Application Note

Peter Li, Joe Shen, and Karan Saxena

摘要

本应用报告重点介绍了德州仪器 (TI) 系统控制接口 (TISCI) 服务器在 Jacinto™ 7 系列器件的 Vector AUTOSAR 中如何集成。文中内容适用于在 MCU R5F 上使用 SDK7.1 或更高版本来运行 AUTOSAR 的系统。

内容

1 引言 2
1.1 SYSFW 2
1.2 TISCI 2
1.3 TISCI 客户端或 SciClient 2
1.4 TISCI 服务器或 SciServer 2
1.5 本文档中使用的首字母缩写词 2
2 目标受众 3
3 Jacinto7 SDK V7.1+ 中 MCU1_0 的作用 3
4 TI-RTOS 上的 TISCI 客户端和服务器 3
4.1 J7 SDK 下载 3
4.2 SciClient 驱动程序位置 3
4.3 TISCI 服务器初始化示例 4
4.4 集成指南 4
5 DaVinci 中的配置 5
5.1 DaVinci Developer 6
5.2 DaVinci Configurator Pro 7
5.3 资源 8
5.4 活动 8
5.5 SciServer 用户任务 9
5.6 同步 Sciserver 用户任务 10
5.7 Sciserver 中断 11
6 AUTOSAR TISCI 客户端 13
6.1 AUTOSAR 中的 TISCI 客户端注册 13
7 AUTOSAR TISCI 中断处理 14
7.1 MCU 域导航系统高优先级中断 14
7.2 主域导航系统高优先级中断 15
7.3 MCU 域导航系统正常优先级中断 15
7.4 主域导航系统正常优先级中断 16
8 AUTOSAR TISCI 用户任务处理 16
8.1 高优先级用户任务初始化 16
8.2 高优先级用户任务可运行文件 17
8.3 正常优先级用户任务初始化 17
8.4 正常优先级用户任务可运行文件 17
9 AUTOSAR 中的 TISCI 服务器验证 18
9.1 Boot App 18
9.2 引导任务配置 18
9.3 AUTOSAR 中的 Boot App 19
10 AUTOSAR 中使用的 PDK 库 20
11 AUTOSAR 所需的 R5F 配置 20
11.1 Cortex-R5F 的存储器布局 20
11.2 R5F 缓存配置 21

商标

1 引言

1.1 SYSFW

Jacinto 7 器件推出了集中式电源、资源和安全管理概念，以便缓解传统分布式系统控制方案中的各种挑战。

SYSFW 是一个用于描述 TI 基础安全 (TIFS) 和资源管理 (RM)/电源管理 (PM) 服务的统称。有关这方面的更多信息，请参阅公开提供的文档。

1.2 TISCI

德州仪器 (TI) 的系统控制接口 (TISCI) 定义了各种处理实体与 TI SoC 上系统控制实体之间的通信协议，包含一系列必要的报文格式和操作序列，用于进行通信并处理来自 SoC 上系统控制实体的系统服务。

TISCI 协议用于与 SYSFW 进行通信。有关这方面的更多信息，请参阅 TISCI 用户指南。

1.3 TISCI 客户端或 SciClient

SciClient 接口支持 TISCI 协议，面向基于 RTOS 和非 OS 的应用。它向更高级别的软件提供内核报文详情、有效模块/时钟 ID，并基于 TISCI 协议提取与 SYSFW 的通信。

有关这方面的更多详细信息，请参阅该 SDK 中的 SciClient 文档。

1.4 TISCI 服务器或 SciServer

Sciserver 是一个新模块，支持 MCU R5F 处理来自 SoC 上所有非安全实体的设备管理请求。DMSC 仅处理来自安全实体的请求。

1.5 本文档中使用的首字母缩写词

首字母缩写词	说明
J7	Jacinto 7
SYSFW	系统固件
TISCI	德州仪器 (TI) 系统控制器接口
TIFS	德州仪器 (TI) 基础安全
DM	设备管理器
RM	资源管理
PM	电源管理
HSM	硬件安全模块
DMSC	设备管理和安全控制器
MCU	微控制器单元
MCU R5F	MCU 域中的 Arm Cortex-R5F
CSL	芯片支持库
PDK	平台开发套件
SDK	软件开发套件
TCM	紧耦合存储器
M3	Arm Cortex-M3
R5F	Arm Cortex-R5F

2 目标受众

SYSFW 在安全管理器和设备管理器内核上执行。在 J7 器件上，DMSC 子系统是 SoC 中的安全管理器内核 (Arm® Cortex®-M3)。DMSC 本身可包含设备管理器 (DM) 内核，该内核也可以位于 MCU R5F 上。该内核是在 DMSC 还是 MCU R5F 上运行，具体取决于器件的目标应用类型。

对于无需专用安全岛或 HSM 的器件，设备管理器在 DMSC 上运行。如果器件面向的是需要专用 HSM 的应用，则设备管理器内核独立于 DMSC（专为实现安全功能而保留）。

表 2-1. 标题缺失

类别	器件	TIFS 内核	专用安全岛或 HSM？	第三方 HSM/SHE 堆栈内核	RM/PM (DM) 内核
1	DRA829/TDA4VM	DMSC (M3)	否	不适用	DMSC (M3)
2	DRA829/TDA4VM	DMSC (M3)	是	DMSC (Cortex-M3)	MCU R5F 上的库

本应用报告旨在满足上述第二种类别的需求。

3 Jacinto7 SDK V7.1+ 中 MCU1_0 的作用

从 TI Jacinto7 SDK V7.1 开始，MCU1_0 通过为该平台运行 DM 用作资源管理器和电源管理器。

MCU1_0 使用 sciclient_direct.aer5f 来实现 TISCI 函数调用。其他 MCU 或 DSP 使用 sciclient_indirect 库来实现与 RM/PM 相关的 TISCI 功能。

Sciserver 或 TISCI 服务器仅需在 MCU1_0 上托管，以便向 Jacinto 7 SOC 中的其他内核提供 RM/PM 功能。

4 TI-RTOS 上的 TISCI 客户端和服务器

在 TI Jacinto7 TI-RTOS SDK 中，TI 在 MCU1_0 上实现了 TISCI 服务器。在将 TISCI 服务器功能集成到 AUTOSAR 等其他 RTOS 上时，可以参考此处的信息。

有关该实现的详细信息，请参阅该 SDK 中的 SciClient 文档。

4.1 J7 SDK 下载

平台	器件	下载链接
J721E	DRA829	https://www.ti.com.cn/tool/cn/download/PROCESSOR-SDK-RTOS-J721E#downloads
	TDA4VM	
J7200	DRA821	https://www.ti.com.cn/tool/cn/download/PROCESSOR-SDK-RTOS-J7200#downloads

4.2 SciClient 驱动程序位置

表 4-1 显示了 SciClient 的驱动程序代码和公共存储库：

表 4-1. SciClient 的驱动程序代码和公共存储库

驱动程序代码	`$J7SDK/ti-processor-sdk-rtos-j721e-evm-xx_xx_xx_xx/pdk_jacinto_xx_xx_xx_xx/packages/ti/drv/sciclient`
存储库	PDK 的公共 Git

4.3 TISCI 服务器初始化示例

该代码片段提供了 SciClient 和 SciServer 初始化的一个实现示例。

```
sint32 SetupSciServer(void)
{
#if (defined (BUILD_MCU1_0) && (defined (SOC_J721E) || defined (SOC_J7200)))
    Sciserver_TirtosCfgPrms_t appPrms;
    Sciclient_ConfigPrms_t clientPrms;
    sint32 ret = CSL_PASS;

    appPrms.taskPriority[SCISERVER_TASK_USER_LO] = 1;
    appPrms.taskPriority[SCISERVER_TASK_USER_HI] = 4;

    /* Sciclient 需要在 Sciserver 之前被初始化。Sciserver 依赖
     * Sciclient API 来转发信息 */
    ret = Sciclient_configPrmsInit(&clientPrms);
    if (ret == CSL_PASS)
    {
        ret = Sciclient_init(&clientPrms);
    }

    if (ret == CSL_PASS)
    {
        ret = Sciserver_tirtosInit(&appPrms);
    }

    if (ret == CSL_PASS)
    {
        AppUtils_Printf(MSG_NORMAL, "Starting Sciserver.....PASSED\n");
    }
    else
    {
        AppUtils_Printf(MSG_NORMAL, "Starting Sciserver.....FAILED\n");
    }

#endif
    return ret;
}
```

Sciserver_tirtosInit() 的示例代码可以在 PDK 公共 GIT 中的 **$J7SDK/ti-processor-sdk-rtos-j721e-evm-xx_xx_xx_xx/pdk_jacinto_xx_xx_xx_xx/packages/ti/drv/sciclient/src/sciserver/sciserver_tirtos.c** 处找到，点击此处即可查看。

4.4 集成指南

AUTOSAR 和 TI-RTOS 之间存在差异，因此有多个实现需要移植。

如果 MCU1_0 不运行 TI-RTOS，而是运行 AUTOSAR 等操作系统，则应用程序开发人员应使用以下库，同时还要创建和编译等效文件并将该文件链接到 sciserver_tirtos.c。

例如，如节 4.3 所示，应当使用目标操作系统提供的 API 来实现 `ret = Sciserver_tirtosInit(&appPrms);` 函数。客户可以根据自有操作系统将 Sciserver_tirtosInit() 函数重命名为更合适的名称。

在实现 **Sciserver_tirtosInit**() 时需要下述操作系统组件。在该 SDK 中，目标操作系统为 TI RTOS，因此下述讨论均基于 TI-RTOS。后续部分将介绍如何将该实现移植到 AUTOSAR。

4.4.1 信标

在 TI-RTOS 中，信标用于：

- 触发 SciServer 用户任务执行
- 同步两个 SciServer 用户任务

用于初始化信标的代码可以在 **$J7SDK/ti-processor-sdk-rtos-j721e-evm-xx_xx_xx_xx/pdk_jacinto_xx_xx_xx_xx/packages/ti/drv/sciclient/src/sciserver/sciserver_tirtos.c** 函数 **Sciserver_tirtosInitSemaphores**() 和 PDK 的公共 GIT 中找到，点击此处即可查看。

4.4.2 中断注册

这里有四种中断用于 SciServer 用户任务处理。

下表提供了这些中断的详细信息。应在 MCU R5F 上为 SciServer 保留这四个中断。

	IRQ	说明
1	70	来自 MCU 域导航系统的高优先级请求
2	71	来自主域导航系统的高优先级请求
3	72	来自 MCU 域导航系统的正常优先级请求
4	73	来自主域导航系统的正常优先级请求

用于初始化 Hwis 的代码可以在 **$J7SDK/ti-processor-sdk-rtos-j721e-evm-xx_xx_xx_xx/pdk_jacinto_xx_xx_xx_xx/packages/ti/drv/sciclient/src/sciserver/sciserver_tirtos.c** 函数 **Sciserver_tirtosInitHwis ()** 和 PDK 公共 GIT 中找到，点击此处即可查看。

4.4.3 中断处理

SciServer 中断的处理程序代码可以在 **$J7SDK/ti-processor-sdk-rtos-j721e-evm-xx_xx_xx_xx/pdk_jacinto_xx_xx_xx_xx/packages/ti/drv/sciclient/src/sciserver/sciserver_tirtos.c** 函数 **Sciserver_tirtosUserMsgHwiFxn ()** 和 PDK 的公共 GIT 中找到，点击此处即可查看。

4.4.4 用户任务注册

存在两种不同优先级的用户任务：一种用于处理高优先级 TISCI 请求，而另一种用于处理正常优先级 TISCI 请求。

用于注册这些用户任务的代码可以在 **$J7SDK/ti-processor-sdk-rtos-j721e-evm-xx_xx_xx_xx/pdk_jacinto_xx_xx_xx_xx/packages/ti/drv/sciclient/src/sciserver/sciserver_tirtos.c** 函数 **Sciserver_tirtosInitUserTasks ()** 和 PDK 的公共 GIT 中找到，点击此处即可查看。

4.4.5 用户任务处理

具有较高优先级的用户任务是处理 MCU 域导航系统高优先级请求以及主域导航系统高优先级请求。而具有正常优先级的任务是处理 MCU 域导航系统正常优先级请求以及主域导航系统正常优先级请求。

用于处理这些用户任务的代码可以在 **$J7SDK/ti-processor-sdk-rtos-j721e-evm-xx_xx_xx_xx/pdk_jacinto_xx_xx_xx_xx/packages/ti/drv/sciclient/src/sciserver/sciserver_tirtos.c** 函数 **Sciserver_tirtosUserMsgTask ()** 和 PDK 公共 GIT 中找到，点击此处即可查看。

5 DaVinci 中的配置

5.1 DaVinci Developer

DaVinci Developer 是用于为 AUTOSAR ECU 设计软件组件 (SWC) 架构的工具。

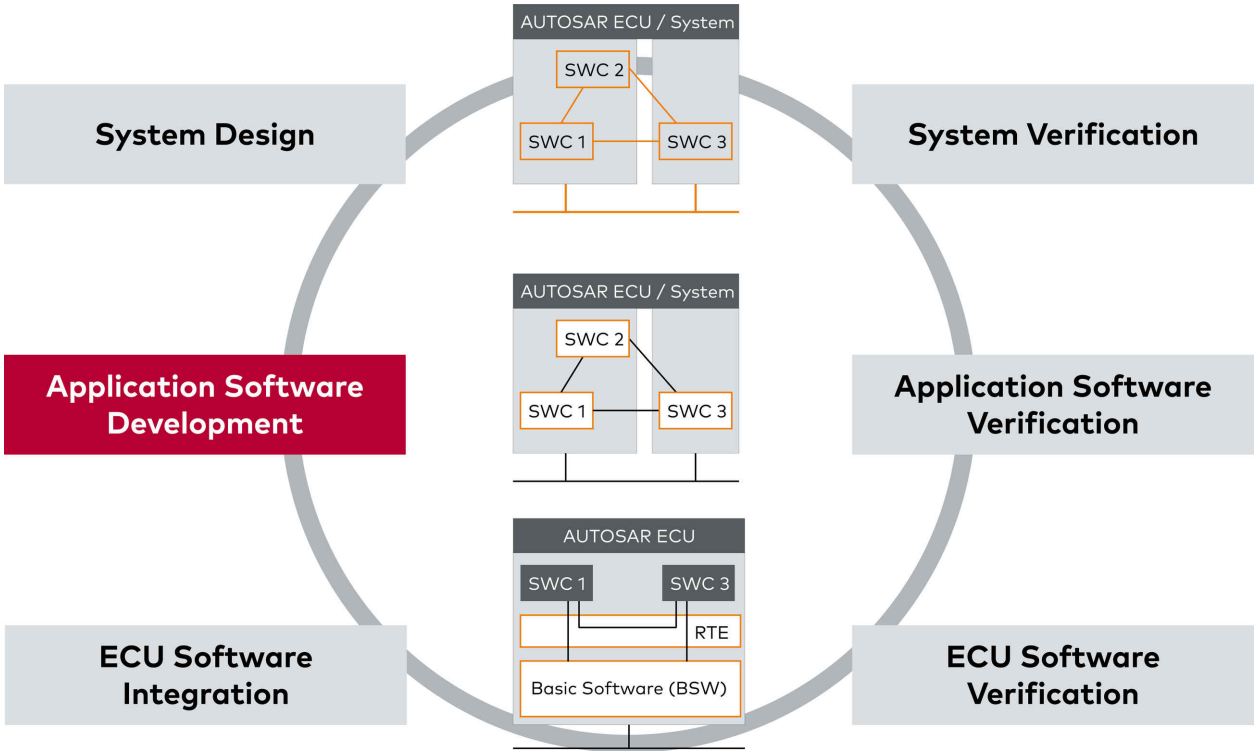

图 5-1. 典型 AUTOSAR 工程的循环模式

DaVinci Developer 用于应用软件开发阶段。

5.2 DaVinci Configurator Pro

DaVinci Configurator Pro 是用于配置、验证和生成 AUTOSAR ECU 基本软件 (BSW) 和运行时环境 (RTE) 的重要工具。

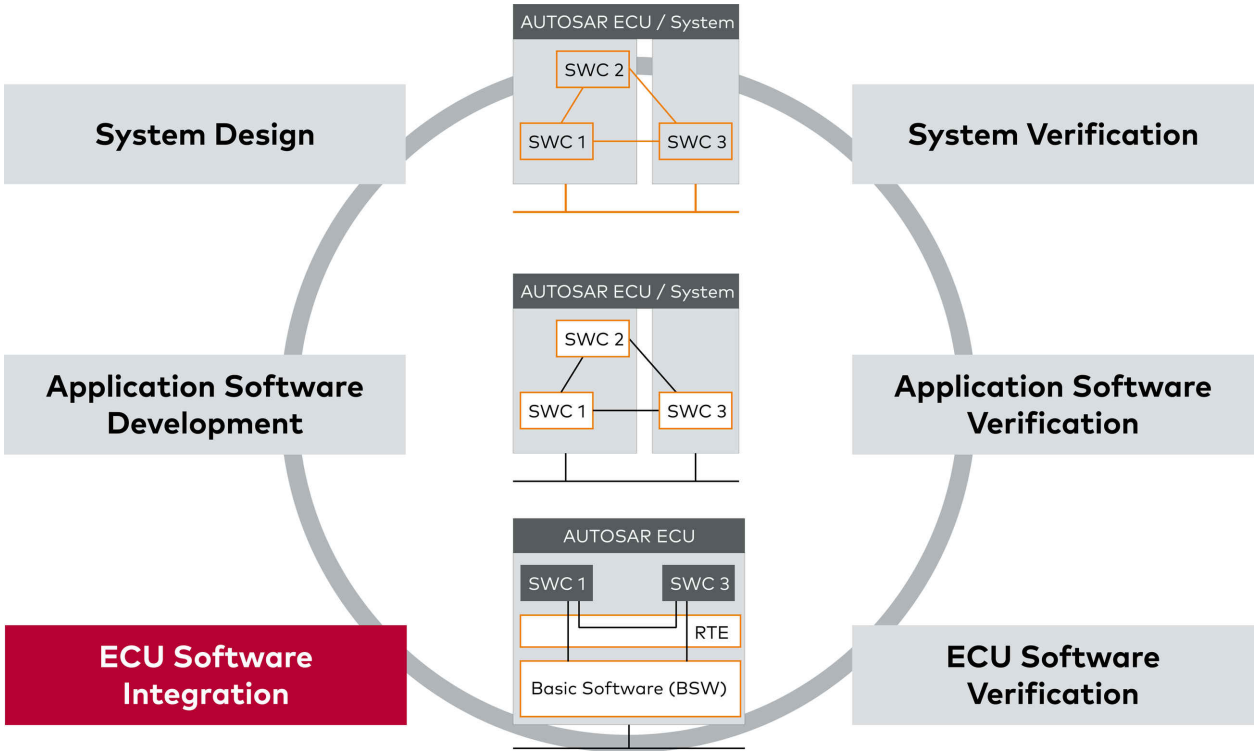

图 5-2. 典型 AUTOSAR 工程的循环模式

DaVinci Configurator Pro 用于 ECU 软件集成阶段。

更多相关信息，请访问：

https://www.vector.com/int/en/products/products-a-z/software/davinci-configurator-pro/

5.3 资源

OsResource 对象用于协调任务和 ISR 对共享资源的并发访问，例如调度程序、任意程序序列、存储器或任意硬件区域。更多相关信息，可在 AUTOSAR 网站上找到。

此资源用于同步两个 SciServer 用户任务。

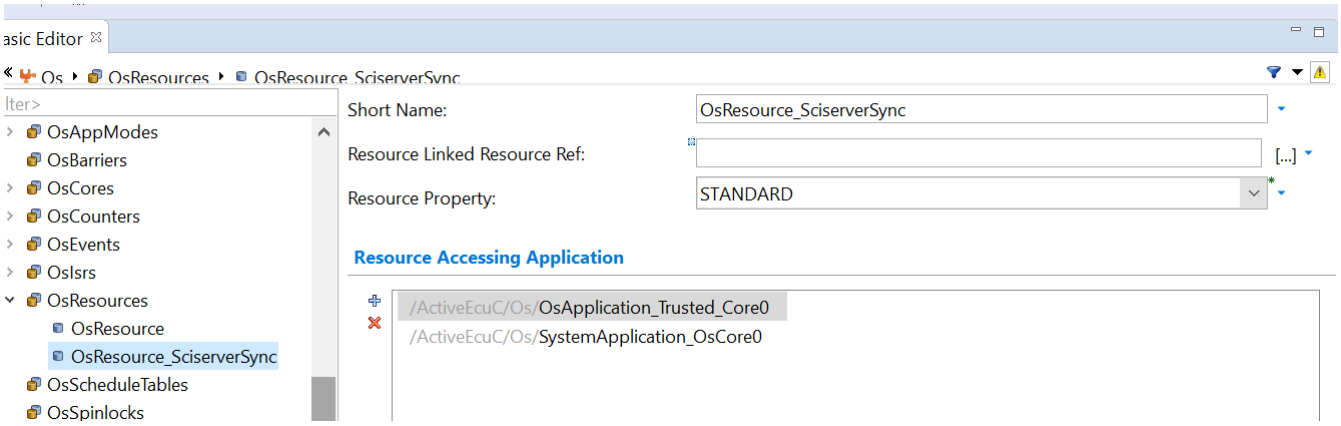

图 5-3. 资源配置

5.4 活动

AUTOSAR 中没有信标，因此事件用于从硬件中断触发 SciServer 用户任务执行。

5.4.1 高优先级请求事件

该事件将触发执行用于处理高优先级请求的 SciServer 用户任务。

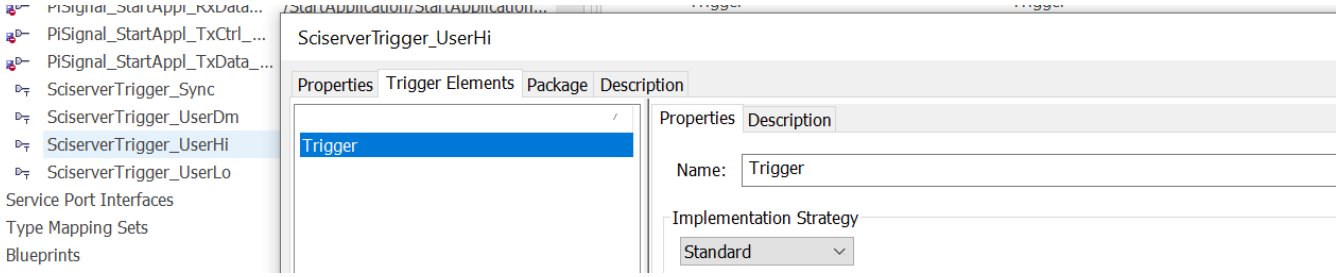

图 5-4. 高优先级请求的事件配置

5.4.2 正常优先级请求事件

该事件将触发执行用于处理正常优先级请求的 SciServer 用户任务。

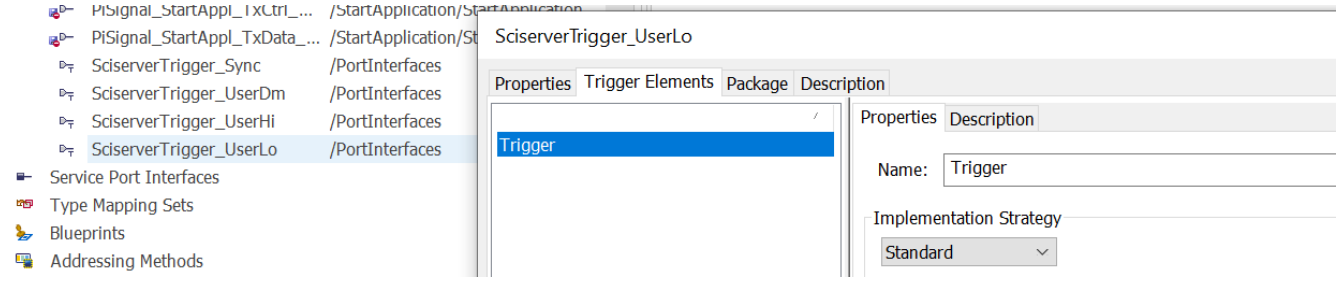

图 5-5. 正常优先级请求的事件配置

5.5 SciServer 用户任务

5.5.1 高优先级用户任务

该任务处理来自 MCU 域导航系统和主域导航系统的高优先级请求。

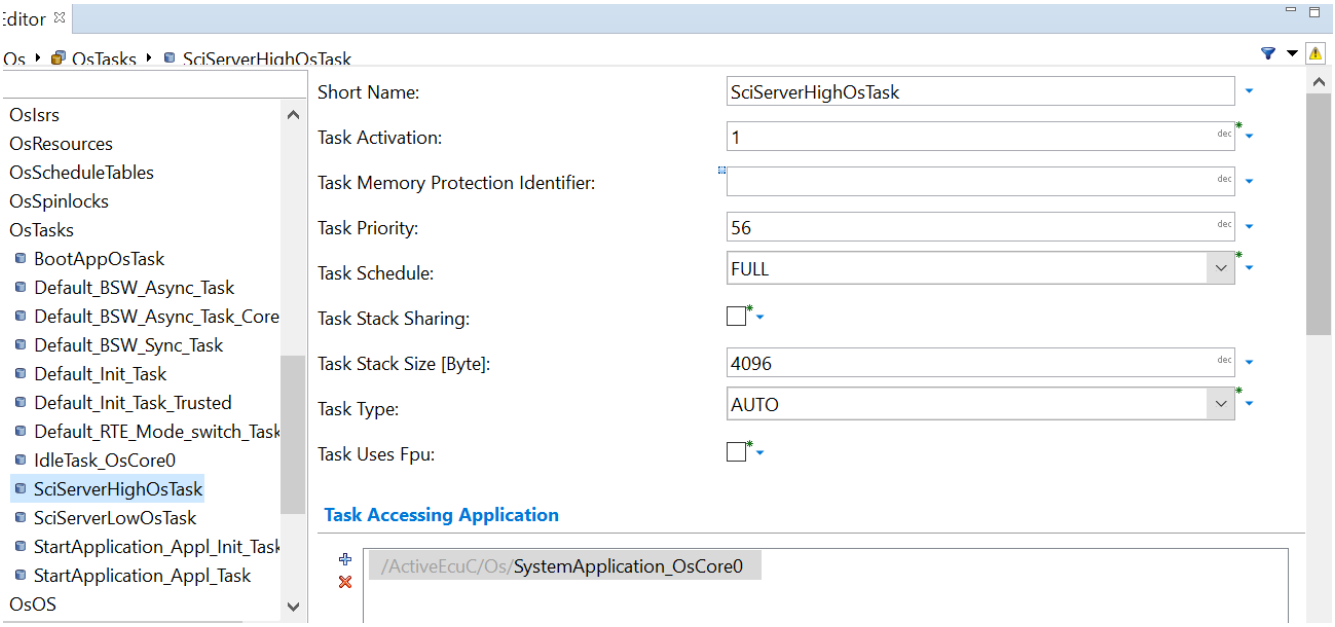

图 5-6. 高优先级用户任务配置

如节 5.4.1 所述，SciserverTrigger_UserHi 事件用于触发此任务执行。

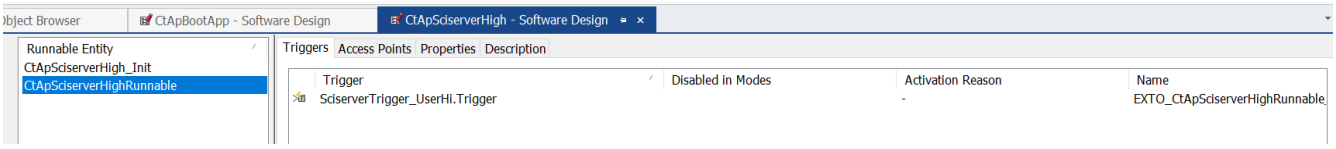

图 5-7. 高优先级用户任务触发器

5.5.2 正常优先级用户任务

该任务处理来自 MCU 域导航系统和主域导航系统的正常优先级请求。

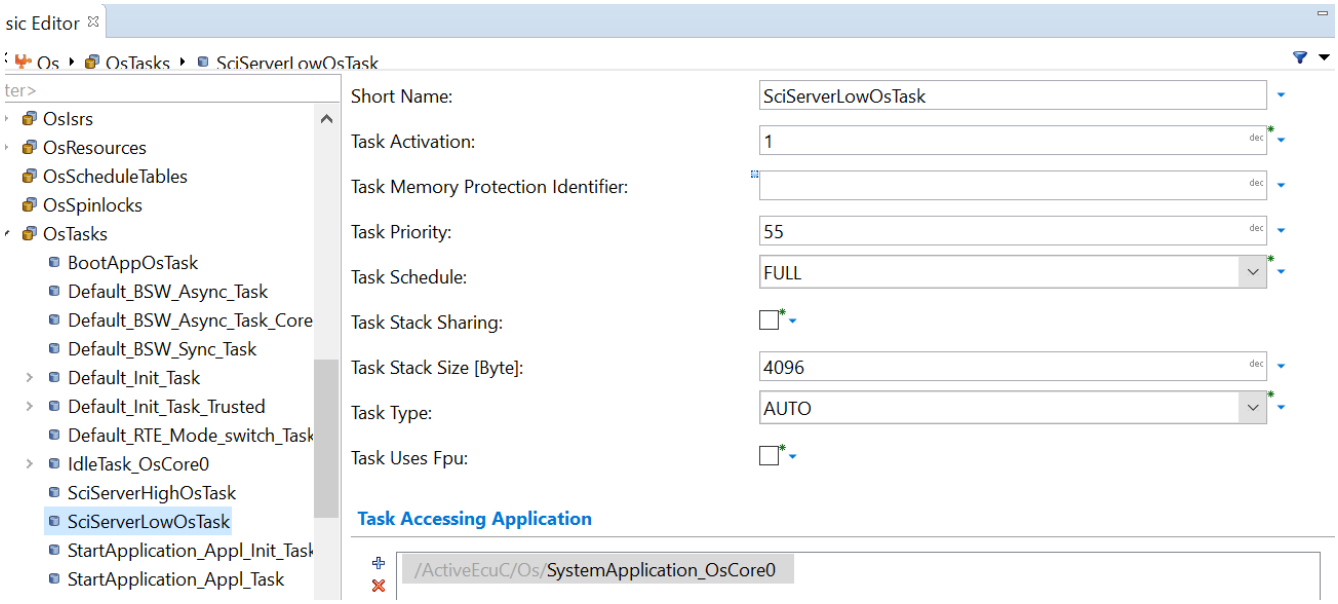

图 5-8. 正常优先级用户任务配置

如节 5.4.2 所述，SciserverTrigger_UserLo 事件用于触发此任务执行。

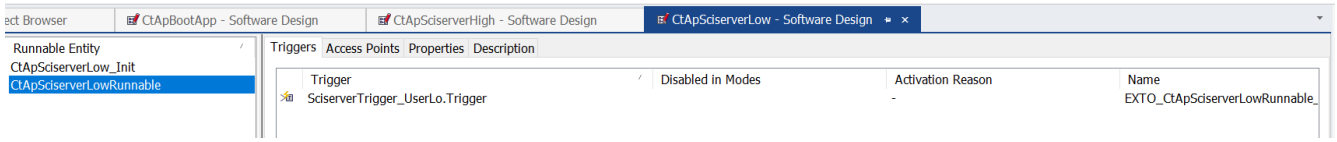

图 5-9. 正常优先级用户任务触发器

5.6 同步 Sciserver 用户任务

OSResource 用于同步两个 Sciserver 用户任务。

5.6.1 为高优先级用户任务配置资源

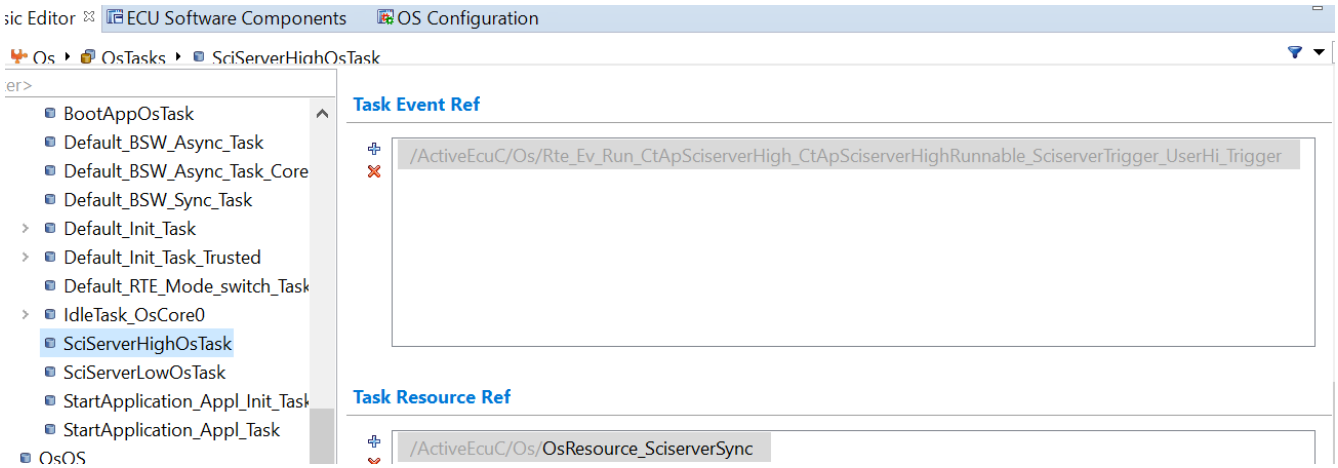

图 5-10. 高优先级用户任务的资源配置

5.6.2 为正常优先级用户任务配置资源

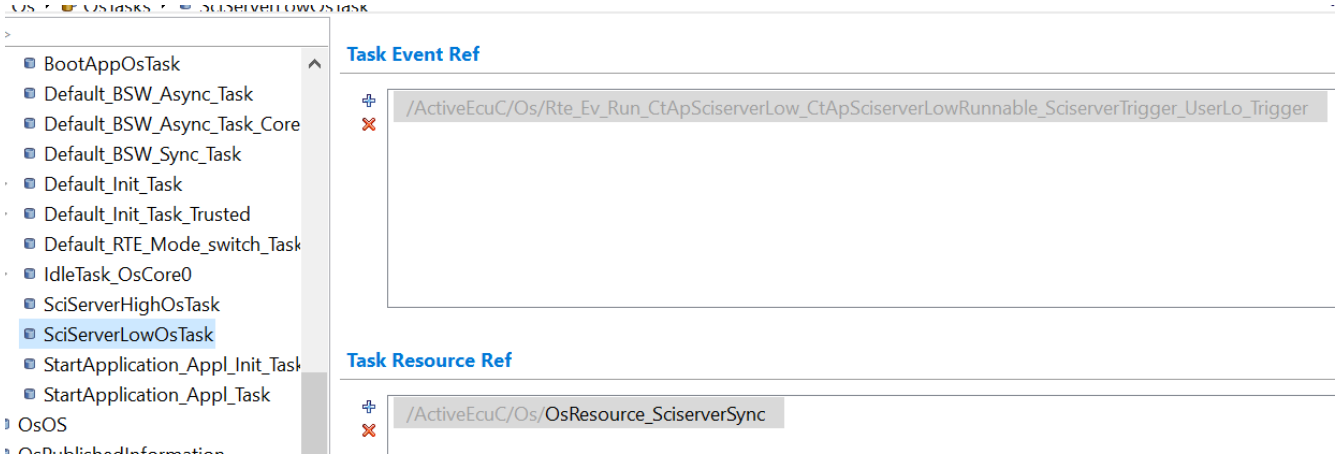

图 5-11. 正常优先级用户任务的资源配置

5.7 Sciserver 中断

如节 5.4.2 所述，Sciserver 需要四种中断。

下面是在 AUTOSAR 中注册这四种中断的示例。

5.7.1 MCU 域导航系统高优先级中断

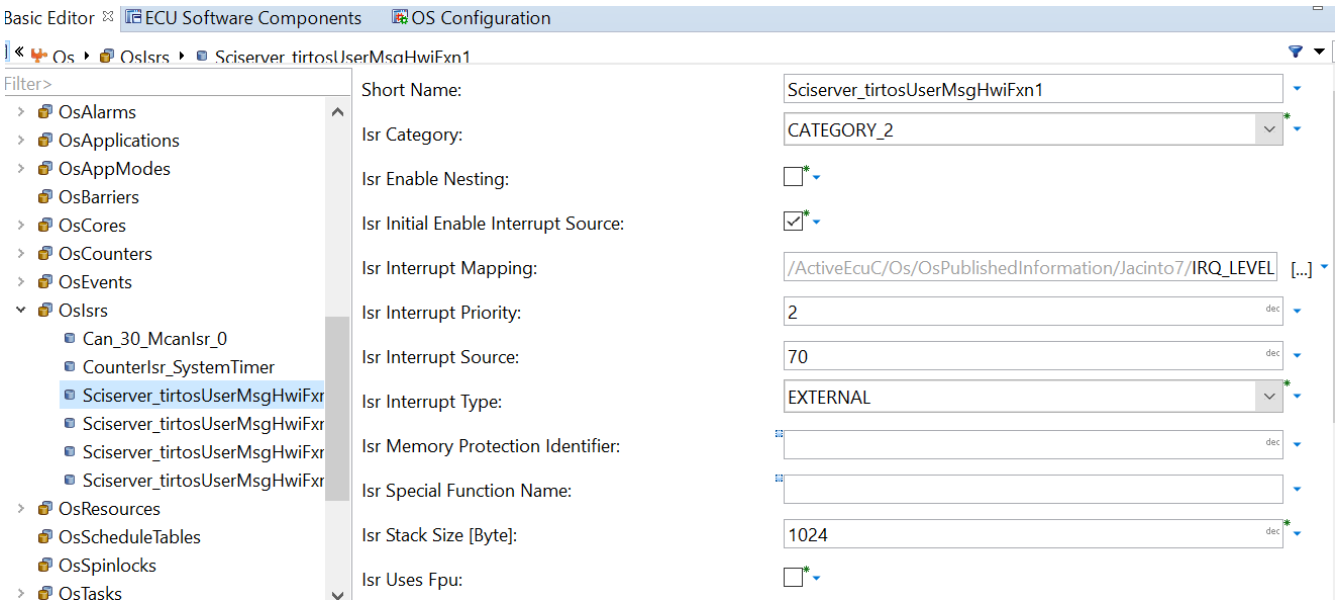

图 5-12. MCU 域高优先级中断的配置

5.7.2 主域导航系统高优先级中断

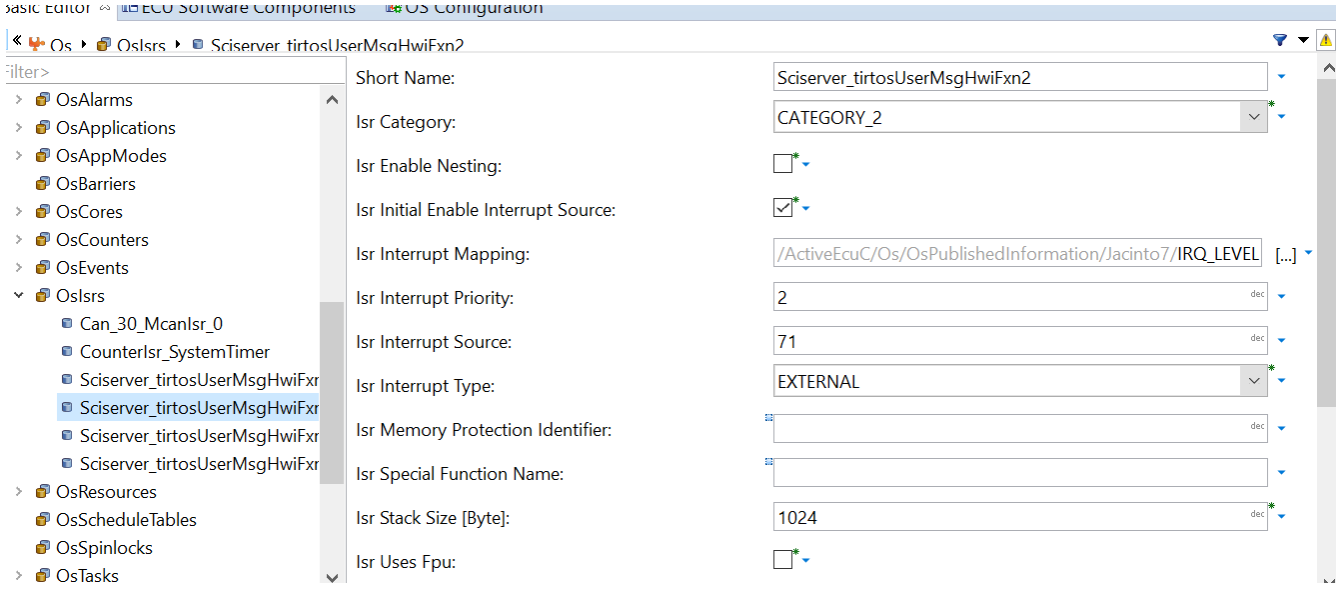

图 5-13. 主域高优先级中断的配置

5.7.3 MCU 域导航系统正常优先级中断

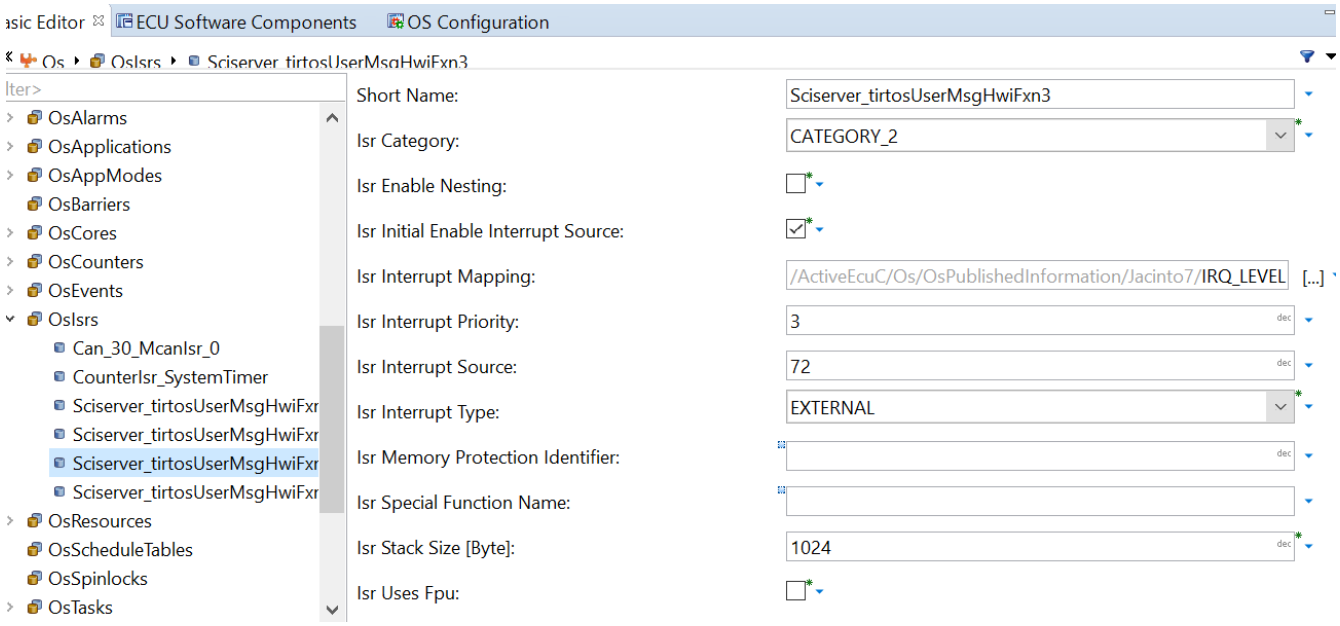

图 5-14. MCU 域正常优先级中断的配置

5.7.4 主域导航系统正常优先级中断

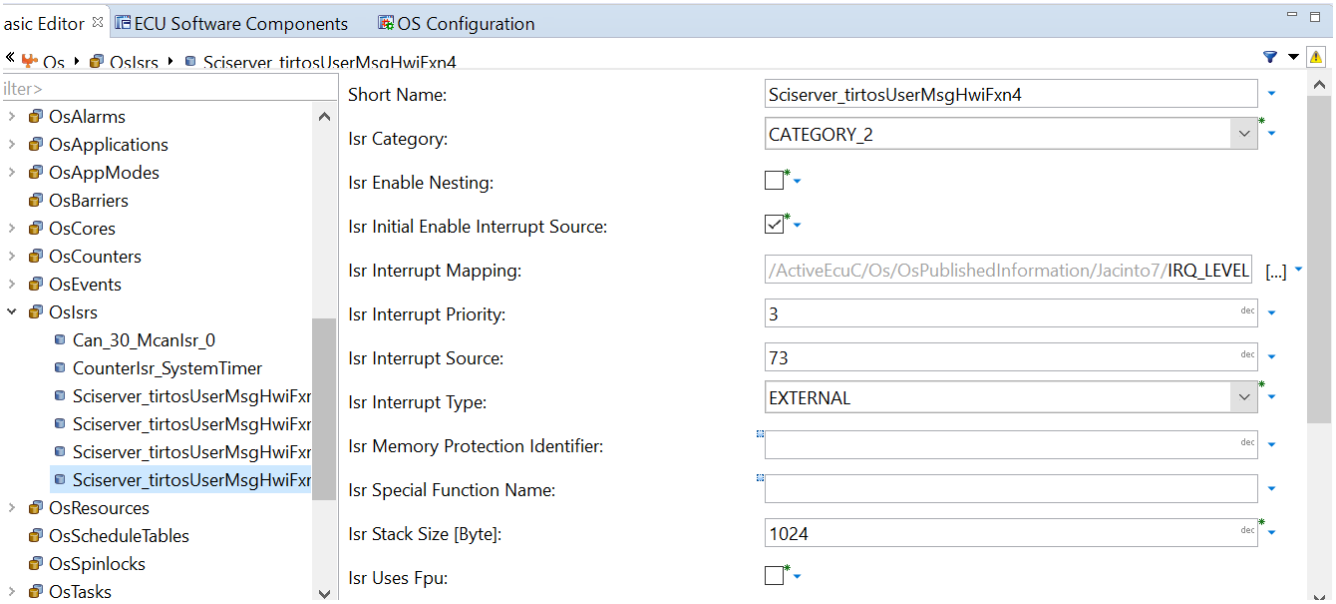

图 5-15. 主域正常优先级中断的配置

6 AUTOSAR TISCI 客户端

如节 1 所述，J7 SoC 中的每个内核都需要注册为 DMSC 的 SCI 客户端，以便从 DMSC 获取 SCI 服务。

6.1 AUTOSAR 中的 TISCI 客户端注册

以下示例显示了 TISCI 客户端注册信息 (Board_sysInit()) 在 AUTOSAR 中的位置。

```
void Brs_PreMainStartup(void)
{
  uint32 coreID;

  /*将向量重定位到 ATCM，请参阅本文档中的其他主题 */
  memcpy((void *)0, (void *)_OS_EXCVEC_CORE0_CODE_START, _OS_EXCVEC_CORE0_CODE_LIMIT);

  /*部分代码未在此处显示 */

  Board_sysInit();

  main();
}
```

其中 **Board_sysInit()** 可以参考以下位置的实现来定义： **$J7SDK/ti-processor-sdk-rtos-j721e-evm-xx_xx_xx_xx/pdk_jacinto_xx_xx_xx_xx/packages/ti/board/src/j721e_evm/board_init.c** 函数 **Board_sysInit()** 和 PDK 的公共 GIT，点击此处即可查看。

```
static int Board_sysInit(void)
{
    int status = 0;
    int ret;
    Sciclient_ConfigPrms_t config;

    if(gBoardSysInitDone == 0)
    {
        Sciclient_configPrmsInit(&config);

        ret = Sciclient_init(&config);
        if(ret != 0)
        {
            status = -1;
        }

        if(status == 0)
        {
            gBoardSysInitDone = 1;
        }
    }

    return status;
}
```

7 AUTOSAR TISCI 中断处理

下文所示的直接寄存器访问应该替换为 AUTOSAR 专用等效中断处理 API。

为了进行比较并更好地进行理解，注释行中显示了 TI-RTOS 中的示例实现。

7.1 MCU 域导航系统高优先级中断

节 5.7.1 显示了该中断的注册。

```
/* user_mcu_nav_high_priority */
ISR(Sciserver_tirtosUserMsgHwiFxn1)
{
    Sciserver_hwiData *uhd =  &sciserver_hwi_list[USER_MCU_NAV_HIGH];
    int32_t ret = CSL_PASS;
    bool soft_error = false;

    /* TI RTOS: Osal_DisableInterrupt(0, (int32_t) uhd->irq_num); */
    *(volatile unsigned int *)(0x40F80000 + 0x400 +
        (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_6/32)*0x20 + 0x0C) = 0x40;

    ret = Sciserver_interruptHandler(uhd, &soft_error);

    if ((ret != CSL_PASS) && (soft_error == true))
    {
        /* TI RTOS: Osal_EnableInterrupt(0, (int32_t) uhd->irq_num); */
        *(volatile unsigned int *)(0x40F80000 + 0x400 +
            (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_6/32)*0x20 + 0x08) =
0x40;
    }
    else
    {
         /* TI RTOS: (void) SemaphoreP_post(gSciserverUserSemHandles[uhd->semaphore_id]); */
        (void)SetEvent(SciServerHighOsTask,
            Rte_Ev_Run_CtApSciserverHigh_CtApSciserverHighRunnable_SciserverTrigger_UserHi_Trigger);
    }

    /* TI RTOS: Osal_ClearInterrupt(0, (int32_t) uhd->irq_num); */
    *(volatile unsigned int *)(0x40F80000 + 0x400 +
        (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_6/32)*0x20 + 0x04) = 0x40;

}
```

7.2 主域导航系统高优先级中断

节 5.7.2 显示了该中断的注册。

```
/* user_main_nav_high_priority */
ISR(Sciserver_tirtosUserMsgHwiFxn2)
{
    Sciserver_hwiData *uhd = NULL;
    int32_t ret = CSL_PASS;
    bool soft_error = false;

    uhd =   &sciserver_hwi_list[USER_MAIN_NAV_HIGH];

    /* TI RTOS: Osal_DisableInterrupt(0, (int32_t) uhd->irq_num); */
    *(volatile unsigned int *)(0x40F80000 + 0x400 +
        (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_7/32)*0x20 + 0x0C) = 0x80;

    ret = Sciserver_interruptHandler(uhd, &soft_error);

    if ((ret != CSL_PASS) && (soft_error == true))
    {
        /* TI RTOS: Osal_EnableInterrupt(0, (int32_t) uhd->irq_num); */
        *(volatile unsigned int *)(0x40F80000 + 0x400 +
            (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_7/32)*0x20 + 0x08) =
0x80;
    }
    else
    {
        /* TI RTOS: (void) SemaphoreP_post(gSciserverUserSemHandles[uhd->semaphore_id]); */
        (void)SetEvent(SciServerHighOsTask,
            Rte_Ev_Run_CtApSciserverHigh_CtApSciserverHighRunnable_SciserverTrigger_UserHi_Trigger);
    }

    /* TI RTOS: Osal_ClearInterrupt(0, (int32_t) uhd->irq_num); */
    *(volatile unsigned int *)(0x40F80000 + 0x400 +
        (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_7/32)*0x20 + 0x04) = 0x80;

}
```

7.3 MCU 域导航系统正常优先级中断

节 5.7.3 显示了该中断的注册。

```
/* user_mcu_nav_low_priority */
ISR(Sciserver_tirtosUserMsgHwiFxn3)
{
    Sciserver_hwiData *uhd =  &sciserver_hwi_list[USER_MCU_NAV_LOW];
    int32_t ret = CSL_PASS;
    bool soft_error = false;

    /* TI RTOS: Osal_DisableInterrupt(0, (int32_t) uhd->irq_num); */
    *(volatile unsigned int *)(0x40F80000 + 0x400 +
        (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_8/32)*0x20 + 0x0C) = 0x100;

    ret = Sciserver_interruptHandler(uhd, &soft_error);

    if ((ret != CSL_PASS) && (soft_error == true))
    {
        /* TI RTOS: Osal_EnableInterrupt(0, (int32_t) uhd->irq_num); */
        *(volatile unsigned int *)(0x40F80000 + 0x400 +
            (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_8/32)*0x20 + 0x08) =
0x100;
    }
    else
    {
        /* TI RTOS: (void) SemaphoreP_post(gSciserverUserSemHandles[uhd->semaphore_id]); */
        (void)SetEvent(SciServerLowOsTask,
            Rte_Ev_Run_CtApSciserverLow_CtApSciserverLowRunnable_SciserverTrigger_UserLo_Trigger);
    }

    /* TI RTOS: Osal_ClearInterrupt(0, (int32_t) uhd->irq_num); */
    *(volatile unsigned int *)(0x40F80000 + 0x400 +
        (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_8/32)*0x20 + 0x04) = 0x100;

}
```

7.4 主域导航系统正常优先级中断

节 5.7.4 显示了该中断的注册。

```
/* user_main_nav_low_priority */
ISR(Sciserver_tirtosUserMsgHwiFxn4)
{
    Sciserver_hwiData *uhd =  &sciserver_hwi_list[USER_MAIN_NAV_LOW];
    int32_t ret = CSL_PASS;
    bool soft_error = false;

    /* TI RTOS: Osal_DisableInterrupt(0, (int32_t) uhd->irq_num); */
    *(volatile unsigned int *)(0x40F80000 + 0x400 +
        (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_9/32)*0x20 + 0x0C) = 0x200;

    ret = Sciserver_interruptHandler(uhd, &soft_error);

    if ((ret != CSL_PASS) && (soft_error == true))
    {
        /* TI RTOS: Osal_EnableInterrupt(0, (int32_t) uhd->irq_num); */
        *(volatile unsigned int *)(0x40F80000 + 0x400 +
            (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_9/32)*0x20 + 0x08) =
0x200;
    }
    else
    {
        /* TI RTOS: (void) SemaphoreP_post(gSciserverUserSemHandles[uhd->semaphore_id]); */
        (void)SetEvent(SciServerLowOsTask,
            Rte_Ev_Run_CtApSciserverLow_CtApSciserverLowRunnable_SciserverTrigger_UserLo_Trigger);
    }

    /* TI RTOS: Osal_ClearInterrupt(0, (int32_t) uhd->irq_num); */
    *(volatile unsigned int *)(0x40F80000 + 0x400 +
        (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_9/32)*0x20 + 0x04) = 0x200;

}
```

8 AUTOSAR TISCI 用户任务处理

下文所示的直接寄存器访问应该替换为 AUTOSAR 专用等效中断处理 API。

为了进行比较并更好地进行理解，注释行中显示了 TI-RTOS 中的示例实现。

8.1 高优先级用户任务初始化

节 5.5.1 中显示了此任务的创建。

```
static Sciserver_taskData *utdHigh = NULL;
static volatile int highIsrEnableVal = 0;

FUNC(void, CtApSciserverHigh_CODE) CtApSciserverHigh_Init(void)
{
    utdHigh = &gSciserverTaskList[SCISERVER_TASK_USER_HI];

    /* 首先设置等待状态 */
    utdHigh->state->state = SCISERVER_TASK_PENDING;
}
```

8.2 高优先级用户任务可运行文件

```
FUNC(void, CtApSciserverHigh_CODE) CtApSciserverHighRunnable(void)
{
    sint32 ret;

    GetResource(OsResource_SciserverSync);

    ret = Sciserver_processtask(utdHigh);
    if (ret != CSL_PASS)
    {
        /* 消息处理失败，发送 nak 响应失败 */
        /* TI-RTOS: BIOS_exit(0); */
        ReleaseResource(OsResource_SciserverSync);
       (void)TerminateTask();
    }
    else
    {
        /* TI-RTOS:
        Osal_EnableInterrupt(0, sciserver_hwi_list[2U * utd->task_id +
                utd->state->current_buffer_idx].irq_num);  */
        highIsrEnableVal = 1 << ((sciserver_hwi_list[2U * utdHigh->task_id + utdHigh->state-
>current_buffer_idx].irq_num) % 32);
        *(volatile unsigned int *)(0x40F80000 + 0x400 +
            (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_7/32)*0x20 + 0x08) =
highIsrEnableVal;
    }

    ReleaseResource(OsResource_SciserverSync);
}
```

8.3 正常优先级用户任务初始化

节 5.5.2 中显示了此任务的创建。

```
static Sciserver_taskData *utdLow = NULL;
static volatile int lowIsrEnableVal = 0;

FUNC(void, CtApSciserverLow_CODE) CtApSciserverLow_Init(void)
{
    utdLow = &gSciserverTaskList[SCISERVER_TASK_USER_LO];

    /* 首先设置等待状态 */
    utdLow->state->state = SCISERVER_TASK_PENDING;
}
```

8.4 正常优先级用户任务可运行文件

```
FUNC(void, CtApSciserverLow_CODE) CtApSciserverLowRunnable(void)
{
    sint32 ret;

    GetResource(OsResource_SciserverSync);

    ret = Sciserver_processtask(utdLow);
    if (ret != CSL_PASS)
    {
        /* 消息处理失败，发送 nak 响应失败 */
        /* TI-RTOS: BIOS_exit(0); */
        ReleaseResource(OsResource_SciserverSync);
       (void)TerminateTask();
    }
    else
    {
        /* TI-RTOS:
           Osal_EnableInterrupt(0, sciserver_hwi_list[2U * utd->task_id +
                utd->state->current_buffer_idx].irq_num);  */
        lowIsrEnableVal = 1 << ((sciserver_hwi_list[2U * utdLow->task_id + utdLow->state-
>current_buffer_idx].irq_num) % 32);
        *(volatile unsigned int *)(0x40F80000 + 0x400 +
            (CSLR_MCU_R5FSS0_CORE0_INTR_MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_8/32)*0x20 + 0x08) =
lowIsrEnableVal;
    }
```

```
    ReleaseResource(OsResource_SciserverSync);
}
```

9 AUTOSAR 中的 TISCI 服务器验证

9.1 Boot App

Boot App 是 MCU1_0 用于在 SoC 上引导其他内核的应用程序。

相关文档可以在 Jacinto7 SDK 中找到：MCUSW boot app。

如果在 AUTOSAR OS 中正确实现了 TISCI 客户端和 TISCI 服务器，MCU1_0 应该能够成功地加载并引导其他内核映像。

9.2 引导任务配置

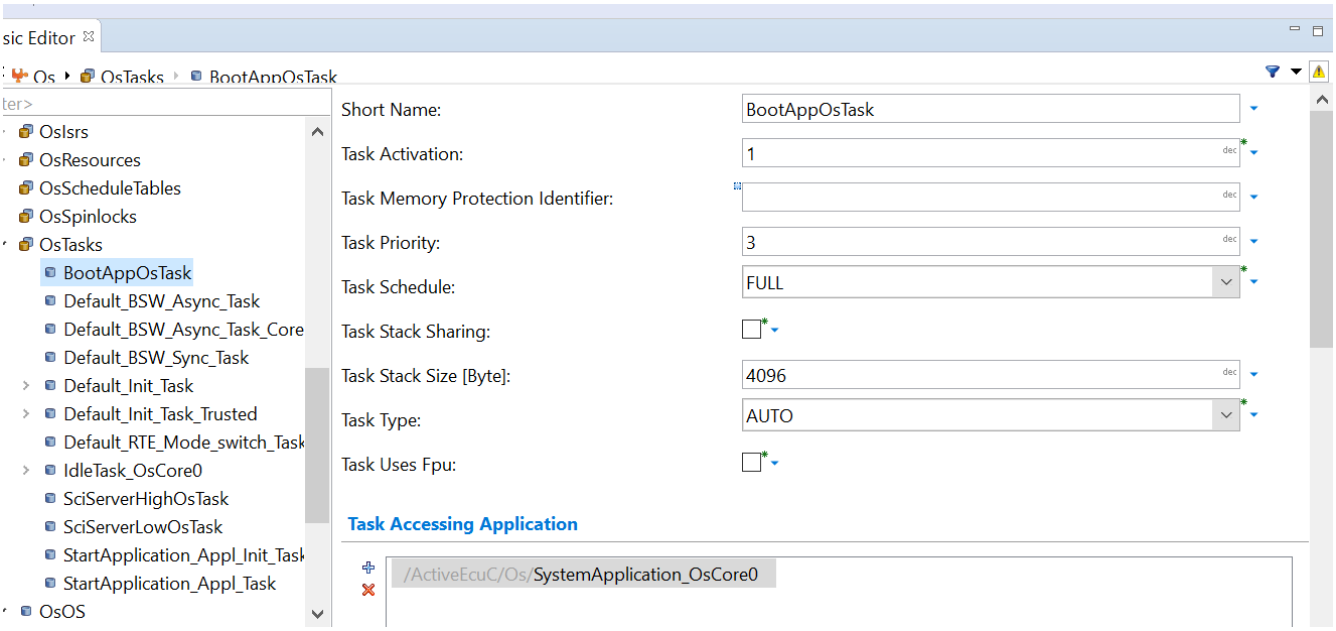

图 9-1. 引导任务配置

无需引导任务可运行文件。

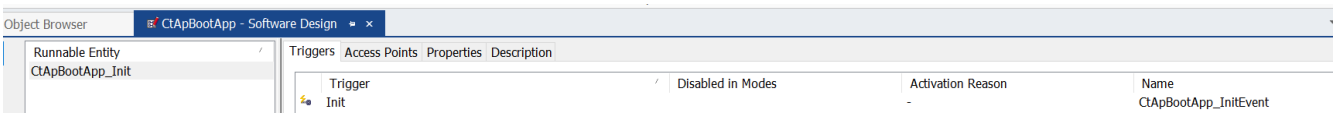

图 9-2. 引导任务触发器

9.3 AUTOSAR 中的 Boot App

9.3.1 Boot App 启动

```
FUNC(void, CtApBootApp_CODE) CtApBootApp_Init(void)
{
    Boot_App();
}
```

9.3.2 Boot App 实现

“**Boot_App()**”的代码可以在 **$J7SDK/ti-processor-sdk-rtos-j721e-evm-xx_xx_xx_xx/mcusw/mcuss_demos/boot_app_mcu_rtos/boot.c** 函数 **Boot_App ()** 中找到。

以下是 Boot_App 函数的代码片段。

```
/* Main Boot task */
static sint32 Boot_App()
{
    sint32          retVal;
    cpu_core_id_t   core_id, *boot_array;
    uint8           i, j, num_cores_to_boot, num_booted_cores = 0;

    MainDomainBootSetup();
    SBL_SPI_init();
    SBL_ospiInit(&boardHandle);
    /* Initialize the entry point array to 0.*/
    for (core_id = MPU1_CPU0_ID; core_id < NUM_CORES; core_id ++) {
        (&k3xx_evmEntry)->CpuEntryPoint[core_id] = SBL_INVALID_ENTRY_ADDR;
    }
    for (j = 0; j < NUM_BOOT_STAGES; j++) {
        retVal = RequestStageCores(j);
        if (retVal != CSL_PASS) {
            ReleaseStageCores(j);
        } else {
            retVal = OSPIBootStageImage(&k3xx_evmEntry, ospi_main_domain_flash_rtos_images[j]);
            if (retVal != CSL_PASS){
            } else {
                retVal = ReleaseStageCores(j);
                if (retVal != CSL_PASS) {
                }
            }
        } /* if (retVal != CSL_PASS) */
        if (retVal == CSL_PASS) {
            /* Start the individual cores for the boot stage */
            num_cores_to_boot = num_cores_per_boot_stage[j];
            boot_array        = boot_array_stage[j];

            for (i = 0; i < num_cores_to_boot; i++) {
                core_id = boot_array[i];
                /* Try booting all cores other than the cluster running the SBL */
                if ((k3xx_evmEntry.CpuEntryPoint[core_id] != SBL_INVALID_ENTRY_ADDR) &&
                    ((core_id != MCU1_CPU1_ID) && (core_id != MCU1_CPU0_ID))) {
                    SBL_SlaveCoreBoot(core_id, NULL, &k3xx_evmEntry,SBL_REQUEST_CORE);
                    num_booted_cores++;
                }
            }
        } /* if (retVal == CSL_PASS) */
    } /* for (j = 0; j < NUM_BOOT_STAGES; j++) */

    SBL_ospiClose(&boardHandle);

    return (retVal);
}
```

10 AUTOSAR 中使用的 PDK 库

下面是 AUTOSAR 中使用的库二进制文件，这些文件都来自 TI J7 SDK 的 PDK。

```
PDK Path:
        $J7SDK/ti-processor-sdk-rtos-j721e-evm-xx_xx_xx_xx/pdk_jacinto_xx_xx_xx_xx/packages
```

- rm_pm_hal.aer5f
- sbl_lib_cust.aer5f
- sciclient_direct.aer5f
- sciserver_baremetal.aer5f
- ti.board.aer5f
- ti.csl.aer5f
- ti.csl.init.aer5f
- ti.drv.spi.aer5f
- ti.osal.aer5f
- ipc_baremetal.aer5f

11 AUTOSAR 所需的 R5F 配置

11.1 Cortex-R5F 的存储器布局

表 11-1 显示了 Cortex-R5F 中的 ATCM 和 BTCM。

表 11-1. Cortex-R5F 中的 ATCM 和 BTCM

区域名称	起始地址	终止地址	大小
ARMSS_ATCM	0x0000 0000	0x0000 7FFF	32KB
ARMSS_BTCM	0x4101 0000	0x4101 7FFF	32KB

在 Cortex-R5F 上运行 AUTOSAR 时，这里有多个检查点。

- **ATCM 使用情况：**R5 SPL 默认不启用 MCU1_0 的 ATCM，因此如果在 MCU1_0 内核上从 R5 SPL 引导 AutoSAR 应用程序，那么不应在存储器布局中使用 ATCM。
  - 在这种情况下，Startup_Code 应该置于 BTCM 中。
- **Startup_Code 对齐方式：**应该为 256 字节对齐。

示例如下所示：

```
MEMORY
{
  OCMCRAM_Common : ORIGIN = 0x41C50000 , LENGTH = 0x00004000 /* 48 KiB */
  OCMCRAM_Common_NonCache : ORIGIN = 0x41C54000 , LENGTH = 0x00000400 /* 1024 Byte */
  OCMCRAM_Core0 : ORIGIN = 0x41C54400 , LENGTH = 0x00000800 /* 2048 Byte */
  OCMCRAM_Core1 : ORIGIN = 0x41C54C00 , LENGTH = 0x00000400 /* 1024 Byte */
  OCMCRAM_Core2 : ORIGIN = 0x41C55000 , LENGTH = 0x00000400 /* 1024 Byte */
  OCMCRAM_Core3 : ORIGIN = 0x41C55400 , LENGTH = 0x00000400 /* 1024 Byte */
  OCMCRAM_Core4 : ORIGIN = 0x41C55800 , LENGTH = 0x00000400 /* 1024 Byte */
  OCMCRAM_Core5 : ORIGIN = 0x41010000 , LENGTH = 0x00000400 /* 1024 Byte */
  DDR0 : ORIGIN = 0x41C55C00 , LENGTH = 0xA5000 /* 16 MiB */
}

  .Startup_Code : ALIGN(256)
  {
    _Startup_Code_START = .;
    *(.brsStartup)
    .= ALIGN(256);
    _Startup_Code_END = .- 1;
    _Startup_Code_LIMIT = .;
  } > OCMCRAM_Core5
```

11.2 R5F 缓存配置

共有三处具有 R5F 缓存配置。sbl_main.c 中的配置适用于 AUTOSAR。此配置在 J7 SDK 中广泛使用。

- SBL 缓存配置
  - **$SDK_Path/ti-processor-sdk-rtos-$platform-evm-$psdkra_ver/pdk_jacinto_$pdk_ver/packages/ti/boot/sbl/board/k3/sbl_main.c** ，名为 **gCslR5MpuCfg** 的结构的一部分
- Boot App 缓存配置
  - **$SDK_Path/ti-processor-sdk-rtos-$platform-evm-$psdkra_ver/mcusw/boot_app_mcu_rtos_overrides/j721e/r5_mpu.xs**。此配置包含在 TI RTOS 配置文件中，对应路径为 **$SDK_Path/ti-processor-sdk-rtos-$platform-evm-$psdkra_ver/mcusw/boot_app_mcu_rtos_overrides/j721e/sysbios_r5f.cfg**。
- CSL 缓存配置（由裸机应用覆盖）
  - **$SDK_Path/ti-processor-sdk-rtos-$platform-evm-$psdkra_ver/pdk_jacinto_$pdk_ver/packages/ti/csl/arch/r5/src/startup/startup.c**。

重要声明和免责声明

TI 提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保或其他要求。这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款 (https:www.ti.com/legal/termsofsale.html) 或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265