Transactions of KSAE, Vol. 34, No. 5, pp.609-618 (May, 2026)

pISSN 1225-6382 / eISSN 2234-0149
DOI http://dx.doi.org/10.7467/KSAE.2026.34.5.609

〈응 용 논 문〉

차량 Android AVN 시스템에서 Suspend to RAM 기능 적용을 위한 계층별 사례 연구

이 기 범[*1] · 임 튼 튼[2] · 장 하 다[3] · 이 현 수[4]

현대모비스 BSP개발팀[*1] · 현대모비스 공용HW개발팀[2] · 현대모비스 인포MCU APP팀[3] · 현대모비스 네이티브미들웨어팀[4]

Layered Case Study for Applying Suspend to RAM Feature in Vehicle Android AVN System

Kibum Lee[*1] · Teunteun Lim[2] · Hada Jang[3] · Hyunsoo Lee[4]

[1]*BSP Development Team, Hyundai Mobis Company, 17-2 Mabuk-ro 240beon-gil, Giheung-gu, Yongin-si, Gyeonggi 16891, Korea*
[2]*Common HW Development Team, Hyundai Mobis Company, 17-2 Mabuk-ro 240beon-gil, Giheung-gu, Yongin-si, Gyeonggi 16891, Korea*
[3]*Infotainment MCU App Team, Hyundai Mobis Company, 17-2 Mabuk-ro 240beon-gil, Giheung-gu, Yongin-si, Gyeonggi 16891, Korea*
[4]*Native Middleware Team, Hyundai Mobis Company, 17-2 Mabuk-ro 240beon-gil, Giheung-gu, Yongin-si, Gyeonggi 16891, Korea*

(Received 17 November 2025 / Revised 30 December 2025 / Accepted 6 January 2026)

Abstract : In recent years, software complexity in vehicle Android AVN(Audio, Video, and Navigation) systems has increased due to rising customer demands. As a result, booting speed has significantly been slower compared to previous systems. This paper is presenting a case study on the implementation of Suspend to RAM(STR) functionality in vehicle Android AVN systems, focusing on its application across various layers. STR reduces power consumption, and enables rapid resumption, thereby enhancing user experience. This study is aimed at exploring the technical principles, implementation methods, and performance evaluation of STR in a real vehicle environment, examining each layer and key considerations.

Key words : Suspend to RAM(절전모드), AVN(오디오, 비디오, 내비게이션), Linux(리눅스), Android(안드로이드), Current consumption(소모전류), Suspend(일시정지), Resume(재개), VCPU(차량용 중앙처리장치), Framework(프레임워크)

1. 서 론

최근 차량용 Android AVN(오디오, 비디오, 내비게이션) 시스템에서는 고객의 요구사항 증대 및 SW복잡도가 점점 증가하는 추세이다. 차량용 AVN 제품의 SW 복잡도가 증가하면서 과거 제품에 비해 부팅속도 또한 현저하게 느려지고 있다. 본 논문은 차량용 Android AVN 시스템에서 Suspend to RAM(이하 STR) 기능의 구현에 대한 사례 연구를 다루며, 다양한 계층에서의 실제 적용을 중심으로 기술한다.

STR 기능은 전력 소비를 줄임과 동시에 빠른 재개를 가능하게 함으로써 사용자 경험을 향상시키는 중요한 기능으로 다양한 모바일 제품군에 걸쳐 광범위하게 연구되고 적용되어 왔다. 그러나 차량용 AVN 시스템에 적용된 사례는 매우 제한적이다. 기존 연구는 주로 STR 기능의 이론적인 배경에 집중되어 있으며, 차량 환경 특유의 하드웨어 구조와 소프트웨어 계층을 고려한 분석은 부족하다.

본 논문은 이러한 분석을 바탕으로, 실제 차량용 Android AVN 시스템에서의 STR 기능 적용을 위한 HW(Hardware), VCPU(Vehicle-CPU), BSP(Board Support Package), 프레임워크 계층에서의 연구 및 고려사항을 기술하며, 콜드 부트 대비 STR 기능 적용을 통한 부팅시간 단축과 STR 상태에서의 소모전류 측정치를 정량적으로 기술한다.

---

*A part of this paper was presented at the KSAE 2025 Fall Conference and Exhibition
*Corresponding author, E-mail: kibum.lee@mobis.com

본 논문의 구성은 다음과 같다. 2장에서는 STR에 관련된 연구를 살펴보고, 3장에서는 STR의 정의와 STR의 사용 목적 그리고 STR기능의 핵심 목표를 설명한다. 4장에서는 STR 기능을 이해하기 위한 사전지식에 대해서 설명한다. 5장에서는 차량용 안드로이드 소프트웨어에서 STR 동작의 시퀀스를 설명하며 6장에서는 HW, VCPU (MCU), BSP, 프레임워크 등 각 Layer에서 STR의 구현 및 고려 사항을 기술한다. 그리고 7장에서는 STR 적용 후 실측 데이터를 제시하며 8장에서 결론을 내린다.

2. 관련 연구

본 장에서는 STR에 관한 기존 연구를 검토하고, 그 한계점을 분석한 후 본 논문의 차별성을 제시한다. 기존 연구들은 주로 Linux 커널에서의 전원 관리(Power management) 개념, 장치 수준(Device-level) 및 시스템 수준(System-level) 전원 관리 기법, ACPI(Advanced Configuration and Power Interface) 기반 절전 상태(S0~S4) 설명, STR 기능의 중요성, 그리고 플랫폼별 전원 관리 전략 등을 폭넓게 다루고 있다.[4-6)]

그러나 이러한 연구들은 대부분 Linux 커널 환경에 국한되어 있으며, STR의 이론적 배경에 집중하는 경향이 있다. 특히, 차량 환경 특유의 하드웨어 구조 및 소프트웨어 계층을 고려한 분석은 이루어지지 않았으며, 차량 환경에 특화된 Android 기반 AVN 시스템에서 STR 관련 연구는 발견되지 않았다. 이에 본 논문에서는 차량용 AVN 제품에서 STR 기능을 적용하기 위한 각 Layer별 사례연구를 수행하고, 이를 기반으로 차량 환경에 적합한 STR 구현 방안을 분석하여 제안한다.

3. Suspend to RAM

본 장에서는 STR의 정의 및 사용 목적 그리고 STR 기능의 핵심 목표를 설명한다.

3.1 STR의 정의

STR 은 전자 기기, 특히 컴퓨터에서 사용되는 저전력 모드로 진입하는 동작을 말하며, 흔히 STR과 Suspend state라는 용어는 혼용되어 사용된다. 이 모드에서는 시스템의 대부분의 구성 요소가 꺼지지만, 시스템 구성 정보, 열려 있는 애플리케이션, 활성 파일 등의 정보는 주 메모리(RAM)에 저장되어 있고 주 메모리의 전원은 끄지 않은 채로 유지된다.

시스템에 전원이 인가되면 시스템은 Working state에 들어가 다양한 애플리케이션을 실행하며 동작한다. 시스템에 전원이 꺼지면 거의 전원을 소비하지 않지만, 애플리케이션도 마찬가지로 실행하지 않는다. Fig. 1에서 표현한 것과 같이, Working state작업 상태와 Power off(전원 꺼짐) 상태 사이에는 Suspend state 라는 상태가 있다. Suspend state는 STR 기능의 동작으로 진입 되는 상태를 말한다. Suspend state는 다음 과 같은 속성을 가진다.[6)]

1) 소모되는 에너지가 Working state 보다 절약된다.
2) CPU는 코드를 실행하지 않는다.
3) 모든 I/O 장치는 저전력 상태에 있거나 전원이 차단된다.
4) 모든 애플리케이션의 실행되어진 마지막 상태가 유지된다.

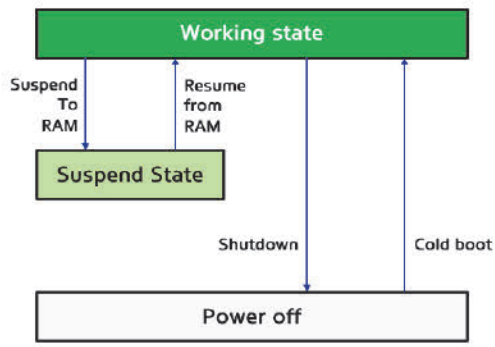

Fig. 1 System state

3.2 STR 적용의 목적

차량 환경에서 사용자가 운전석에 착석하여 AVN기능을 사용하기 위해서는 AVN시스템의 콜드 부팅이 완료되기까지의 긴 대기 시간이 필요하다. 그래서 차량환경에서의 STR의 적용 목적은 다음과 같다.[6)]

1) 에너지 절약

STR 모드는 시스템의 대부분의 구성 요소를 비활성화하고, 주 메모리만 최소한의 전력으로 유지한다. 이를 통해 전력 소비를 크게 줄일 수 있다.

2) 빠른 재개

STR 모드에서 시스템을 재개하면, 이전 상태로 빠르게 복원된다. 이는 시스템이 완전히 종료되지 않고, RAM에 저장된 데이터를 유지하기 때문이다.

3) 작업 연속성 유지

STR 모드는 시스템의 현재 상태를 RAM에 저장하므로, 작업 중인 어플리케이션과 파일을 그대로 유지할 수 있다. 이를 통해 작업의 연속성을 보장한다.

3.3 STR 기능의 핵심 목표

STR기능이 적용되지 않은 제품과 다르게 STR 이 적용되는 제품은 다음과 같은 목표를 가지고 있다.[15)]

1) Working state에서 Suspend state로 STR 동작이 완료되어야 한다.
2) Suspend state에서 소모전류를 최소화하여야 한다.
3) Suspend state에서 다시 Working state로 천이 되는 Resume 시간이 최소화가 되어야 한다.

4. 사전 지식

본 장에서는 차량용 배터리 기준으로 배터리 용량과 소모전류의 사전지식을 설명하고 차량용 제품에서 STR 기능이 적용되지 않은 제품과 STR이 적용된 제품의 동작 차이점에 대한 설명 및 모바일 제품과 차량용 제품의 차이점에 대해서 설명한다.

4.1 배터리용량과 소모전류의 사전지식

차량에 장착되어지는 배터리의 용량과 차량 방전까지의 시간은 다음과 같다. 소형 승용차 기준으로 가장 작은 배터리 셀의 용량 크기는 Fig. 2에서 표현한 것과 같이, 32 Ah이다. AVN 유닛이 STR 동작 시 100 mA를 소모한다고 가정하고 차량의 시동이 꺼진 주차상태에서 AVN만 차량 배터리를 소모된다고 가정한다. 아래와 같은 식으로 완충 상태의 배터리 셀 기준으로 320시간이면 차량의 배터리는 방전이 된다(식 (1)). 단, 이는 이론적 계산으로 실제 차량 환경과는 차이가 있다. 실제 차량에서는 AVN 뿐만 아니라 ECU(Electronic Control Unit) 및 제어기 모듈 등 다양한 상시전원 부하가 존재하므로 방전 시간은 더 짧아진다. 따라서 본 식은 배터리 용량과 방전 시간의 관계를 설명하기 위한 단순화된 예시이다.

$$32{,}000\ \text{mAh} / 100\ \text{mA} = 320\ \text{h} \quad (1)$$

Product Name	Voltage(V)	Capacity (AH)	
		5HR	20HR
GB 40L	12	32	40
GB 40R	12	32	40
GB 40AL	12	32	40
GB 50L	12	40	50
GB 60R	12	48	60
GB 60AL	12	48	60
GB 80L	12	64	80
GB 80R	12	64	80
GB 90L	12	72	90
GB 90R	12	72	90
GB 95R	12	72	90
GB 100L	12	80	100
GB 100R	12	80	100
GB 100BR	12	80	100

Fig. 2 Some vehicle battery products

결론적으로, 전자기기에 탑재된 STR 기능에서 소모전류 최적화는 시스템 품질을 결정짓는 중요한 요소이다. STR이 동작하여 Suspend state 상태에 진입하면 소모 전류가 최저치에 도달한다. 이 최저치를 Rock bottom이라고 하며 이 Rock bottom을 얼마나 더 낮추는지가 제품의 품질을 결정짓는 기준이 된다. Fig. 3은 시간 축을 기준으로 소모 전류를 표현한 그림이며, Rock bottom이 높을수록 시간이 지남에 따라 소모 전류의 누적 량을 쉽게 이해할 수 있도록 시각화 하였다.

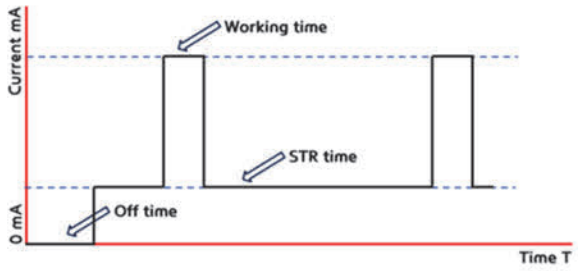

Fig. 3 Current consumption on a time axis

4.2 STR 미적용/적용 제품의 동작 차이점

이미 앞서 소개했지만 Fig. 4와 같이 STR이 미 적용된 AVN 유닛은 CPU의 전원을 VCPU에서 제어하고 사용자의 차량 전원 제어 혹은 Remote 제어가 있는 경우 VCPU는 CPU의 전원을 인가하고 CPU는 콜드 부트를 수행한다. 안드로이드 기반 AVN 시스템은 Bootloader 초기화, 커널 로딩, Zygote 및 System server 구동 등 복잡한 부팅 시퀀스를 거치므로, 콜드 부트가 완료되기까지 약 25초의 시간이 소요된다. AVN 제품에 STR 기능이 적용이 되면 VCPU는 STR 이벤트를 CPU 에게 전달하고 CPU는 STR을 수행한다. 마찬가지로 사용자의 차량 조작 시 VCPU는 GPIO(General Purpose Input/Output)를 이용하여 Suspend state 상태에서 Wake-up을 시키고 CPU는 그 즉시 Resume from RAM 동작으로 Working state 로 천이 한다.

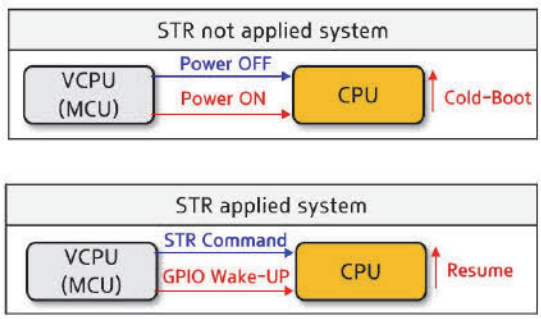

Fig. 4 Difference in operation with/without STR

4.3 모바일 제품과 차량용 제품의 동작 차이점

현재, 모바일 제품들은 대부분 STR 기능을 탑재하고 있으며, 모바일 제품의 STR 동작은 사용자가 물리 전원 버튼을 Toggle하는 동작으로 BSP layer에서 시작하여 Framework layer를 거쳐서 다시 BSP layer로 STR 동작이 진행된다. 최종적으로 BSP에서 STR 동작을 수행하고 STR이 완료된다.

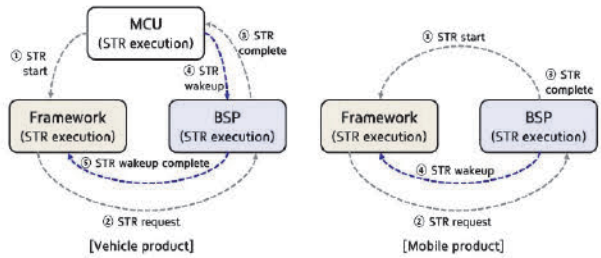

Fig. 5 Differences in STR operation between mobile and vehicle products

그러나, 차량용 AVN제품은 사용자의 차량 전원Off 동작으로 VCPU에서 STR 동작을 수행하고 최종적으로 다시 VCPU에서 STR 동작을 완료한다. 각 제품의 동작은 그림은 Fig. 5와 같다. Fig. 5에서 쉽게 이해할 수 있듯이 차량용 제품에서 STR의 동작은 모바일 제품에 비해 훨씬 복잡하며, 각 Layer별로 통신과 동작이 명확하게 이루어져야 하며, 통신과 동작이 잘못될 경우 차량 제품의 배터리 방전 문제와 품질에 직접적인 영향을 미친다.

5. 차량용 안드로이드 OS 가 적용된 SW 구조에서의 STR 동작 시퀀스

본 장에서는 차량용 안드로이드 OS(이하 AAOS)가 탑재된 소프트웨어 구조에서의 Suspend/Resume 시퀀스를 VCPU, Framework, BSP layer에서의 동작과 각 Layer간의 커뮤니케이션을 기준으로 설명하도록 하며, 이에 대한 도식화된 그림은 아래의 Fig. 6과 같다.

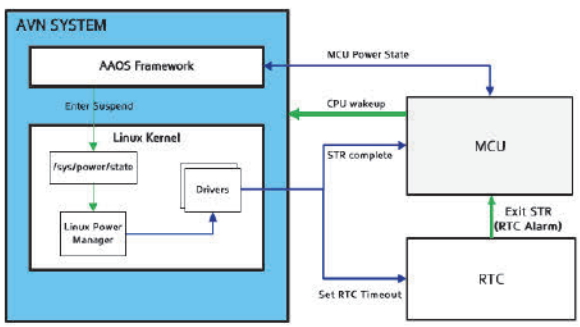

Fig. 6 Suspend / Resume sequence

5.1 STR Suspend 시퀀스

Suspend 시퀀스는 사용자의 차량 전원 Off 동작을 기점으로 시작되며, VCPU, AAOS 프레임워크, BSP를 거쳐 최종적으로 VCPU에 의해 완료된다. 각 계층에서 수행되는 구체적인 동작 시퀀스는 다음과 같다.

1) 사용자의 차량 전원 Off 동작
2) VCPU가 STR 명령을 Framework layer에 전달
3) Framework layer는 모든 Application에 STR 이벤트 전달
4) Framework에서 Wake-up 하기위한 RTC alarm set
5) "/sys/power/state" path를 통해 "mem" 명령으로 BSP layer에 STR 명령 전달
6) BSP layer에서 Filesystem sync 및 Application freezing
7) BSP layer에서 모든 커널 Device driver suspend callback function 수행하며 소모전류 최적화
8) GPIO를 통해 VCPU에게 STR 완료 통지
9) VCPU Layer에서 STR 완료 통지 수신 후, CPU 전원 과 주 메모리 전원을 제외한 주변장치의 전원을 차단

5.2 STR Resume 시퀀스

Resume 동작은 Suspend 동작의 역순으로 사용자 또는 원격을 통한 시동 On 시 VCPU에 의해 CPU가 Wake-up 되며 Resume 동작의 순서는 다음과 같다(Fig. 6, Fig. 12참조).

1) 사용자의 차량 조작 시 VCPU의 Wakeup Interrupt Active
2) VCPU Wake-up Reason 판단 후 CPU resume 여부 판단
3) VCPU에서 GPIO를 이용하여 CPU Wake-up
4) BSP layer에서 모든 커널 Device driver resume callback function 수행하며 STR 이전 상태로 되돌림
5) BSP layer에서 Framework으로 Resume의 결과값을 전달
6) AAOS Framework layer에서 각 어플리케이션으로 Suspend exit 이벤트 전달
7) Suspend exit 이벤트를 받은 모든 어플리케이션 및 Framework 서비스는 작업 연속성 유지된 상태로 동작

6. Layer 별 STR 의 구현 및 고려 사항

본 장에서는 퀄컴 8255 칩셋의 CPU 및 VCPU 하드웨어를 기반으로 한 STR 구현 사례를 제시하고, HW, VCPU, BSP, Framework 계층에서의 실제 제품 적용을 중심으로 개발 과정과 고려 사항을 설명한다.

6.1 HW layer 에서 STR 구현을 위한 블록 설계 고찰

이 절에서는 STR 기능의 효율적 구현을 위해 하드웨어 계층에서 요구되는 설계 제약 및 특성을 고찰한다. 특히, 시스템 대기전력 최소화를 위한 Suspend 모드에서의 필수 전원 공급 블록과 각 블록의 절전 기능 적용 방안을 설명한다.

6.1.1 STR 상시 전원 공급 블록 설계

STR 상태에서 소모전류를 최소화하기 위해서는 차량 배터리로부터 전원이 지속적으로 공급되어야 하는 하드웨어 블록을 명확히 식별해야 한다. 주요 블록은 다음과 같다.

1) 시스템 상태 유지를 위한 CPU, DRAM, UFS(Universal Flash Storage)
2) 차량의 Accessory 및 Ignition 신호 입력 부
3) Wake-up 하기 위한 CAN(Controller Area Network) 및 Ethernet
4) Suspend 시간 설정을 위한 RTC
5) Resume 조건 판단 및 전원 모드 전환을 위한 MCU (Micro Controller Unit)

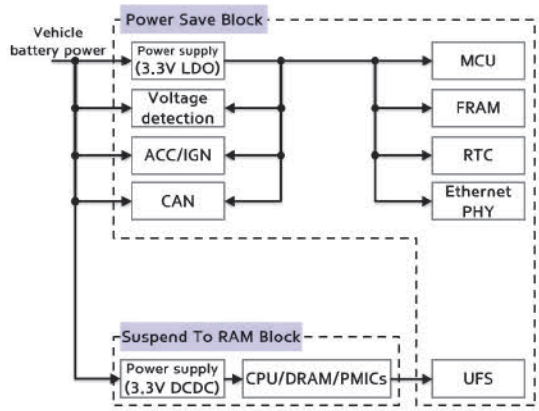

Fig. 7 STR mode power supply hardware block

각 HW 블록은 STR 상태에서 대기전력을 최소화 하기 위해 절전모드(Power save)기능이 고려되어 적용되었으며, 관련 구조는 Fig. 7, Fig. 8에 나타낸다.

HW	Power mode
CPU	Suspend to RAM
DRAMs	Self-Refresh (Suspend to RAM)
UFS (storage memory)	Standby
MCU	Deep Sleep
FRAM (storage memory)	Standby
CAN	Standby
Ethernet PHY	(Deep) Sleep
RTC	Normal (I2C bus inactive, CLK disabled)

Fig. 8 STR mode power save

퀄컴 8255 SIP(System In Package) 칩셋의 경우, Fig. 9과 같이 STR 모드에서의 전력 소모(Pout_SIP)는 최대 38 mW (레퍼런스 보드 기준)로 측정되며, 실제 제품 적용시 내부 불필요 전원 블록의 차단을 통해 최대 34.09 mW 수준으로 감소할 수 있을 것으로 예상된다. 이에 따라 STR 블록의 입력 전력(Pin_DCDC)은 칩셋의 소비 전력에 따른 전원공급장치의 효율을 감안했을 때 최대 40.2 mW로 추정된다. 또한, 전원 공급 장치 내부 회로 구동을 위한 전력(Pin_Vbias)은 최대 3.55 mW, Power save 블록의 전력 소모는 최대 12 mW 이하로 분석된다. 결과적으로 STR 모드에서의 총 전력 소모(P_total)는 최대 55.74 mW로 예측된다.

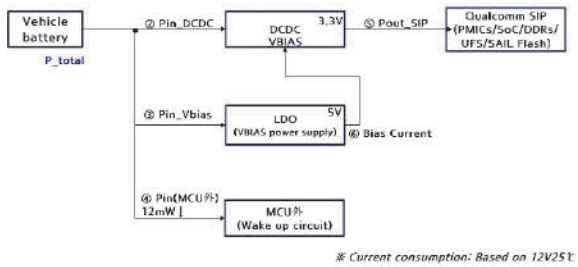

Fig. 9 STR mode current consumption prediction

6.2 VCPU(MCU) layer 에서 STR 구현을 위한 고려사항

VCPU layer에서 STR 미적용 AVN의 Power 시퀀스와 STR 기능을 위한 VCPU 고려사항에 대해서 설명한다.

6.2.1 VCPU 에서 STR 미적용 AVN 의 Power 시퀀스

STR 이 적용되지 않은 AVN시스템의 전원 제어는 Fig. 10과 같이 VCPU에 의해 수행된다. 시스템은 차량의 전원과 네트워크 상태 변화에 따라 여러 단계로 구분되며, 각 단계는 시스템의 전환 상태를 명확히 나타낸다. 각 단계에 대한 세부 설명은 다음과 같다.

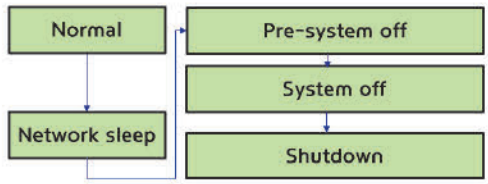

Fig. 10 VCPU power sequence without STR

1) Normal 단계: 차량의 전원 및 네트워크가 모두 활성화되어 AVN 시스템이 동작하는 상태를 의미한다.
2) Network sleep 단계: 차량의 Ignition 및 Accessory 전원이 모두 Off된 이후 진입하는 단계로, 네트워크 슬립 상태를 대기하는 구간이다.
3) Pre-system off 단계: 차량의 전원 및 네트워크가 완전히 종료된 이후 진입하는 단계로, CPU가 제어하

는 주변 장치(Peripheral device) Off를 준비하는 과정이다. 시스템 종료를 위한 준비 단계로, 각 장치의 안전한 종료가 수행된다.

4) System off 단계: 모든 주변 장치의 사용을 종료하는 단계이다. 이 단계에서는 시스템의 주요 기능이 모두 정지되며, VCPU는 CPU로부터 최종 종료 명령 수신을 위한 대기 상태로 전환된다.

5) Shutdown 단계: CPU로부터 종료 요청을 수신한 이후, VCPU는 CPU의 전원 및 주변 장치의 전원을 차단하고 Sleep 모드로 진입한다. 이로써 전체 시스템의 전원 차단 절차가 완료된다.

6.2.2 STR 적용 AVN의 VCPU 설계

STR 미적용 시스템에서는 VCPU가 최종적으로 Shutdown 단계에 진입하여 시스템의 소모 전류를 최소화하도록 설계되었다. 이 단계에서는 CPU의 전원이 완전히 차단되며, VCPU 역시 Sleep 모드로 전환되어 전체 시스템의 전력 소비를 극소화 한다. 그러나 CPU의 STR 기능을 지원하기 위해 VCPU에서 새로운 전원 모드인 STR 단계가 추가되었다. 이 단계는 기존의 전원 차단 방식과 달리, CPU의 백업 전원 및 주 메모리의 전원을 유지하면서도 주변 회로의 전원은 차단하는 방식으로 동작한다. Fig. 11에서 확인할 수 있듯이, 기존의 System Off 단계 이후에 STR 단계로 진입하는 시퀀스가 새롭게 추가되었으며, 이는 VCPU의 전원 관리 전략의 유연성과 효율성을 동시에 확보하기 위한 설계이다.

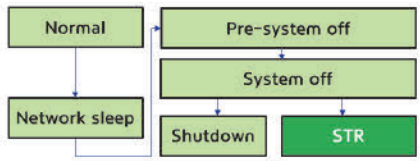

Fig. 11 VCPU power Sequence with STR

VCPU의 Pre-System OFF 및 System OFF 단계에서는 CPU로부터 처리 완료 상태를 정상적으로 수신해야 한다. 그러나 시스템 오류 또는 통신 장애로 인해 해당 정보를 수신하지 못할 경우, 차량 제품의 배터리 방전 문제를 유발할 수 있으며, 이는 제품 품질에 직접적인 영향을 미친다. 따라서 이러한 상황에서 시스템이 Shutdown 단계로 강제 진입하기 위한 대응 방안도 고려되었다.

6.2.3 STR 단계에서 Wake-up 조건

CPU가 STR 단계로 진입한 후, 해당 단계가 정상적으로 완료되면 시스템의 Wake-up 동작은 VCPU에 의해 제어된다. VCPU는 특정 조건이 충족될 경우 시스템을 STR 상태에서 복귀시키며, 이 조건들은 다음과 같다.

1) 차량 전원(Accessory) 상태 변경

차량의 Accessory on 신호가 인가되면, VCPU는 이를 Wake-up 트리거로 인식한다.

2) 네트워크 활성화(Network activation)

차량 내 통신 네트워크가 활성화되면, VCPU는 외부 통신 요구 또는 내부 제어 신호에 따라 복귀 동작을 수행한다.

3) RTC 인터럽트(Real-time clock interrupt)

사전에 설정된 시간에 따라 RTC로부터 인터럽트가 발생하면, VCPU는 이를 Wake-up 트리거로 인식한다.

이와 같은 Wake-up 조건은 시스템의 저전력 상태 유지와 동시에 신속한 복귀를 가능하게 하며, 차량 환경에서의 전력 효율성과 사용자 경험을 동시에 제공한다.

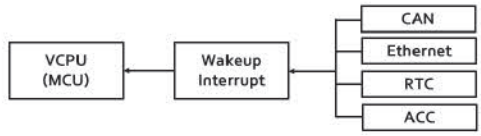

Fig. 12 Source lists for VCPU to Wake-up

Fig. 12은 AVN 시스템이 STR 상태에서 VCPU를 깨우기 위한 소스 목록을 도식화 한 것이다.

6.3 BSP layer 에서 STR 구현을 위한 고려사항

BSP layer에서 STR 기능 Enable 방법, 고려사항, 그리고 리눅스 Power manager SW architecture에 대해서 설명한다.

6.3.1 리눅스 커널에서 STR Feature enable

리눅스 커널에서 STR 기능을 Enable 하기 위해서는 다음과 같은 CONFIG enable을 해야 한다.[14] Fig. 13에서 표현한 것과 같이 아래 3개의 CONFIG를 Enable 하고 AVN 시스템의 소스를 전체 빌드를 해야 한다.

1) CONFIG_PM

2) CONFIG_SUSPEND

3) CONFIG_PM_SLEEP

```
msm-kernel/kernel/power/Makefile

..
obj-$(CONFIG_PM)            += main.o
obj-$(CONFIG_SUSPEND)       += suspend.o
obj-$(CONFIG_SUSPEND)       += wakeup_reason.o
..
msm-kernel/drivers/base/power/Makefile
..
obj-$(CONFIG_PM)            += sysfs.o generic_ops.o common.o qos.o runtime.o wakeirq.o
obj-$(CONFIG_PM_SLEEP)      += main.o wakeup.o wakeup_stats.o
..
```

Fig. 13 STR core driver file in Linux

여기서 한가지 반드시 주의해야 할 CONFIG가 하나 있다. 차량용 안드로이드 AVN 시스템은 모바일 제품과 다르게 VCPU에서 STR 시작점을 알리고 STR 완료까지 VCPU에서 완료한다. BSP에서 협의된 프로토콜 및 통신을 무시하고 스스로 Auto STR이 동작하게 되면 심각한 품질 문제를 초래하게 된다. 하여 CONFIG_PM_AUTOSLEEP은 반드시 Disable 해야 한다.

6.3.2 리눅스 커널의 Power Manager core

리눅스 커널은 정의된 인터페이스를 통해 플랫폼과 상호작용하며, 특정 플랫폼에서 필요한 Low level suspend 및 Resume을 수행하는 플랫폼 드라이버를 포함하고 있다. 이 드라이버들은 struct dev_pm_ops를 통해 Callback세트를 제공한다.

```
msm-kernel¥include¥linux¥pm.h

struct dev_pm_ops {
        ..
        ..
        int (*suspend)(struct device *dev);
        int (*resume)(struct device *dev);
        ..
        ..
};
```

Fig. 14 Struct dev_pm_ops

각 Peripheral low level driver는 dev_pm_ops의 자료구조를 이용하여 Suspend 및 Resume callback을 리눅스 커널에 등록하며 관련 구조는 Fig. 14에 나타낸다.

Framework layer에서 STR 과정이 완료되면, Framework은 "/sys/power/state" path를 통해 "mem" 명령으로 BSP layer에 STR 명령 전달한다. STR 명령을 받은 커널 Power Manager core는 STR 모드에 진입하기 위해 Peripheral low level driver 중 dev_pm_ops의 자료구조를 이용하여 Suspend에 등록된 모든 Driver를 순차적으로 호출한다.

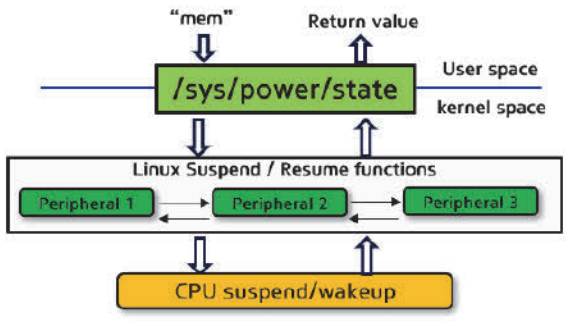

Fig. 15 Linux kernel STR software architecture

Fig. 15는 리눅스 커널 내에서 Suspend 및 Resume callback 의 동작을 도식화해서 표현해주고 있다. 각 Peripheral low level driver는 Suspend 함수에서 소모전류를 최적화하기 위해 Driver가 사용하는 I/O pin을 Low level로 제어해야 한다.

6.3.3 리눅스 커널에서 I/O pin 제어 구조

다음은 Peripheral low level driver의 Suspend/Resume callback function에서 I/O 핀을 제어하는 구조를 설명한다.

```
qcom/proprietary/devicetree/qcom/mobis-pinctrl.dtsi

mobis_gpio96_pins_active: mobis_gpio96_pins_active {
                mux {
                        pins = "gpio96";
                        function = "gpio";
                };

                config {
                        pins = "gpio96";
                        output-high;   // ACTIVE_HIGH
                        input-disable;
                };
};
mobis_gpio96_pins_sleep: mobis_gpio96_pins_sleep {
                mux {
                        pins = "gpio96";
                        function = "gpio";
                };

                config {
                        pins = "gpio96";
                        bias-pull-down;  // PULL DOWN
                        input-enable;
                };
};
mobis_gpio_pinctrl: mobis_gpio_pinctrl {
        compatible = "mobis,mobis_gpio_pinctrl";
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&mobis_gpio96_pins_active>;
        pinctrl-1 = <&mobis_gpio96_pins_sleep>;
}
//-----------------------------------------------------------------------
msm-kernel/drivers/misc/mobis/mobis_pinctrl.c

struct pinctrl          *g_pinctrl;
struct pinctrl_state    *g_state_active;
struct pinctrl_state    *g_state_suspend;

static int mobis_pinctrl_resume(struct device *dev) {
      pinctrl_select_state(g_pinctrl,  g_state_active);
}
static int mobis_pinctrl_suspend(struct device *dev) {
      pinctrl_select_state(g_pinctrl,  g_state_suspend);
}
static int mobis_pinctrl_probe(struct platform_device *pdev) {
      struct device *dev = &pdev->dev;
      g_pinctrl = devm_pinctrl_get(dev);
      g_state_active = pinctrl_lookup_state(g_pinctrl, "default");
      g_state_suspend = pinctrl_lookup_state(g_pinctrl, "sleep");
}
```

Fig. 16 Linux kernel driver I/O control when STR

1) Device tree: 리눅스 커널에서 하드웨어 구성 요소를 설명하는 데이터 구조이며 디바이스 트리는 시스템의 하드웨어를 커널에 알리기 위해 사용된다.
2) Pinctrl subsystem: 핀 제어를 관리하는 서브 시스템이며 I/O 제어를 위한설정을 관리하며, Pinctrl 과 pinctrl_state 구조체를 사용한다.
3) Peripheral low level driver: pinctrl_probe 함수는 디바이스가 초기화될 때 호출되며, 핀 제어 핸들과 상태를 설정한다. 또한, Pinctrl_suspend 함수와 Pinctrl_resume 함수는 시스템이 절전 상태로 전환되거나 복귀할 때 호출되어 핀 상태를 전환한다.

Fig. 16에서 표현한 것과 같이 모든 Peripheral low level driver는 Suspend/Resume callback function에서 소모전류 최적화를 위해 노력해야 한다.

6.3.4 Resume 시간 최적화 고려사항

VCPU가 CPU에게 GPIO Wake-up을 수행한 후CPU는 HW 인터럽트를 전달받는다. 그 이후 모든 커널 Device driver resume callback function 수행하며 STR 이전 상태로 되돌리는 작업을 수행한다. 여기서 BSP layer의 Resume 수행시간을 최적화 하기위해서는 다음과 같은 작업이 필요하다.

1) Interrupt service routine 구간에서는 빠르게 처리하고 그 루틴에서 Return 해야 한다.
2) 각 Device driver resume 함수에서는 sleep( ) 또는 delay( ) 같은 시간 지연 함수를 최적화하여 사용해야 한다.
3) 각 Device driver resume 함수에서 시간을 지연시키는 printk( ) 같은 로그 출력을 최대한 적게 사용해야 한다.

6.3.5 장기간 미사용 차량에 대한 STR 모드 전환 전략

장기간 차량을 이용하지 않는 경우, STR 기능이 적용되지 않은 AVN 제품은 전원이 완전히 차단되어 전력 소비가 극소화 되므로, STR이 동작하는 제품에 비해 장기간 주차 시 소모 전류가 낮다. 이러한 특성으로 인해, CPU에서 STR 진입 직전 RTC칩을 활용하여 장기간 미사용 차량에 대해 일정 시간이 경과하면 STR 상태에서 Shutdown 모드로 전환하는 전략이 필요하다.

6.4 Framework layer 에서 STR 구현을 위한 고려사항

AAOS를 기반으로 한 Framework layer 구조설계의 핵심은 차량의 전원동작을 기준으로 정의된 VCPU에서의 전원동작을 기반으로 AAOS Car power framework을 구동시키는 부분이다. 이를 설명하기 위해 본 절에서는 AAOS Car power state와 전원정책에 대해 설명하고, 이를 토대로 Vehicle HAL(이하 VHAL) interface 설계부분에 대해 기술한다.

6.4.1 AAOS Car power framework

AAOS는 다양한 벤더의 전원 동작 사양을 포괄적으로 지원하기 위하여 총 6개의 Car power state를 범용적이며 직관적인 방식으로 정의하고 있다. 각 Power state는 고유의 전원 정책(Power policy)을 가지며, 해당 정책에는 선택적으로 On/Off 동작이 가능한 하드웨어 및 소프트웨어 구성요소들의 기대 동작 사항이 명시되어 있다. Power state 간의 천이는 어플리케이션 및 VCPU와 VHAL 인터페이스를 통해 Fig. 17과 같이 정의되며, 정의된 방향의 시퀀스 외의 천이 경로는 지원되지 않는다. 이러한 구조는 시스템의 일관성과 안정성을 확보함과 동시에, 다양한 차량 플랫폼 간의 호환성을 보장하는 데 기여한다.

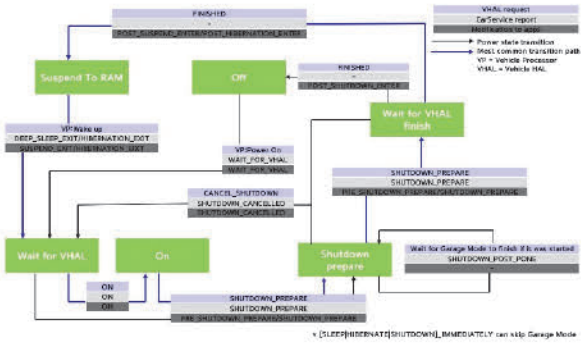

Fig. 17 AAOS car power sequence

6.4.2 AAOS 기반 Framework 구조 설계

앞서 기술한 바와 같이, VCPU의 전원 동작 사항을 기준으로 Car power state의 천이 조건은 다음과 같이 정의할 수 있다.

1) Shutdown prepare 단계

차량의 전원이 On 상태에서 AVN시스템이 동작하는 Normal 단계에서, Ignition 및 Accessory 전원이 모두 Off되어 차량 네트워크가 Sleep 상태로 진입하는 시점을 Shutdown prepare 단계로의 천이 시점으로 정의할 수 있다.

2) Suspend to RAM 단계

차량의 네트워크가 완전히 종료되고, STR 단계로의 진입을 위해 주변 장치 및 시스템 리소스의 해제가 완료되는 System Off 단계 시점을 Suspend to RAM 단계로의 천이 시점으로 정의할 수 있다.

3) Wait for VHAL & ON 단계

STR 상태에서 차량 전원 상태의 변화, 네트워크 활성화, 또는 RTC alarm 인터럽트 발생 등의 조건이 충족될 경우, 시스템은 STR 상태에서 Normal 동작 상태로 복귀하게 되며, 이 시점을 Suspend resume에 의한 천이 시점으로 정의할 수 있다.

6.4.3 추가 사례 연구

Framework 단의 구조 설계에 있어 안정성과 성능을 확보하기 위해 다음과 같은 주요 사례들을 중심으로 면밀한 검토가 필요하다.

1) STR 모드에서 Resume시 VHAL 통신 지연 최소화

STR 모드에서 시스템이 Resume될 때, VHAL과의 통신이 가능한 한 빠르게 재개될 수 있도록 설계되어야 한다. 해당 구간에서 통신 지연이 발생할 경우, 차량의 주요 정

보 업데이트가 지연되어 사용자 경험 저하 및 시스템 신뢰성 저하로 이어질 수 있다.

2) Suspend 진입 안정성을 위한 시스템 조건 이해

시스템이 Suspend 상태로 안정적으로 진입하기 위해서는 Garage mode 조건, Auto-suspend 정책 등 시스템 차원에서 발생할 수 있는 다양한 조건들에 대한 충분한 이해가 선행되어야 한다.

7. STR 미적용/적용 SW 실측 Data

본 장에서는 STR 기능이 적용되지 않은 상태에서의 콜드 부팅 시간과 STR 상태에서의 Resume 시간을 비교하기 위한 수식을 제안한다. 또한, STR 상태에서 시스템이 소비하는 전류에 대한 측정 데이터를 함께 제시한다. 먼저 퀄컴 8255 칩셋 CPU의 HW spec 과 SW spec은 아래 Table 1과 같다.

Table 1 AVN system hardware and software specification

CPU	Kryo™ Gen 6 CPU built on Arm v8.2 Cortex technology 8x Kryo Gold Prime (64bit-ARMv8-A)
CPU PMIC	PMM8650AU (×4) power management IC
CPU Main memory	LPDDR5 (×3) SDRAM 12GByte
CPU Storage	UFS 3.1 gear 4 256GByte
Kernel	Linux kernel 5.15
Android	Android 13 software

7.1 측정 환경 및 반복 측정 기준

측정 환경은 Photo. 1에 나타낸 바와 같이 AVN 전원부에 전류 측정을 위한 멀티미터를 연결한 후, STR 진입한 이후 12 V@25 ℃ 조건으로 소모 전류를 측정하는 방식으로 수행하였다. 또한, 소모 전류 및 Resume 시간 측정 결과의 신뢰성을 확보하기 위해 동일 조건에서 20회 반복 측정을 실시하였으며, 본 논문에서는 그 평균값을 기준으로 작성하였다.

7.2 부팅 시간 측정기준

실측 시간 기준은 Accessory on부터 모니터의 Home 화면이 표시되는 순간까지 시간을 측정했다. 본 논문에서 비교하기 위한 부팅시간은 아래와 같이 제안한다. 콜드 부트의 부팅시간은 Bootloader 로그시간과 Kernel boot 로그시간과 안드로이드 Boot 로그 시간, 그리고 Home 화면이 표시되는 시간의 로그로 합산 가능하다(식 (2)).

$$T_{cold\_boot} = T_{bootloader} + T_{kernel} + T_{android} + T_{home} \quad (2)$$

또한, STR 상태에서의 Resume시간은 VCPU가 CPU를 깨우기 위해 GPIO Wake-up 한 후 커널의 Resume 로그 시간과 Framework의 Resume 로그 시간, 그리고 Home 화면이 표시되는 타임의 로그시간으로 합산한다(식 (3)).

$$T_{STR\_resume} = T_{kernel\_resume} + T_{framework\_resume} + T_{home} \quad (3)$$

또한, 위의 두 식으로부터 STR을 적용함으로써 사용자가 체감하는 시간 차이를 도출할 수 있다(식 (4)).

$$T_{diff} = (T_{bootloader} + T_{kernel} + T_{android}) - (T_{kernel\_resume} + T_{framework\_resume}) \quad (4)$$

7.3 STR 미적용/적용 SW 부팅시간 결과

STR 미적용 상태의 콜드 부트의 부팅 시간과 STR 상태에 Resume 시간을 실측한 시간의 평균값은 아래 Fig. 18과 같다.

아래 Fig. 18에서 볼 수 있듯이, STR 적용 후 Resume 시간은 콜드 부트의 부팅 시간과는 현저히 다른 결과를 보여준다. 이러한 차이는 사용자 경험과 요구를 충족시키는 데 중요한 역할을 한다.

Photo. 1 AVN system STR current consumption

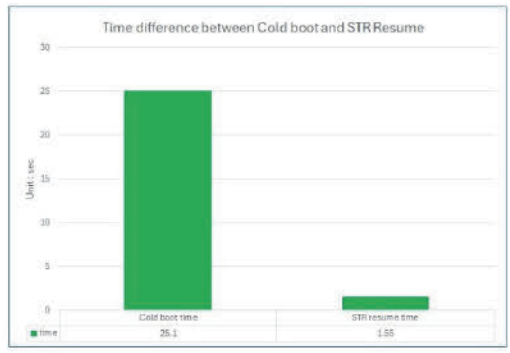

Fig. 18 Cold boot time and STR resume time

7.4 STR 상태의 소모전류

STR 진입한 이후 소모 전류는 Photo. 1에서 볼 수 있듯이, 12 V 입력 전압과 25 ℃ 조건에서 약 3.3 mA의 전류가 소비되는 것을 확인할 수 있다.

8. 결 론

본 논문에서는 차량용 AVN 제품의 STR 기능을 적용하기 위한 각 계층별 사례 연구를 통해 STR 기능을 심도 있게 분석하고 제품에 적용하기 위한 방법을 제안하였다. 이 STR 기능은 국내 양산형 AVN 시스템에 이미 적용되어 상용 생산이 진행 중이다. 이에 따라 향후 다양한 AVN 플랫폼에서 본 기능의 확산이 예상되며, 사용자 경험 향상과 요구 충족에 핵심적인 역할을 할 것으로 기대된다. 또한, 상용 생산 중인 모델 및 향후 출시 예정 모델에 대해 보다 안정적인 STR 기능과 전원 관리 구현을 위해 장시간 미사용 차량에 대한 Shutdown 모드 전환에 관한 추가 연구를 계획하고 있다. 아울러, Hypervisor 기반의 QNX OS와 안드로이드 OS 가 동시에 탑재되는 멀티 OS 기반의 개발 Needs에 대해서도 추가적인 연구를 진행할 예정이다.

References

1) Advanced Configuration and Power Interface Specification, ACPI, http://www.acpi.info.
2) L. Brown, A. Keshavamurthy, D. S. Li, R. Moore, V. Pallipadi and L. Yu, “ACPI in Linux—Architecture, Advances, and Challenges,” Proceedings of the Linux Symposium, Ottawa, Ontario, Canada, July 2005.
3) R. J. Wysocki, “Suspend and Hibernation Status Report,” LWN.net, http://lwn.net/Articles/243404.
4) P. Mochel, Linux Kernel Power Management.
5) P. Mochel, The State of Linux Power Management 2006.
6) A. L. Brown and R. J. Wysocki, “Suspend-to-RAM in Linux.”
7) H. Kim, E. Kim, J. Choi, D. Lee and S. H. Noh, “Building Fully Functional Instant On/Off Systems by Making Use of Non-Volatile RAM,” IEEE International Conference on Consumer Electronics, pp.675-676, 2011.
8) HP, Intel, Microsoft, Phoenix and Toshiba, Advanced Configuration and Power Interface Specification, Revision 5.0, December 6, 2011, https://uefi.org/sites/default/files/resources/ACPI_5.pdf.
9) Sleep States, Linux Kernel Documentation, https://docs.kernel.org/admin-guide/pm/sleep-states.html.
10) GB Battery Document, https://easyskin.godohosting.com/rocket/pdfs/gbbattery.pdf.
11) Android PowerManager API, Android Developers, http://developer.android.com/reference/android/os/PowerManager.html.
12) Automotive Power Management, Android Source, https://source.android.com/docs/devices/automotive/power/power.
13) QAM8255P Data Sheet, Rev. AV, April 3, 2025.
14) https://www.kernel.org.
15) K. Lee, T. Lim, H. Jang and H. Lee, “Layered Case Study for Applying Suspend To RAM Feature in Vehicle Android AVN Models,” KSAE Annual Conference Proceedings, pp.631-641, 2025.