM 관리자는 FLASH EEPROM 또는 EEPROM의 NV 데이터를 관리하는 것이 가능하다. NVRAM 관리자는 NV 데이터(초기화/읽기/쓰기/제어)의 유지와 관리를 위한 동기적/비동기적인 서비스의 요구를 제공한다[4]. 그림 4는 NVRAM 관리자가 타 서비스들과의 상호관계를 보여준다.

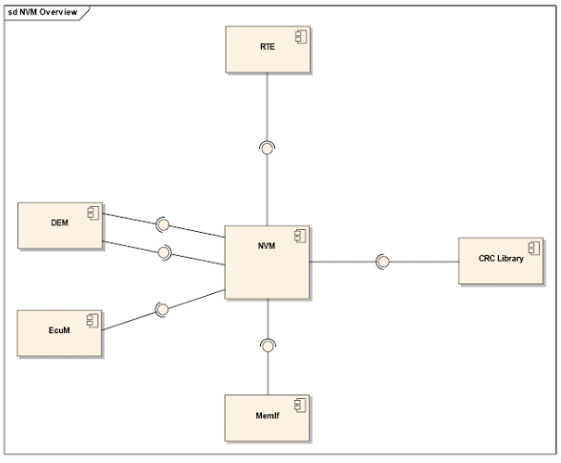

그림 4 NVRAM 관리자의 상호작용

NVRAM 관리자는 크게 startup, shutdown, error recovery 등으로 이루어져 있다. 그림 5는 읽기 과정의 RAM 블록 상태천이를 보여 준다.

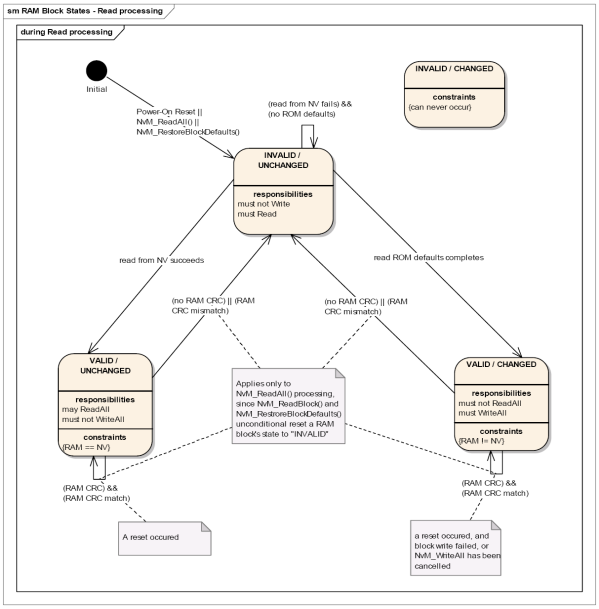

그림 5 RAM 블록 상태 천이

2.2.3 OS[8]

OS 모듈은 멀티 태스크, 실시간 처리, 태스크 동기화 등 운영체제가 제공해야 할 서비스를 제공한다. 기본적으로 AUTOSAR OS는 OSEK OS 표준에 기반으로 하며 일부 기능이 추가되었다.

3. 시스템 서비스 설계

본 논문에서는 시스템 서비스 중에서 모드 관리 모듈의 ECU State Manager를 구현하였으며 구현 환경은 아래 표 1과 같다.

Table 1. 구현 환경

	내 용
Board	Freescale s12x
OS	Microsar OS
OIL	Vector OIL Configurator
IDE	Freescale CodeWarrior v5.7

ECU State Manager를 구현하기 위해서 AUTOSAR OS 기반의 Microsar OS를 이용하였으며 Freescale s12x 보드 상에 구현하였다.

실제로 ECU State Manager의 경우 앞에서 설명한 것과 OS, RTE, Communication 그리고 Watchdog Manager 등과 같은

Basic Software를 초기화, 종료화 시키며 ECU의 상태를 결정하는 역할을 하기 때문에 Basic Software와 하위의 하드웨어의 특성에 따라 개발된다. 따라서 RTE, Communication 등과 같은 다른 소프트웨어 컴포넌트는 기존에 Vector사에서 개발된 모듈을 이용하였다. 장치가 처음 시작할 때 먼저 ECU State Manager의 EcuM_Init() 함수가 호출되며 이 함수에서 OS가 수행되기 전에 필요한 하드웨어 초기화를 수행한다. 그 후, EcuM_MainFunction() 함수가 호출되어 이 함수 내에서 OS가 수행되고 정상적으로 태스크들을 수행하게 된다. 또한 ECU 상태 관리자에 의해 NvM_Init 함수가 Invoke 되어 NVRAM 관리자가 시작된다. 초기 NvM_Init 이 수행되면 NvR_ReadALL이 요청되어서 모든 데이터들을 읽어 들이게 된다. NvR_ReadALL은 ECU 상태 관리자와는 상호배제적으로 수행된다. 이 과정에서 ECU State가 RUN 상태일 때 모든 ECU가 일반적인 정상 동작을 수행하며 ECU의 모든 동작이 마무리 되고 사용자에 의해 종료 명령이 내려지면 ECU의 종료 과정이 진행된다. ECU 상태 관리자에 의해 종료 과정이 진행되게 되면 NvR_WriteALL이 Invoke 되어 NVRAM 관리자는 shutdown과정으로 들어간다. 이 외에 일시적으로 ECU가 동작하진 않지만 ECU가 곧 동작해야 하는 경우를 대비해야 하는 경우 있는데 이와 같은 경우 ECU 상태를 SLEEP으로 변경하여 소비 전력을 최소로 하며 ECU가 빨리 RUN 상태로 변경될 수 있도록 한다. 일반적으로 SLEEP 상태는 ECU에 전력은 공급되지만 코드는 수행되지 않는 상태를 나타낸다. SLEEP 상태의 경우 하드웨어의 특성에 따라 다양한 SLEEP 모드 중에 하나로 동작할 수 있는데, freescale s12x 보드의 경우 WAIT Mode, PSEUDO STOP Mode 그리고 FULL STOP Mode가 있다. 여기서 WAIT Mode는 PLL이 비활성 될 수 있는 상태를 나타내며 PSEUDO STOP Mode는 오실레이터는 활성 되지만 다른 시스템이나 코어 클럭은 멈춘 상태를, FULL STOP Mode는 오실레이터를 포함한 모든 시스템 상태가 멈춘 상태를 나타낸다. SLEEP 상태에서 사용자의 요청이 있는 경우 wake-up 이벤트가 발생하여 WAKEUP 상태를 거쳐 RUN 상태로 천이하게 되는데 본 논문에서는 IRQ에 외부 버튼 입력을 연결하여 wake-up 이벤트가 발생하도록 하였다. 이와 같이 wake-up 이벤트 발생시 외부 버튼으로 구현한 이유는 SLEEP 상태에서 기본적으로 코드가 수행되지 않으므로 물리적인 IRQ를 발생시킴으로써 해결하였다. 검증을 위해 OS 태스크를 생성하고 응용 프로그램을 구현하였으며 ECU 상태를 순차적으로 변화시키도록 하였다. ECU 상태가 SLEEP 상태로 변경된 경우, 사용자가 직접 키 버튼을 누르면 wake-up 이벤트가 발생하여 ECU가 RUN 상태로 변경될 수 있도록 하였다. 이와 같이 구현된 ECU State Manager는 Vector에서 기존에 개발된 다른 소프트웨어 컴포넌트에 연결되어 정상적으로 동작함을 확인할 수 있었다.

4. 결 론

차량에 많은 전기 전자 장치가 사용됨에 따라 자동차 업계에서 소프트웨어의 중요성은 점점 더 증가할 것이다. 이에 따라 소프트웨어를 개발하고 유지, 보수하는데 드는 비용을 줄이고 기간을 단축하는 것이 필요하며 AUTOSAR가 이에 대한 해법이 될 수 있을 것이다. 앞에서 살펴본 것과 같이 AUTOSAR는 여러 기능 모듈이 결합되어 구성되어 있으며 많은 기능을 포함하고 있다. 이에 각 모듈 별 각 계층별 기능과 요구 사항 분석하여 AUTOSAR를 기반으로 한 개발 도구의 개발이 시급하다. 이에 본 논문에서는 AUTOSAR 기본 개념과 시스템 서비스 및 NVRAM 관리자에 대해 살펴보았고 이를 실제 보드 상에 구현한 부분을 확인하였다. 추후 연구에서는 구현된 모듈 외에 추가적인 모듈을 구현하여 성능을 검증, 수정 보완하고자 한다.

Acknowledgement

본 연구는 교육과학기술부에서 지원하는 기관고유사업비로 수행하였습니다.

참고문헌

[1] Bernd Hardung, Thorsten Kolzow and Andreas Kruger, "Reuse of software in distributed embedded automotive systems", International Conference On Embedded Software, pp. 203-210, 2004.

[2] 유우석, 박지용, 유종훈, 김세화, 홍승수, "AUTOSAR에 기반한 차량용 소프트웨어의 구조, "한국자동차공학회 워크숍 및 심포지엄 논문집(전기전자, ITS 부문 심포지움), pp. 60-65, Aug 2006.

[3] AUTOSAR, "AUTOSAR Technical Overview" 2006.

[4] AUTOSAR, "Requirement on Mode Management", 2006.

[5] AUTOSAR, "Specification of ECU State Manager", 2006.

[6] AUTOSAR, "Specification of Watchdog Manager", 2006.

[7] AUTOSAR, "Specification of Communication Manager", 2006.

[8] OSEK, "OSEK/VDX Operating System Version 2.2.3," 2005.