·第二十八届中国科协年会学术论文专题——车规芯片架构改进与可靠性优化关键技术·

基于国产车规MCU的AUTOSAR MCAL架构改进与实现

吴凡　张远　黄小奇　庞博

（小华半导体有限公司，上海 201210）

【摘要】针对国产汽车芯片厂商在汽车开放系统架构（AUTOSAR）微控制器抽象层（MCAL）开发过程中面临的技术积累不足、开发周期长等挑战，提出一种创新性的分层架构设计方案。该方案将MCAL重新划分为3个层次，通过功能解耦和接口标准化，实现对AUTOSAR和传统软件开发工具包（SDK）开发模式的兼容支持。经小华半导体XC277芯片的验证表明：所有模块均以100%的覆盖率通过单元与集成测试，包括语句、分支及修正条件/判定条件覆盖率（MC/DC），且关键任务执行时间达到微秒级，为国产AUTOSAR MCAL的技术创新提供了经过充分验证的解决方案，对提升我国汽车电子软件的自主可控能力具有重要实践意义。

主题词：AUTOSAR　MCAL　分层架构　代码复用　国产XC277

中图分类号：TP311　　**文献标志码：**A　　**DOI:** 10.19620/j.cnki.1000-3703.20250349

Improvement and Implementation of AUTOSAR MCAL Architecture Based on Domestic Automotive-Grade MCUs

Wu Fan, Zhang Yuan, Huang Xiaoqi, Pang Bo

（Xiaohua Semiconductor Co., Ltd., Shanghai 201210）

【**Abstract**】To address the challenges faced by domestic automotive chip manufacturers in developing the Microcontroller Abstraction Layer (MCAL) for the Automotive Open System Architecture (AUTOSAR)—such as insufficient technical expertise and prolonged development cycles, this paper proposes an innovative layered architecture design. This approach redefines the MCAL into 3 distinct layers. Through functional decoupling and standardized interfaces, it achieves compatibility with both AUTOSAR and traditional Software Development Kit (SDK) development models. Validation on Xiaohua Semiconductor's XC277 chip demonstrates that all modules passed unit and integration tests with 100% coverage (including statement, branch, and Modification Condition/Decision Condition Coverage (MC/DC), and critical task execution times reached microsecond levels. This research provides a well-validated solution for technical innovation in domestic AUTOSAR MCAL, holding significant practical implications for enhancing the autonomous and controllable capabilities of China's automotive electronic software.

Key words: AUTOSAR, MCAL, Layered Architecture, Code Reuse, Domestic XC277

【引用格式】吴凡, 张远, 黄小奇, 等. 基于国产车规MCU的AUTOSAR MCAL架构改进与实现[J]. 汽车技术, 2026(2): 1-6.

WU F, ZHANG Y, HUANG X Q, et al. Improvement and Implementation of AUTOSAR MCAL Architecture Based on Domestic Automotive-Grade MCUs[J]. Automobile Technology, 2026(2): 1-6.

1　前言

随着汽车主控芯片配套的软件国产化需求日益凸显，集成电路技术的自主可控已成为我国突破“卡脖子”关键技术的重要国家战略。汽车开放系统架构（AUTomotive Open System Architecture，AUTOSAR）作为主流汽车电子软件架构标准，在汽车电子领域发挥着重要作用。在该架构中，微控制器抽象层（Microcontroller Abstraction Layer，MCAL）是连接硬件和软件的关键桥梁，直接影响AUTOSAR架构与芯片的匹配[1]。

目前，研究主要集中于应用层软件开发、实时运行环境优化、服务层功能实现、电子控制单元（Electronic Control Unit，ECU）抽象层设计以及复杂驱动层开发等。由于国内汽车芯片领域起步较晚，从芯片型号到技术积

累均存在明显差距。在实际开发过程中，芯片升级或外设变更不仅需要重新构建和开发相应的软件组件，还必须严格遵循汽车电子行业标准，进行全面的功能验证和可靠性测试，且验证流程大幅延长了产品的开发周期，因而开发维护成本大幅增长。项目开发中常使用基于传统软件开发工具包（Software Development Kit，SDK）的开发方式，在开发标准AUTOSAR代码的同时，需额外开发维护一套面向传统SDK应用的代码库。因此，亟需开发满足AUTOSAR标准规范的新型架构，同时契合国内汽车电子市场的实际情况，具备良好的可扩展性。

为了解决国产芯片厂商在MCAL开发中技术积累和开发周期方面的挑战，本文深入分析MCAL规范，创新性地提出将MCAL重新划分为硬件层（Hardware Layer，HWL）、知识产权层（Intellectual Property Layer，IPL）和高层（High Layer，HL）3层分层架构的设计方案。同时，使用小华半导体车规级微控制器（Microcontroller Unit，MCU）XC277作为验证平台，采用多项符合ISO 26262的测试方法[2]，对该分层架构进行全面验证。

2　系统构架

AUTOSAR通过标准化和可重用性提升汽车软件质量、降低开发成本，其结构如图1所示。该架构分为3层：应用层负责具体的功能实现；实时运行环境负责上、下层连接；基础软件层提供底层支持，包括服务层、ECU抽象层、MCAL层和复杂驱动层。其中，服务层负责提供系统服务，复杂驱动层主要管理特殊外设。

AUTOSAR通过多层软件架构，实现汽车嵌入式软件系统的逐层解耦，使软件能在不同车型、电子控制单元和芯片间实现复用[3-4]。按照AUTOSAR标准开发的基础软件模块，应用软件仅需微调参数，减少硬件差异化的同时，提升代码质量。本文重点考虑AUTOSAR软件架构的MCAL部分[5]。

作为AUTOSAR软件架构的基础，MCAL层负责与微控制器外设的交互，处理来自上层的控制指令，执行相应的硬件操作，同时将硬件状态反馈给上层。MCAL通过封装底层硬件细节，为上层软件提供统一的标准接口。根据AUTOSAR 4.4.0版本规范[6]，可将MCAL划分为6个驱动组：微控制器（Microcontroller）驱动组、存储器（Memory）驱动组、通信（Communication）驱动组、输入/输出（Input/Output，I/O）驱动组、加密（Crypto）驱动组和无线通信驱动组。各驱动组均包含相应外设的驱动模块，如模数转换器（Analog-to-Digital Converter，ADC）、通用定时器（General Purpose Timer，GPT）、脉宽调制器（Pulse Width Modulation，PWM）和控制器局域网（Controller Area Network，CAN）总线等[7]，其与微控制器外设的映射关系如图2所示。本文重点关注各驱动组中的具体驱动模块实现[8]。

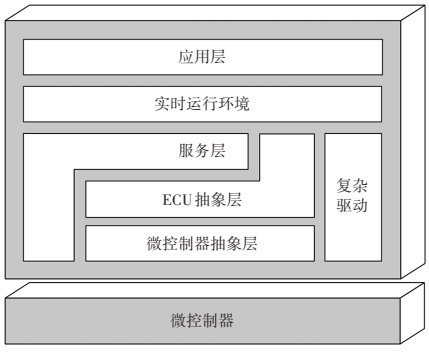
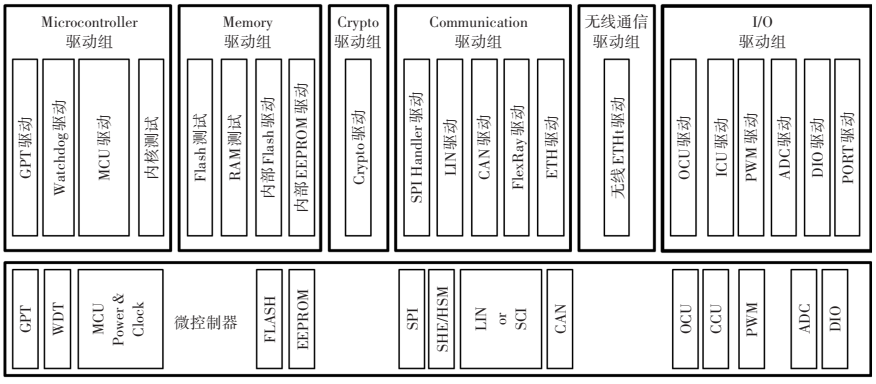


图1　AUTOSAR软件架构

MCAL规范详细定义了驱动模块的功能规格和接口标准，确保各类软件模块能够无缝衔接。该标准化设计使上层软件能够在不同芯片平台间迁移，无需重新开发底层软件，节省开发的成本和时间的同时，提高了芯片选择的自由度。然而，AUTOSAR规范未对其内部功能组件间的交互关系作出具体要求，允许芯片厂商自主设计内部架构。这导致功能组件间缺乏明确的边界划分，组件间存在较强的耦合性，对某一组件的修改可能会造成其他相关组件产生连锁反应，如图3所示。



PORT：Port，端口
DIO：Digital Input/Output，数字输入/输出
ICU：Input Capture Unit，输入捕获单元
OCU：Output Compare Unit，输出比较单元
ETH：Ethernet，以太网
LIN：Local Interconnect Network，本地互联网络
SPI：Serial Peripheral Interface，串行外设接口
EEPROM：Electrically Erasable Programmable Read-Only Memory，带电可擦除可编程只读存储器
Flash：Flash Memory，闪存
RAM：Random Access Memory，随机存取存储器
WDG：Watchdog，看门狗

图2　MCAL驱动组与微控制器外设的映射关系

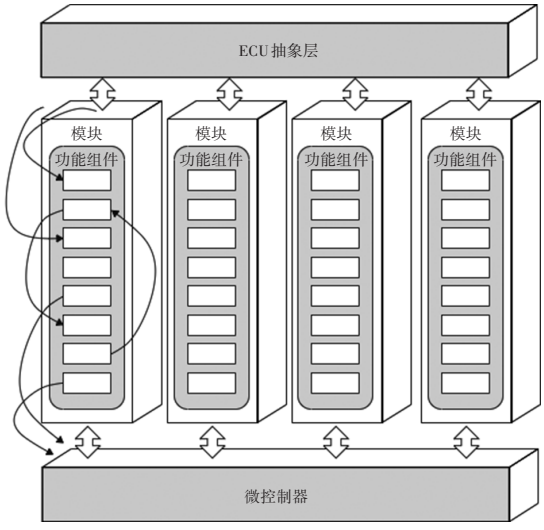


图3　功能组件耦合

3　MCAL分层架构

实践表明，提升代码抽象层次能够有效增强系统兼容性。同种驱动在不同知识产权（Intellectual Property，IP）的实现上存在明显技术差异，随着代码层次向顶层推进，其逻辑结构和功能实现却呈现出高度相似性和一致性。例如，多控制器局域网（Multi-Controller Area Network，MCAN）和灵活控制器局域网（Flexible Controller Area Network，FlexCAN）在MCAL的实现均为CAN，此类趋同现象与AUTOSAR对MCAL层的严格标准化密切相关。

为了突破传统MCAL架构的局限性，本文引入更细致的分层抽象概念，采用抽象水平分层的组织方式，如图4所示。将MCAL重新划分为3个相互关联的层次，从底层到顶层分别为硬件层、IP层和高层。这种设计不仅能够提升代码的可维护性和系统灵活度，还可有效降低硬件变更的影响，大幅节省开发成本。

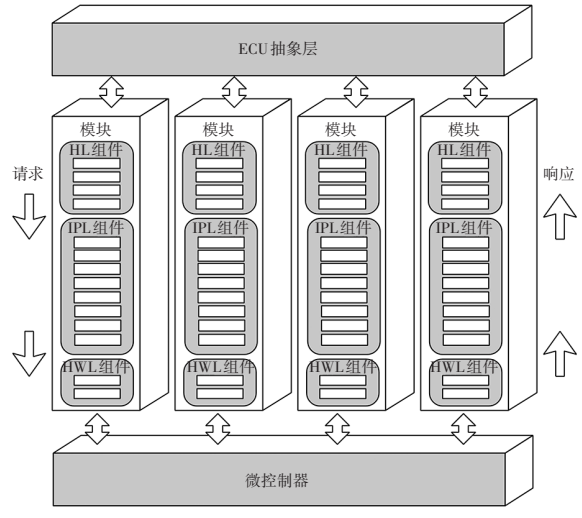


图4　MCAL分层架构

在实际开发中，使用2个主要部署单元方案：将HL作为独立的部署单元，将IPL和HWL整合为一个部署单元，如图5所示。该方式既能屏蔽底层硬件的具体细节，又为后续平台迁移提供便利。在保证各层接口稳定的前提下，各单元均可保持相对独立性。同时，该方案可以灵活替换任意部署单元，如直接使用FlexCAN的底层部署单元替代MCAN的对应部署单元，其他部分无需调整[9]。

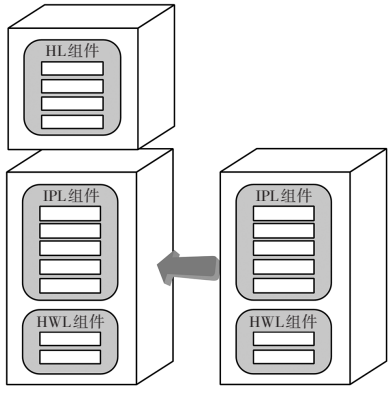


图5　分层部署

3.1　HWL实现策略

HWL是架构的底层，统一管理所有硬件访问操作，并通过规范化的接口确保对寄存器的安全操作。由于不同IP特性的差异较大，该层需要针对具体IP开发专门的软件组件。HWL主要职责包括寄存器抽象和位操作封装：寄存器抽象将硬件寄存器映射为软件可操作的数据结构，并提供读写接口；后者位操作封装常用的位操作函数，包括置位、清零和读取等。

在HWL的实现中，可采用以下策略：

a. 使用内联函数：通过内联函数优化寄存器操作，既保证代码的模块化组织，又能有效降低函数调用产生的性能开销。

b. 统一命名规范：建立统一的HWL命名规范，确保代码的一致性和可维护性。

3.2　IPL实现策略

IPL主要负责对各类硬件IP进行功能封装，提供统一的高层抽象接口。通过抽象和封装IP特征，使其不受外部需求变化影响。IPL在抽象程度方面介于底层和顶层间，尽管各IP的内部实现不同，但通过统一的接口定义，可保证整体架构的稳定性。其主要职责包括：

a. IP功能实现：基于HWL层接口实现不同IP（CAN、ADC、PWM等）驱动模块功能。

b. 状态管理：管理IP的各种状态，包括初始化、运行和停止等。

c. 错误处理:实现基本的错误检测和超时处理机制。

在IPL的实现中,可采用以下策略:

a. 模块化设计:将各IP设计为独立模块,配备清晰的接口定义,提高代码可维护性的同时,便于后续扩展和优化。

b. 错误机制:所有函数返回标准化错误码,便于错误处理并提高代码可读性和可维护性。

c. 统一命名规范:制定严格的IPL命名规范以保持接口定义不变。

3.3 HL实现策略

HL作为架构顶层,严格遵循AUTOSAR规范,实现标准要求的功能和接口,确保MCAL架构与AUTOSAR生态系统完全兼容,为上层应用提供规范化接口。HL屏蔽底层实现细节,使不同IP的实现对上层应用透明。其主要职责包括:

a. AUTOSAR接口实现:接口设计严格遵循AUTOSAR规范,保证系统兼容性。

b. 配置管理:处理AUTOSAR配置工具生成的配置数据。

c. 诊断支持:实现AUTOSAR要求的诊断功能。

d. 控制路由与多IP协调:根据上层请求将控制命令路由到相应IP;协调多个IP间工作,处理潜在冲突。

在HL的实现中,可采用以下策略:

a. 接口标准化:采用统一规范设计接口,实现数据传输、控制和状态查询等核心功能,以适应整合后的功能需求。

b. 严格遵循AUTOSAR标准:确保所有接口的命名、参数和返回值严格符合AUTOSAR标准,实现与生态系统的完全兼容。

c. 诊断功能集成:在各模块中集成诊断功能,实现全面的系统监控和故障诊断。

d. 灵活配置机制:处理AUTOSAR配置工具生成的配置数据,支持对底层IP进行配置调整,使系统能够适应不同场景的需求。

4 分层隔离的应用与验证

4.1 场景应用

分层架构主要分为开放层和封闭层,在封闭层中,所有请求需严格遵循预定的层次顺序进行传递,不允许跨层访问或绕过中间层,保证数据流的可控性和可追踪性。相反,开放层提供了更大的灵活性,特定情况下允许请求跳过中间层,直接访问下层组件,从而提高系统响应效率。这种分层设计明确了各层间的交互方式,使系统更加清晰可控,层次传递机制实现请求的正确处理和转换,为系统架构建立了明确的边界,同时便于实现统一的接口定义。

对于AUTOSAR应用,ECU抽象层发起的请求和芯片硬件的响应均需严格遵循层次顺序进行调用传递。如图6所示,请求信号从ECU抽象层发起后,经由HL、IPL逐级下传至HWL,硬件响应则沿原路径反向上传,最终返回ECU抽象层。

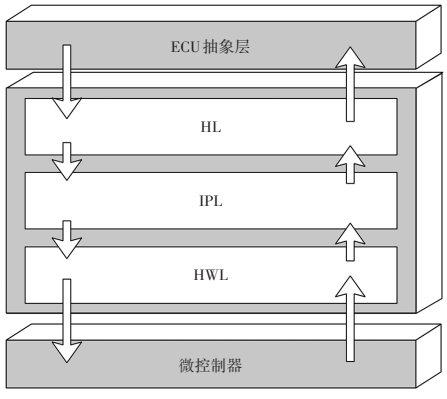


图6 AUTOSAR应用场景数据流传输

SDK应用在开发时具有更灵活的调用方式,可根据具体需求直接访问IPL甚至HWL,如图7所示。允许请求绕过中间层级,直接访问底层组件的架构模式被称为快速通道模式,不仅简化了开发流程,还能有效提升系统响应速度。因此,本文架构将HL设计为开放式结构,允许上层请求直接访问底层接口,保持系统架构的规范性的同时,提升系统整体性能表现。

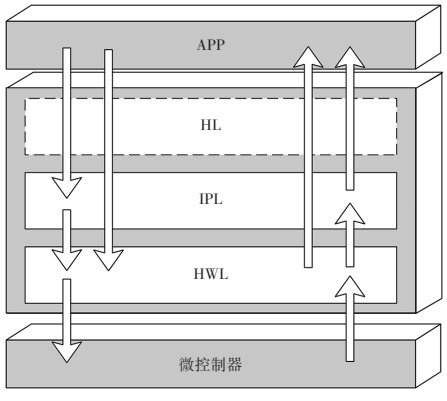


图7 SDK应用场景数据流传输

4.2 开发环境

为了全面评估架构设计的实际效果,本文选择小华半导体XC277芯片作为试验平台开展验证工作。XC277基于ARM Cortex-M7双核架构的车规级微控制器,主频为300 Hz,符合ISO26262 ASIL B功能安全标准。

基于改进架构的AUTOSAR MCAL软件开发遵循V

模型开发流程[10]，包括需求设计、架构设计、详细设计、单元测试、架构测试、需求测试，以及相关功能安全和质量审核等。为了保证代码的可靠性和兼容性，本文架构支持多种主流工具：编译工具可使用IAR编译器（版本号8.50.10）、GCC编译器（版本号10.3-2021.10.290）、GHS编译器（版本号2021.1.4）；配置工具支持EB tresos软件（版本号29.3.0）；调试工具可使用CMSIS-DAP、J-Link、I-JET和T32。

4.3 单元测试和集成测试

TESSY嵌入式软件对C/C++代码进行单元测试和集成测试，能够自动执行测试并生成评估测试报告。在V模型开发中，单元测试通过执行代码寻找函数内的错误，如算法错误、边界控制错误、数据溢出等；集成测试在单元测试的基础上，验证各单元间的接口是否正确[11]。验证模块包括：ADC、CAN、循环冗余校验（Cyclic Redundancy Check，CRC）、DIO、ETH、FEE、FLS、GPT、内部集成电路（Inter-Integrated Circuit，I2C）、ICU、LIN、MCU、OCU、外设管理（Peripheral Manager，PM）、PORT、PWM、SPI、系统（SYSTEM）、通用异步收发传输器（Universal Asynchronous Receiver/Transmitter，UART）、WDG。

使用代码覆盖率作为评估单元测试和集成测试的关键指标[2]。其中：C0为语句覆盖率（Statement Coverage），表示已执行语句的百分比；C1为分支覆盖率（Branch Coverage），计算测试中已执行计算机程序的控制流分支占比；修改条件/判定覆盖率（Modified Condition/Decision Coverage，MC/DC）表示测量控制流中独立影响判定结果的单一条件执行百分比；函数覆盖率（Function Coverage，FC）表示已执行函数的百分比；调用对覆盖率（Call Pair Coverage，CPC）表示已执行的函数调用相对于所有实现调用的百分比。经过测试各项指标均达到100%。

4.4 功能测试

功能测试重点考察各模块的实际应用表现，测试用例覆盖了常规功能、异常处理机制以及边界条件等维度[11]，确保系统能够稳定运行。通过使用信号发生器、逻辑分析仪、Vector CANoe、Kvaser等专业测试工具，对MCAL各模块进行全面的功能验证，结果如表1所示，所有模块通过率100%，均符合AUTOSAR规范的性能要求。

4.5 性能测试

为了满足系统硬件架构设计要求，对改进架构进行性能评估，评估内容涵盖任务执行时间、存储资源使用RAM堆栈、只读存储器（Read-Only Memory，ROM）程序空间）和各类总线（如CAN、SPI、I2C等）通信带宽等关键指标[11]，任务执行时间开销与计算开销结果分别如表2、表3所示，关键任务响应时间优化至微秒级，确保了系统的实时性能。系统资源利用高效，各模块的ROM和RAM占用率均控制在1%以内，为后续功能扩展预留了充足的空间。

表1 功能测试用例数量统计 个

模块	数量	模块	数量
ADC	1 398	LIN	861
CAN	1 053	MCU	1 125
CRC	453	OCU	666
DIO	738	PM	2 790
ETH	1 398	PORT	1 869
FEE	924	PWM	1 536
FLS	1 455	SPI	870
GPT	879	SYSTEM	390
I2C	696	UART	516
ICU	1 992	WDG	843

表2 不同编译器下任务执行时间开销 μs

模块	IAR		GCC		GHS	
	min	max	min	max	min	max
ADC	0.20	180.01	0.25	192.42	0.21	180.98
CAN	0.10	115.43	0.20	114.42	0.29	115.77
CRC	0.13	7.07	0.20	6.23	0.31	6.73
DIO	0.19	1.50	0.27	1.47	0.30	1.35
ETH	0.21	7 917.68	0.23	7 084.31	0.33	7 917.72
FEE	0.05	1.10	0.05	1.51	0.07	2.02
FLS	0.27	38 131.20	0.21	38 273.60	0.30	41 976.10
GPT	0.24	81 303.40	0.26	33 624.20	0.30	33 101.50
I2C	0.22	1 661.01	0.26	1 661.03	0.24	1 661.92
ICU	0.18	14.35	0.15	12.66	0.25	14.35
LIN	0.16	79 904.60	0.16	81 870.80	0.25	80 085.10
MCU	0.18	1 805.63	0.15	1 863.49	0.25	1 757.74
OCU	0.15	19.46	0.15	19.55	0.25	20.62
PM	0.18	20 006.70	0.20	18 339.60	0.28	20 006.60
PORT	0.20	290.22	0.24	244.52	0.21	266.45
PWM	0.19	30.24	0.18	28.07	0.19	28.64
SPI	0.16	212.81	0.20	205.24	0.18	210.66
SYSTEM	0.12	22.50	0.13	22.41	0.15	22.50
UART	0.21	80 708.30	0.20	71 269.50	0.32	82 913.50
WDG	0.40	3.03	0.31	3.25	0.27	3.53

表3　不同编译器下任务执行计算开销　　B

模块	IAR		GCC		GHS	
	min	max	min	max	min	max
ADC	5 992	7 780	6 726	8 762	5 938	8 030
CAN	11 742	12 888	13 000	14 208	12 150	13 300
CRC	3 930	5 258	5 240	6 922	5 766	7 228
DIO	718	1 156	850	1 606	702	1 392
ETH	17 592	19 150	19 972	21 480	18 586	20 034
FEE	4 822	9 502	6 176	9 198	4 596	8 838
FLS	15 500	20 252	16 726	22 706	16 672	22 890
GPT	6 902	8 658	6 840	7 524	5 994	7 868
I2C	6 880	7 922	5 218	5 354	7 638	8 830
ICU	8 214	11 130	5 238	5 782	8 260	11 312
LIN	6 790	8 096	6 162	7 958	7 358	8 190
MCU	14 344	14 840	15 244	15 808	16 244	16 848
OCU	4 050	5 450	4 226	5 544	3 862	4 834
PM	27 106	30 916	30 960	35 790	30 480	34 656
PORT	3 144	3 318	3 666	3 946	3 714	4 024
PWM	16 052	17 360	15 084	16 320	13 976	14 730
SPI	6 778	9 286	6 752	9 028	7 622	9 464
SYSTEM	7 186	8 282	7 804	8 978	7 880	9 032
UART	7 558	10 080	7 174	10 146	8 278	10 770
WDG	666	922	692	960	670	956

5　结束语

本文方案通过XC277芯片平台验证,在功能和性能方面均已达到预期目标。与现有的MCAL架构相比,具有显著优势:在架构设计方面,现有MCAL架构功能组件间耦合度高、边界模糊,易引发连锁反应,而本文架构通过HWL、IPL、HL 3层清晰划分,实现了功能解耦,模块职责明确,使系统结构更清晰可控;在模式开发方面,现有MCAL架构仅专注于单一开发模式,适应性相对受限,而本文架构创新性地支持AUTOSAR应用和SDK应用2种模式,兼顾了标准合规性与开发灵活性;在维护成本方面,现有MCAL架构每次芯片升级或外设变更,需投入约3~6个月的开发周期,而本文架构层间接口稳定,可通过替换对应部署单元快速适配,开发周期缩短至1~2个月,维护成本可降低60%。

因此,本文架构满足AUTOSAR标准和ISO 26262规范,为国产AUTOSAR MCAL的发展提供了实用的技术方案。

参　考　文　献

[1] 刘宏倩. 软件定义汽车时代下AUTOSAR助力中国汽车产业变革[J]. 交通节能与环保, 2024, 20(3): 74-77.
LIU H Q. In the Era of Software-Defined Vehicles, AUTOSAR is Helping to Transform China's Automotive Industry[J]. Transport Energy Conservation & Environmental Protection, 2024, 20(3): 74-77.

[2] International Organization for Standardization. Road Vehicles-Functional Safety Part 6: Product Development at the Software Level: ISO 26262-6: 2018[S]. International Organization for Standardization, 2018.

[3] 朱元, 陆科, 吴志红. 基于AUTOSAR规范的车用电机控制器软件开发[M]. 上海: 同济大学出版社, 2017.
ZHU Y, LU K, WU Z H. Software Development of Vehicle Motor Controller Based on AUTOSAR Specification[M]. Shanghai: Tongji University Press, 2017.

[4] 孙升, 宋珂, 章桐. AUTOSAR标准发展及应用现状[J]. 机电一体化, 2014(12): 33-38+44.
SUN S, SONG K, ZHANG T. Development and Application Status of AUTOSAR[J]. Mechatronics, 2014(12): 33-38+44.

[5] 米罗斯拉夫·斯塔隆. 汽车软件架构[M]. 北京: 机械工业出版社, 2020.
STARON M. Automotive Software Architectures[M]. Beijing: China Machine Press, 2020.

[6] AUTOSAR. Layered Software Architecture[EB/OL]. (2021-02-01) [2025-09-05]. https://www. autosar. org/fileadmin/standards/R18-10_R4.4.0_R1.5.0/CP/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf.

[7] 张成硕. 基于AUTOSAR标准的跨ECU平台微控制器抽象层(MCAL)的开发[D]. 杭州: 浙江大学, 2012.
ZHANG C S. AUTOSAR-Related MCAL, Software Development on Many ECU Platforms[D]. Hangzhou: Zhejiang University, 2012.

[8] 尼古拉斯·纳威特, 佛朗西斯·西蒙-莱昂. 汽车嵌入式系统手册[M]. 北京: 机械工业出版社, 2016.
NAVET N, SIMONOT-LION F. Automotive Embedded Systems Handbook[M]. Beijing: China Machine Press, 2016.

[9] 理查德·马克, 尼尔·福特. 软件架构:架构模式、特征及实践指南[M]. 北京: 机械工业出版社, 2021.
MARK R, FORD N. Fundamentals of Software Architecture: An Engineering Approach[M]. Beijing: China Machine Press, 2021.

[10] 高尚兵, 高丽. 软件测试基础教程[M]. 北京: 北京工业大学出版社, 2023.
GAO S B, GAO L. Fundamentals of Software Testing[M]. Beijing: Beijing University of Technology Press, 2023.

[11] 赵鑫, 李明勋. 汽车电子功能安全实战应用[M]. 上海: 同济大学出版社, 2021.
ZHAO X, LI M X. Practical Application of Functional Safety in Automotive Electronics[M]. Shanghai: Tongji University Press, 2021.

(责任编辑　瑞　秋)
修改稿收到日期为2025年9月5日。