恩智浦半导体
应用笔记

文档编号：AN13435
第 0 版，2021 年 10 月

从 SDK/MCAL 迁移到实时驱动程序

SDK/MCAL 到实时驱动程序的迁移指南

作者：恩智浦半导体

1. 介绍

实时驱动程序（RTD）是生产级的，是复杂硬件功能的软件抽象层，可用于 AUTOSAR 和非 AUTOSAR 应用。

RTD 提供符合 ISO26262 的标准化 API（AUTOSAR 和 AUTOSAR 扩展接口），可用于各种产品和专用硬件特定接口。

RTD 提供多种软件功能作为 AUTOSAR 标准功能的扩展功能（用于开放专门的硬件功能），覆盖所有硬件功能和硬件外设。RTD 集成了带有默认配置的驱动示例，针对 CT 和 EB Tresos 配置工具都提供了示例，SDK 和 MCAL 中的方法都有涉及。RTD 由 AUTOSAR MCAL 驱动和一组复杂器件驱动组成。

RTD 在外设上的映射，请参见特定平台的附录。

RTD 整合了 AUTOSAR 和非 AUTOSAR 环境中的用例，因此从客户角度来看，SDK 和 MCAL 中的现有功能得以保留。标准软件包中的各个扩展功能可启用或禁用。

目录

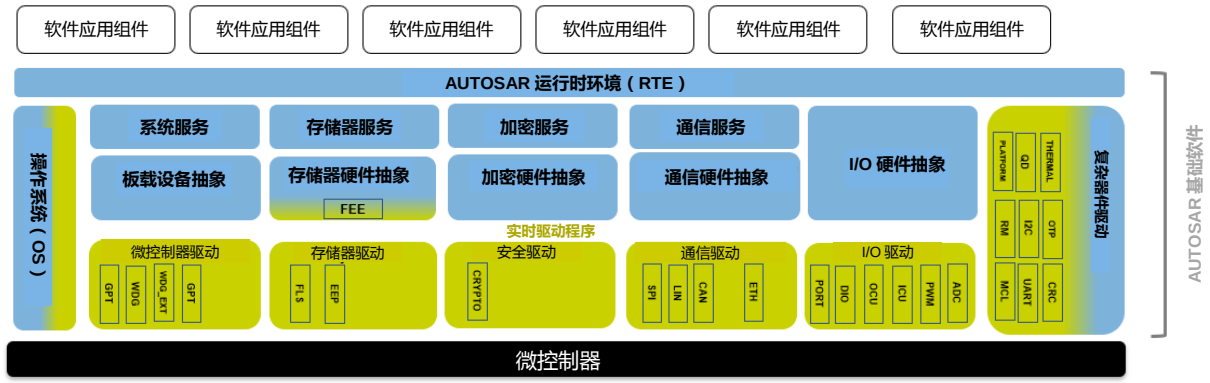


图 1. AUTOSAR 架构中的实时驱动程序综述

从 MCAL 客户的角度来看，RTD 扩展了 AUTOSAR 标准定义的标准功能，涵盖了所有硬件功能。对于 AUTOSAR 应用，提供标准化接口，建议使用该接口以保证应用之间的可移植性。

从 SDK 客户的角度来看，RTD 扩展了标准化功能，并添加了多核支持、用户模式运行支持、代码和数据到特定存储器分段的存储器映射。对于非 AUTOSAR 应用，提供两种接口（标准化接口和 IP 接口）。具体取决于应用类型和项目限制。

RTD 可以使用任何 AUTOSAR 配置程序和恩智浦 S32Design Studio 进行配置。在配置方面，恩智浦 S32 Design Studio 支持独立于驱动程序来配置驱动层和外设层。在同一个硬件单元中标准化接口和外设层接口的使用是不可兼容的。

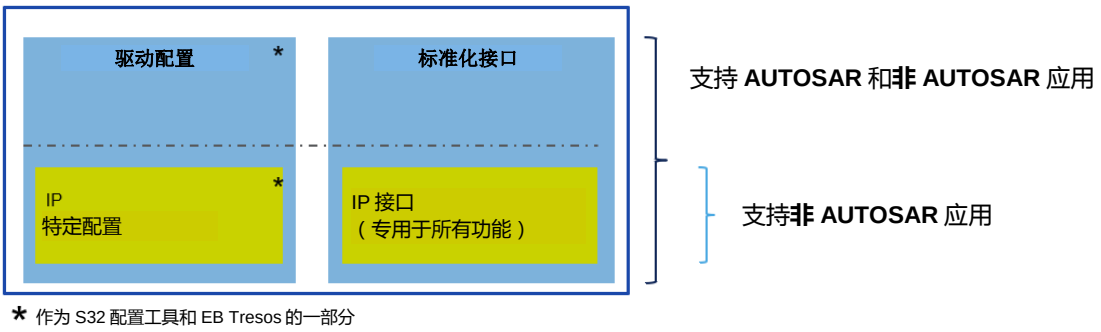


图 2. 驱动接口和配置综述

2. MCAL 到 RTD 的迁移指南

将应用从 MCAL 迁移到 RTD 时，应考虑以下要素。

2.1. AUTOSAR 版本和配置影响

标准 AUTOSAR MCAL 模块是 RTD 的一部分，按照 AUTOSAR 4.4 要求来实施。因此，这些驱动程序的接口和配置是符合此标准的。

支持相同的配置工具。为模块提供的默认配置文件可用于配置 AUTOSAR 项目中的驱动程序。已支持的工具不需要更改。

旧的 MCAL 配置可以导入 RTD 项目。工具导入器不会自动更新或添加任何参数，需要在项目配置阶段对参数进行相应更新。

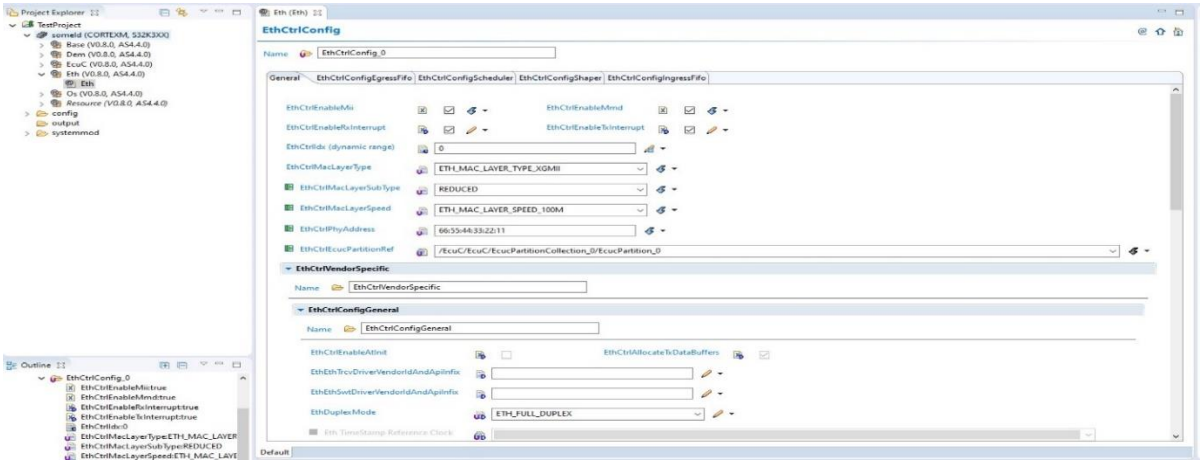


图 3. EB Tresos 中的驱动配置

此外，驱动程序还集成了对 S32 Design Studio 配置工具的支持。可配置整个驱动程序（即其在 Tresos 中的配置）和独立于外设接口的配置。例如，配置 AUTOSAR CAN 驱动程序（AUTOSAR 接口），或仅配置非 AUTOSAR FlexCan 模块（外设接口）。

如需了解每个特定平台迁移的详细信息，请参阅特定平台的附录。

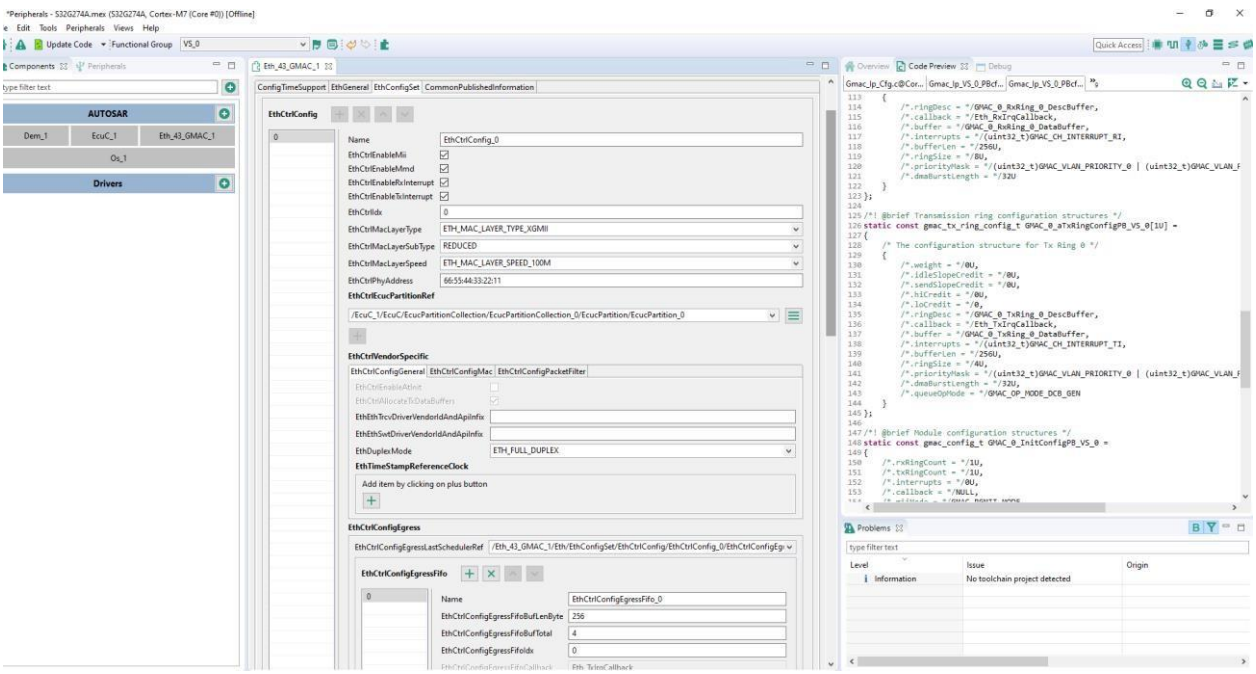


图 4. S32DS 中的驱动配置

2.2. 功能影响

RTD 允许访问一组扩展的硬件功能。从 AUTOSAR 的角度来看，RTD 附带了额外的扩展功能和新的 CDD，可在已标准化的 AUTOSAR 功能基础上解决大多数硬件功能问题。

2.3. 标准功能影响

如果项目采用的是 AUTOSAR 中介绍的所有标准功能，则迁移是无缝的。RTD 也支持 AUTOSAR 扩展功能，参见旧版 MCAL。

此外，为了支持旧版 MCAL 用例，标准驱动程序采用新的 API 来扩展硬件功能覆盖范围。而图形界面经过更新，可以配置这些功能。

2.4. CDD 功能影响

添加新的 CDD 可启用以前 MCAL 版本未涵盖的硬件外设（即 UART、Quadrature、RM、Platform）。

通过平台 CDD 可为裸机应用提供一项新功能。此功能可配置和处理应用的中断。此 CDD 的使用是可选的，还有一个选项是允许对应用施用任何其他方式来实现中断管理。

以前仅在 MCL 上可用的功能现在可迁移到 RM（资源管理器 CDD）和 MCL，更好地覆盖硬件平台功能并拆分功能。MCL 支持启用 DMA 和缓存。RM 将支持启用 XRDC、SEMA42、MPU 和与 crossbar 相关的（如果适用）配置。

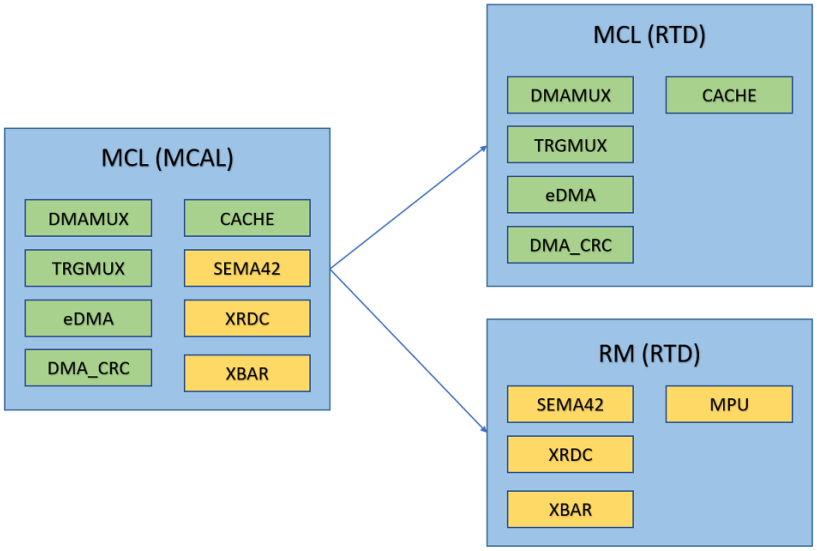


图 5. MCAL 到 RTD 的迁移

2.5. 文件结构影响

2.5.1. 插件结构

对于 MCAL 用户来说，RTD 插件结构的影响是有限的。它有一个与旧版 MCAL 类似的文件和文件夹结构。其他文件和文件夹（例如 Base 插件中的“headers”文件夹）需要包含在内。每个文件依赖项的详细信息在驱动 IM 中指定。

配置数据文件现在按照更精细的方法进行拆分，确保可以使用独立外设驱动。在功能方面，AUTOSAR 应用中需要的所有数据都将通过 HLD 文件导出，因此应用流程中不会发生任何变化。

所有模块在使用前都需要配置并生成配置文件。可从附带的默认配置文件入手。

表 1. MCAL 和 RTD 的差异

MCAL	RTD	备注
<Mdl>_Cfg.h	<Mdl>_Cfg.h <Mdl>_Ipw_Cfg.h <Ip>_Cfg.h	包含驱动程序中使用的预编译参数，通常是定义和常量、外部声明和数据类型。
<Mdl>_Cfg.c	<Mdl>_Cfg.c <Mdl>_Ipw_Cfg.c <Ip>_Cfg.c	静态配置结构仅包含不能识别变体的变量，它们仅配置和生成一次。该文件本身并不包含<*Mdl*>*_Init* 函数配置驱动程序所需的整个结构。根据 EcuC 中配置的变体数量，一个模块可以有多个配置结构，即使对于预编译变体也是如此。
<Mdl>_PBcfg_<Variant>.c	<Mdl>_PBcfg_<Variant>.c <Mdl>_Ipw_PBcfg_<Variant>.c <Ip>_PBcfg_<Variant>.c	每个变体都有一个文件。文件名包含 EcuC 中定义的变体名称。此文件包含可感知变体的驱动程序所使用的配置结构。每个文件都包含相应变体的配置参数。如有需要，所有不能识别变体的、且在<Mdl>_Cfg.c 文件中仅生成一次的参数和/或结构，会在<Mdl>_PBcfg_<Variant>.c 文件中的结构中引用。所有变体中都使用配置结构。
<Mdl>_PBcfg_<Variant>.h	<Mdl>_PBcfg_<Variant>.h <Mdl>_Ipw_PBcfg_<Variant>.h <Ip>_PBcfg_<Variant>.h	创建它是为了导出每个配置结构的外部声明，以便在应用中调用<*Mdl*>*_Init* 时使用。每个变体都有一个文件。文件名包含 EcuC 中定义的变体名称。

2.6. 独占区

采用独占区方式可确保线程安全，独占区名称不会更改，将与旧版 MCAL 保持相同。

2.7. 超时处理

OsIf 模块被添加到 RTD 中，支持应用在 OS 集成方面采用更灵活的方法，并在配置超时时为用户提供更多选项。例如，使用 OS 或硬件定时器进行精确定时或循环计数，可避免使用任何额外的资源。

OsIf 模块需要在基本组件中配置：

- 使用的 OS 类型
- 要启用的计数器/定时器类型
- 根据具体情况，引用 OS 计数器或 MCU 时钟

可以在每个驱动程序中选择超时类型（以微秒为单位的精确定时或循环计数）。

从迁移的角度来看，超时处理是向后兼容的，因为循环计数是驱动程序中的有效配置选项。

2.8. 编译器抽象

删除与 AUTOSAR 特定宏（即 FUNC、VAR）相关的存储器映射，可简化编译器抽象。从迁移的角度来看，预计不会产生任何影响，因为存储器映射是通过 MemMap 功能来支持的，而受支持的编译器不需要这样的抽象宏，并且有助于提高代码的清晰度并与代码解析器工具相符。

2.9. 迁移步骤

下面的小节介绍了将 MCAL 项目迁移到 RTD 项目需要遵循的步骤。

2.9.1. 驱动配置

由于硬件外设的变化，需要通过以下步骤迁移配置。

1. 使用所选配置工具的导入功能，可以从旧项目导入所有通用配置。这将导入所有未更改的配置字段（所有 AUTOSAR 特定参数、大多数高级配置参数等）。
2. 对于工具无法直接导入的所有平台特定参数，需要在配置工具中根据特定平台进行更新。所提供的推荐配置从默认配置入手来减轻迁移工作。

3. 需要重新配置 RTD 更新/全新的参数。所提供的推荐配置从默认配置入手来减轻迁移工作。

2.9.2. 驱动构建（Build）

1. 文件结构更新，在需要时包括新的来源。
2. 映射重新构建的驱动程序（即 MCL、RM、MCU。更多详细信息请参见其相应的文档部分）的配置、功能和符号名称的更改。

2.9.3. 功能更新

1. RTD 保留了旧版 MCAL 提供的功能，并在此基础上构建对其他硬件外设的支持。

3. SDK 到 RTD 的迁移指南

RTD 提供 SDK 中已提供的功能，并使用 AUTOSAR 实施方法扩展了 PAL，支持 AUTOSAR 和非 AUTOSAR 应用的用例。RTD 对所有硬件功能进行抽象，与 SDK 中的支持一样。

RTD 架构能够提供外设接口来实现外设层的解耦，以便能够独立使用。

将应用从 SDK 迁移到 RTD 意味着要根据 RTD 架构进行一些更改。

3.1. 配置工具的影响

从 SDK 的角度来看，S32 配置工具可用于配置驱动程序。在 S32 配置工具中，可以配置驱动程序的 HL 接口和 IP 接口，以保持 SDK 中已提供的功能。

具体细节见特定平台的附录。

3.2. 驱动配置更改

RTD 的配置目的是满足 ASR 和非 ASR 客户的需求，因此可以使用 S32CT 和 EB Tresos 生成配置代码。驱动配置文件遵循相同的分层架构，在 HL 层和 IP 层之间拆分，这两层之间为内部 IPW 胶合层。

注意

SDK 和 RTD 配置方案的主要区别在于对动态（可配置）的预编译配置参数的使用。它使应用禁用不打算使用的部分驱动程序代码（RTD 驱动源文件，包括配置标头文件），从而在应用端获得更大的灵活性和更少的代码。

3.2.1. 配置类和变体支持

RTD 配置遵循适用于所有层的 ASR 配置概念。为了支持多个配置以供在不同绑定时间进行选择，添加了变体和配置类支持。因此，配置代码可在多个文件中生成，与项目中定义的变体相对应。

变体支持意味着可生成多个配置结构，在运行时进行选择。变体支持主要用例是可在运行时重新配置 ECU（不同的模式支持——启动、运行时及关闭模式下的驱动行为，相同的代码与不同的行为取决于外部输入——左门与右门汽车集成，等等）。

配置类支持则意味着可生成多个配置结构，这些结构可在预编译时、链接时、构建后进行选择。每个驱动程序都支持预定义的一个、多个或所有配置类。图 6 描绘了配置文件的组织方式。

所有配置结构都可作为常量生成，避免虚假损坏，并分配在驱动程序的存储器空间中，支持独立的存储器映射（例如：将配置数据映射到慢速/只读存储器中，将驱动代码映射到快速存储器中）。

注意

无需将配置指针传递给初始化函数即可支持初始化驱动已被 PRE_COMPILE 模式配置支持所取代（如 AUTOSAR 方法中一样）。

默认配置不会作为内部驱动全局变量存储，而是可以生成（单一/唯一的）预编译配置，由使用内部机制的驱动程序直接引用。这样做的好处是支持相同的用例，并可借助图形用户界面配置程序定制此配置、支持存储器重新映射、减少内存消耗，为所有配置模式提供一致的方法。

所有模块在使用前都需要配置和生成配置文件。附带的默认配置文件将作为起始点。这些需要先由配置工具进行处理，然后才能执行模块初始化。

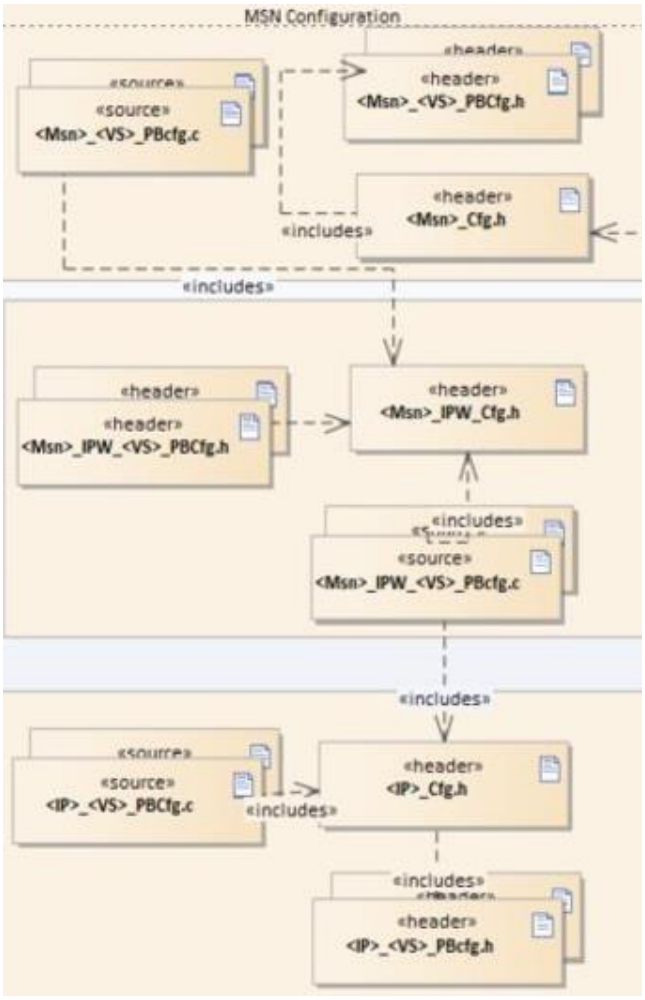


图 6. 驱动配置文件结构

3.3. 功能影响

RTD 对所有硬件功能进行抽象，并提供跨平台的标准化接口。由于架构和命名规范的变化，SDK API 和数据类型已更改，可支持当前的方法，这意味着即使从功能的角度来看，RTD 也可提供与 SDK 相同的功能，迁移对客户来说将不透明。

RTD 的命名规范意味着数据类型和 API 的更改是一致的，如下所示。

3.3.1. 数据类型

RTD 中的数据类型使用以下命名规范：

<Prefix>_<TypeName>Type

其中：

- **<TypeName>** 应遵循所谓的 CamelCase 命名规范（每个单词的第一个字母为大写，后面的字母为小写）。
- **<Prefix>**为
    - HLD: <MSN> ModuleShortName
    - IPW: <MSN>_Ipw (*1)
    - 共享 IP：<Ip>_<MSN>_Ip
    - 非共享 IP：<Ip>_Ip
    - 示例：typedef uint16 Spi_NumberOfDataType；

注意

"IPW"符号是专用的，不用于应用。

3.3.2. API（函数）名称

RTD 中的 API 遵循以下命名规范：

- **HL 接口：**<MSN>_<Function>()（例如：Wdg_Init()）
- **IP 接口：**
    - **<IP>_<MSN>_IP_<Function>()**，用于所有共享 IP（例如 eTimer_Icu_Ip_StartSignalMeasurement()）
    - **<Ip>_Ip_<Function>**，用于不共享的所有 IP（例如 Swt_Ip_Init()）

注意

"IPW"符号是专用的，不用于应用。

下表汇总了 SDK API 名称和 RTD 名称之间的差异。

表 2. SDK 和 RTD 名称的差异

名称	SDK	RTD
数据类型	*_t* suffix & *snake_case* style	*Type* suffix & *CamelCase* style
IP 层 API	*<MDL>_DRV_FunctionName*	*<MDL>_IP_FunctionName*
高级 API	*<MDL>_PAL_FunctionName*	*<MDL>FunctionName*

注意

这并不意味着 API 中的这些形式更改是从 SDK 移植到 RTD 所需的唯一更改。RTD 中的 API 也包含语义更改，这是与更高级集成的需要。从功能的角度来看，RTD API 在逻辑上映射到旧版 SDK API（没有回归/降级功能）；但是，一些函数名称/参数的含义可能不同。仔细阅读用户手册后会发现，使用适当的 API 来实现所需的功能是由应用来负责的。

3.4. 存储器映射

RTD 引入了将代码和数据映射到特定存储器分段的机制，以避免 RAM 浪费，支持使用特定 RAM 属性、使用特定 ROM 属性、引导加载程序和应用使用相同的模块源代码，还可以支持封装和隔离。

RTD 数据包中提供了默认的存储器分段，因此从 SDK 迁移到 RTD 意味着只将提供给 Base 插件的<Driver>_MemMap.h 文件添加到项目中，并更新链接文件。

<DRIVER>_MemMap.h 文件存根已提供，预计会在 AUTOSAR 上下文中更新，并且可以按原样在非 AUTOSAR 上下文中使用。

3.5. 开放接口

RTD 的目的是同时满足 AUTOSAR 和非 AUTOSAR（旧版 SDK）用例。RTD 提供两组接口：

- 跨平台通用的标准化接口
- 具有相同 IP 功能的跨平台通用 IP 接口

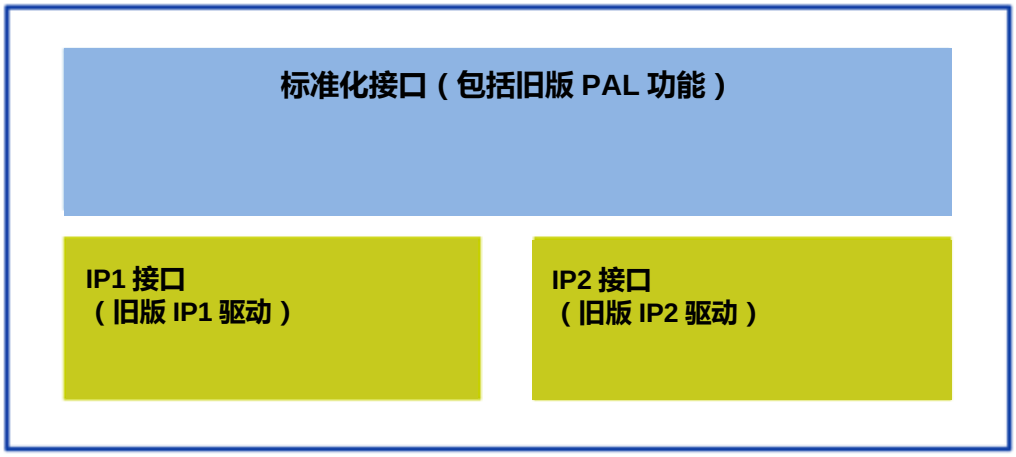


图 7. 开放接口

将应用从 SDK 迁移到 RTD 时，以下限制仍然适用：

- 禁止在 HL 和 IP 中使用相同的硬件实例（例如，如果在 HL 上下文中使用 SPI1，则不能通过 IPL 也使用 SPI1）。
- IP 层不提供 Tresos 配置。它只能在 Design Studio（CT）上配置。
- IP 层不适用于 AUTOSAR 应用，因为它不满足 ASR 合规性约束（DEM、DET、多核等）；IP 层不是一个独立的 AUTOSAR CDD。

3.6. 错误管理

RTD 的错误检测和报告机制是为目标应用类型量身定制的。错误管理概念包含开发错误和运行时错误的报告，使用如下所述的不同报告机制。

从错误管理的角度来看，要迁移使用 PAL 功能的应用，意味着需要对架构进行修改。对于高层，错误管理遵循默认错误跟踪器（Default Error Tracer）和诊断事件管理器（Diagnostics Event Manager）的 AUTOSAR 规范。RTD 为实现这些模块提供了参考代码，客户应用可以使用或覆盖这些代码。

3.6.1. HL API

需要同步反应时，HL API 返回 Std_ReturnType (E_OK/E_NOT_OK)。

当定义了异步反应时，HL API 将返回一个额外的特定错误，然后可以调用 DEM/DET 中的专用 API 来提取此错误。

默认错误跟踪器（DET）提供了处理开发错误和运行时错误的机制。

诊断事件管理器（DEM）提供了处理关键运行时错误的机制，以防这些错误对应用完整性产生重大影响。

RTD 提供这些 ASR 模块的“stub”实施方案，客户应用可以使用或覆盖这些参考代码。

3.6.2. IP API

IP 层报告的错误仍分为两类：

- **开发错误：**通常是参数检查、函数调用时的参数合法性等。这些错误是使用 DevAssert 函数检查的。如果检测到错误，它会在默认实施中暂停程序执行。此应用也可以覆盖 DevAssert 函数的默认行为。这种机制与旧版 SDK 中的 DEV_ASSERT 功能几乎相同，唯一的改进是现在可以分别为每个驱动程序启用/禁用这些语句，而不像原 SDK 方法那样使用的是全局配置（参见图 8）。
- **运行时错误：**在 SDK 中，驱动程序报告的所有运行时错误都归在名为 Status_t 的通用枚举组中，而与 SDK 不同的是，RTD 根据每个驱动程序定义一组运行时错误。这些错误的命名规范为<IP_NAME>_IP_StatusType。

每个驱动程序定义的一组错误可以由受控 IP 报告，这些错误由 IP 层上部署的非 ASR 应用来使用以提取驱动程序的状态，或进一步馈送到顶层的高级状态机。

3.7. 文件结构

为了同时符合 AUTOSAR 和非 AUTOSAR 配置工具，RTD 在文件布局上使用模块化方法。虽然在 S32SDK 中，驱动程序共享一个公共文件夹，但与 PAL 不同，RTD 模块包含在单独的文件夹中。

因此，RTD 中的模块是 IP 层和 HL 驱动程序的容器。基于此，模块将有一个 IP 文件夹，其中包含目标 IP 的实施以及 HL 的两个文件夹（Include 和 src）。

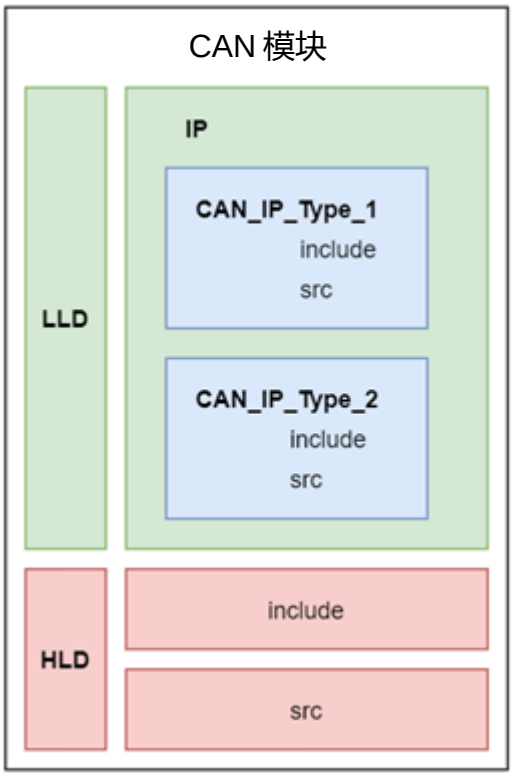


图 8. RTD 文件结构 include

3.8. 中断管理

与 S32 SDK 不同，RTD 不在系统级管理中断请求。有一种外部假设，即在中断控制器中启用适当的中断，并且 IVT 中存在正确的处理程序，可使驱动程序正常运行。因此，应用负责配置中断控制器并定义适当的 ISR，参见每个驱动文档的描述。

从迁移的角度来看，RTD 定义了不同平台专用的驱动程序，称为 PLATFORM_CDD，它为设置中断提供 API 和配置支持。Platform_CDD 的配置包含中断设置（启用、优先级、处理程序等）所需的所有信息。调用这个新驱动程序的初始化函数可设置适当的配置，而且也是其他需要中断例程的驱动程序正确运行的先决条件。

3.9. 超时处理

对于超时处理，RTD 不支持互斥量和信号量。通过提供异步（中断驱动）和同步（轮询）服务，可以简化超时处理。如需了解更多详细信息，请查看超时处理。

3.10. 安全

RTD 包含安全分析（FMEA）和涵盖两个开放接口（HL+IP）的安全措施，包括应用的安全措施和外部假设。

4. OS 抽象 – OSIF

4.1. 从 MCAL 迁移到 RTD-OSIF

OsIf 模块可添加到 RTD，支持应用在 OS 集成方面采用更灵活的方法，并在配置超时时为用户提供更多选项，例如，使用 OS 或硬件定时器进行精确定时或循环计数，可避免使用任何额外的资源。

OsIf 模块需要在基本组件中配置：

- 使用的 OS 类型
- 要启用的计数器/定时器类型
- 根据具体情况，引用 OS 计数器或 Mcu 时钟

然后，在每个驱动程序中，可以选择超时类型（以微秒为单位的精确定时或循环计数）。

4.2. 从 SDK 迁移到 RTD-OSIF 再迁移到 OSIF

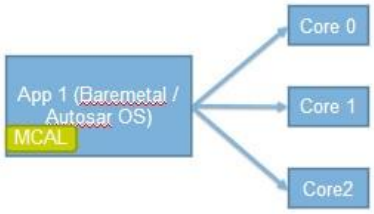
RTD 中的 OsIf 模块与 SDK 中的对应模块有很大不同。互斥量和信号量不再受支持，定时服务主要针对超时，而不是测量时间或延迟。

选择 OS 类型的项目级符号（编译器-D 选项），- DUSING_OS_BAREMETAL 或- DUSING_OS_FREERTOS 保持不变。而且，现在还支持-DUSING_OS_AUTOSAROS。

5. 多核支持

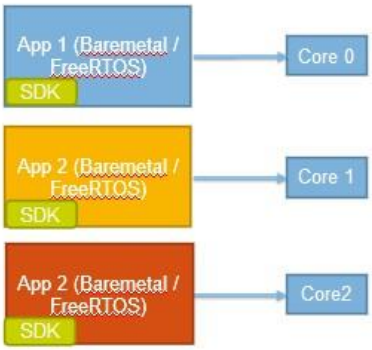
RTD 提供多核支持，支持旧版 MCAL AUTOSAR 多核概念和每核 SDK 实例方法。有关更多详细信息，请参见下图。

单映像、多核（旧版 MCAL）



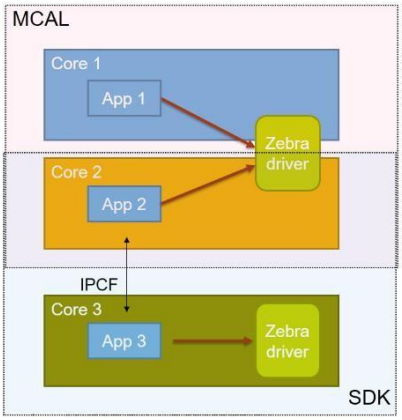
- 单一应用映像和数据空间
- 每个内核运行特定的应用代码，这些代码在运行时由 **GetCoreId ()**选择
- 共享驱动代码（内核），支持多核，例如 init()根据内核 id 初始化不同的硬件模块

每个内核单独的应用（旧版 SDK）



- 每个内核都有一个单独的应用映像（elg/二进制）
- 没有共享的驱动代码（没有内核）—如同有单独的“项目”
- 对系统范围的初始化（例如时钟）建立了质询口令

RTD 方法



- **RTD 建议 = MCAL 旧方法**，同时满足 SDK 和 MCAL 用例：
  - 两个内核运行相同的二进制代码（MCAL）
  - 两个内核运行不同的二进制文件（SDK）
  - 并可选择在应用级别通过 IPCF 进行同步
- IP 实施方案与 SDK 中的相同，不需要感知多核，直接从 HLD 接收正确的配置，所有全局变量都分配到硬件索引数组中。

图 9. 多核支持

6. 版本包释放

RTD 版本包包含源代码、Tresos 配置、S32CT 配置、示例和文档。

从交付的角度来看，有两种交付方法：

- 通过 Design studio 更新站点
- 在 Flexera 中发布安装程序

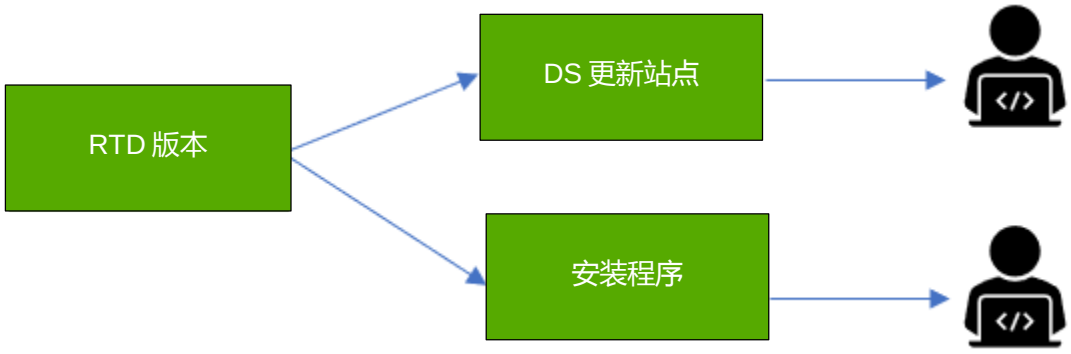


图 10. RTD 版本打包方法

从旧版 SDK 用户的角度来看，将产品部署为 S32 Design Studio 更新站点，与配置/构建/调试工具的集成将达到 IDE 集成所固有的级别。通过 S32 配置工具可以使用该产品，可配置 HL 和 IP 层；它还包含作为现成的 DS 项目交付的示例应用，突显了驱动程序的使用情况，并可通过内置的工具链支持随时下载到目标位置，为非 ASR 客户提供出色的“开箱即用”体验。

此外，RTD 与捆绑到 S32 DS 中的其他恩智浦软件产品相集成，包括可移植到驱动程序之上的各种非 ASR 中间件和堆栈。

从 MCAL 的角度来看，将产品作为 Flexera 中发布的安装程序来部署，将包含与 RTD 相同级别的集成。

附录 A. S32KXX 产品系列

第 1 章. 概述

下面总结了特定平台迁移所涉及的各个方面，包括平台特有的驱动/IP 映射、配置工具。

表 3. 外设组件和 S32K3XX 驱动的映射

RTD(实时驱动程序)	S32K3XX IP	备注
DEM	-	诊断事件管理器 参考代码由恩智浦提供，可用于非 AUTOSAR 应用。对于 AUTOSAR 应用，将替换为 AUTOSAR 标准实施方案。
DET	-	默认错误跟踪器 参考代码由恩智浦提供，可用于非 AUTOSAR 应用。对于 AUTOSAR 应用，将替换为 AUTOSAR 标准实施方案。
ECUC	-	Ecu 配置——添加对多核的支持 参考代码由恩智浦提供，可用于非 AUTOSAR 应用。对于 AUTOSAR 应用，将替换为 AUTOSAR 标准实施方案。
ECUM	-	Ecu 管理器 参考代码由恩智浦提供，可用于非 AUTOSAR 应用。对于 AUTOSAR 应用，将替换为 AUTOSAR 标准实施方案。
RTE	-	实时环境——实现独占区 参考代码由恩智浦提供，可用于非 AUTOSAR 应用。对于 AUTOSAR 应用，将替换为 AUTOSAR 标准实施方案。
OsIf	-	操作系统抽象层 集成支持 FreeRTOS 和 AUTOSAR OS

Resource	-	资源驱动——所有衍生功能的集合
BASE	-	基本驱动——标头文件的集合
	REG_PROT	由恩智浦提供一部分驱动的文件，可用于非 AUTOSAR 应用 由恩智浦提供一部分驱动的文件，对于 AUTOSAR 应用，可替换为 AUTOSAR 标准实施方案
MCU	MC_CGM	微控制器单元驱动 提供基本的微控制器初始化、模式管理和时钟管理等服务
	FIRC	
	SIRC	
	PLLDIG	
	FXOSC	
	MC_RGM	
	MC_ME	
	SXOSC	
	CMU	
	MC_PCU	
	PFLASH	
	PRAMC	
	PMC	
PLATFORM	MSCM	平台——复杂器件驱动 集成了平台专用的功能和中断管理
	NVIC	
	INTM	
	MCM	
RM	MPU	资源管理器——复杂器件驱动
	XRDC	
	SEMA42	
	PFLASH	

	VIRT_Wrapper	
	XBIC - MCR reg	
	XBAR	
MCL	eDMA	微控制器库——复杂器件驱动提供 DMA 和缓存功能
	LCU	
	DMAMUX	
	TRGMUX	
	Cache_M7	
PORT	SIUL2	端口驱动 提供初始化端口驱动的整体结构的服务
DIO		数字输入输出驱动 提供访问微控制器硬件引脚的服务
EEP	FLEXRAM	EEPROM 驱动 提供对 EEPRM 的读取、写入和擦除服务
	FLEXNVM	
FLS	QuadSPI	Flash 驱动
	C40	
	PFLASH	
ICU	eMIOS	输入捕获单元驱动
	SIUL2	
	LPCMP	
	WKPU	
OCU	eMIOS	输出捕获单元驱动
GPT	eMIOS	通用定时器驱动
	PIT(-RTI)	

	RTC	
	STM	
PWM	eMIOS	脉宽调制驱动
	FlexIO_PWM	
QD	eMIOS	正交复杂器件驱动
I2C	LPI2C	I2C 复杂器件驱动
	FlexIO_I2C	
ETH	ENET	以太网驱动
UART	LPUART	UART 复杂器件驱动
	FlexIO_UART	
LIN	LPUART	Lin 驱动
SPI	LPSPI	串并行接口驱动
	FlexIO_SPI	
CAN	FlexCan	Can 驱动
ADC	ADC	模拟数字比较器驱动
	BCTU	
Crypto	HSE_M	加密驱动
	MU	
WDG	SWT	看门狗驱动
SENT	FlexIO_SENT	SENT 驱动
CRC	CRCU	CRC 复杂器件驱动
SAI	SAI	SAI 复杂器件驱动

第 2 章. AUTOSAR 配置和版本影响

AUTOSAR MCAL 标准模块是实时驱动程序（RTD）版本一部分，将按照 AUTOSAR 4.4 要求实施，因此这些驱动程序的接口和配置将符合标准。

S32K1 和 S32K2 MCAL 项目是根据 AUTOSAR 4.3 开发的，S32K3 是根据 AUTOSAR 4.4 开发的。因此，其中的更新会对 AUTOSAR 特定参数产生影响。考虑到 AUTOSAR 4.4 是递增更新，与 AUTOSAR 4.3 相比没有颠覆性变化，因此预期的影响不会很大。受影响的模块包括：加密模块和执行配置更新以适应 AUTOSAR 多核概念的所有模块。

AUTOSAR 兼容的配置工具可以利用所有参数导入功能，保留从 MCAL 迁移到 RTD 模式的对应关系。大多数 AUTOSAR MCAL 标准参数和大多数供应商特定的扩展参数预计与新的 RTD 项目兼容，可导入到新 RTD 项目中。

对于所有新的 AUTOSAR RTD CDD 和少量的供应商特定参数，需要从头开始创建配置。可从默认配置文件入手，以减少工作量。

采用相同的配置工具和配置流程（包括默认配置）。

第 3 章. SDK 配置和工具影响

对于 SDK，S32 配置工具是 S32 Design Studio 的一部分，用来配置驱动程序。采用 S32 配置工具，可以配置驱动程序的 HL 接口和 IP 接口，保持 SDK 中已提供的功能。

S32K3 将支持相同的配置工具（S32CT）。

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN13435
Rev. 0
10/2021