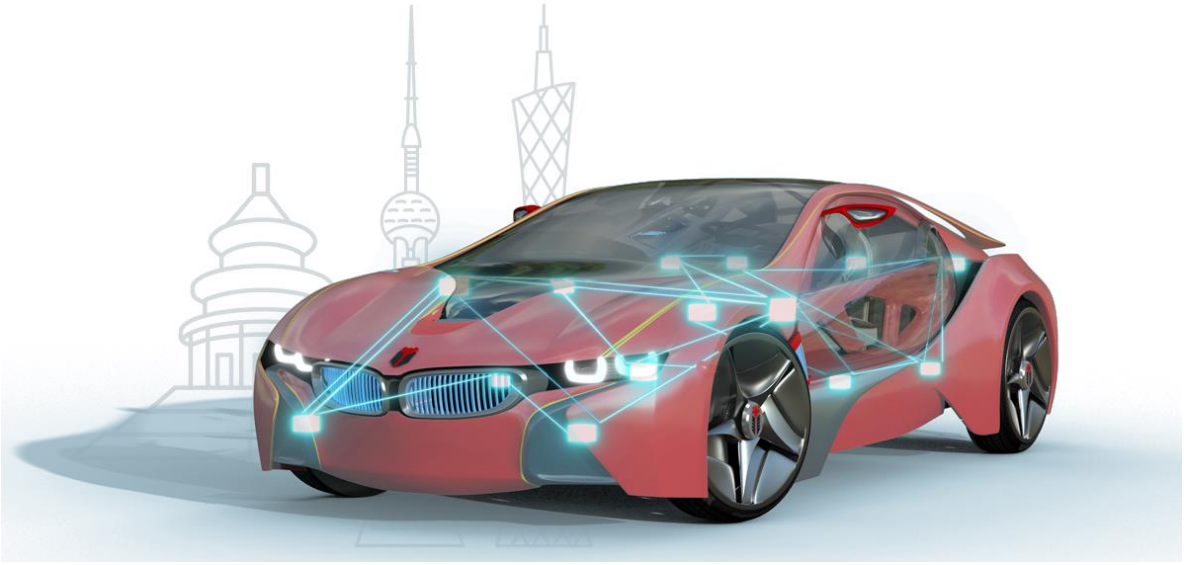
瑞萨 RH850 G3MH/G3KH 到 ARM CORTEX-R52/R52+的迁移指南

ZC®工程服务



瑞萨 RH850 G3MH/G3KH 到 ARM CORTEX-R52/R52+ 的迁移指南

ZC®工程服务

第 1 章 概述

本文为从瑞萨RH850 G3MH/G3KH迁移到Arm Cortex-R52/R52+的用户提供了指南。

1.1 芯片平台介绍

1.1.1 ARM 平台介绍

ARM 是 Advanced RISC Machines Limited 公司推出的一种 RISC 处理器体系结构的相关技术。ARM 架构广泛使用在嵌入式系统设计，具有低功耗特性，适用于移动通讯领域、消费电子产品，例如移动电话、多媒体播放器、掌上型电子游戏、计算机、电脑外设等。同时也适用于工业、汽车、航天等领域。

ARM 处理器内核分为三个系列：Cortex-A 系列、Cortex-R 系列和 Cortex-M 系列。

- Cortex-A 系列

Application Processors 应用处理器。包括 Cortex-A5、Cortex-A8、Cortex-A9、Cortex-A15、Cortex-A5x、Cortex-A7x、Cortex-A71x 等基于 ARMv7-A、ARMv8-A 和 ARMv9-A 架构的处理器。此系列架构为运行复杂操作系统（例如 Linux、Android、IOS）的设备提供了一系列解决方案。广泛适用于从低成本手持设备到智能手机、平板电脑、机顶盒以及企业网络设备等各类应用，可以进行海量数据处理和高性能计算。这类处理器一般运行在很高的主频下（一般超过 1GHz），支持像 Linux，Android，Windows 和移动操作系统等需要的内存管理单元（MMU）等功能。

- Cortex-R 系列

Real-time Processors 实时处理器。包括 Cortex-R4、Cortex-R5、Cortex-R7、Cortex-R8、Cortex-R52 等基于 ARMv7-R、ARMv8-R 架构实现的处理器。Cortex-R 系列是一个实时微控制器内核系列，专为高安全和高性能嵌入式系统等领域而设计。 旨在提供快速且确定的响应时间，使其成为高实时性和高安全性应用领域的理想选择，例如汽车、工业和航空航天系统等。虽然实时处理器不能运行完整版本的 Linux 和 Windows 操作系统（Cortex-R82 除外）， 但是支持大量的实时操作系统（RTOS）。

- Cortex-M 系列

Microcontroller Processors 微控制器处理器。包括 Cortex-M0、Cortex-M0+、Cortex-M3、Cortex-M4、Cortex-M7、Cortex-M33、Cortex-M55 等基于 ARMv6-M、 ARMv7-M、ARMv8-M 架构实现的处理器。Cortex-M 系列是一种低功耗、高性能、可扩展性的处理器内核，包含许多特别适合嵌入式系统的特性，同时设计的非常容易使用，因此在单片机、IOT、嵌入式系统市场非常成功。 Cortex-M 系列已广泛应用于从消费电子产品到工业控制系统的一系列应用中，包括微控制器市场、IOT、嵌入式系统和汽车控制器等场景。

本文以 ARM 的 Cortex-R52/R52+作为移植的目标平台，其对应的 ARM 内核架构为 ARMv8-R。

1.1.2 瑞萨平台介绍

瑞萨拥有广泛的 8 位、16 位和 32 位产品组合，主要用于工业、物联网和基础设施等其他关键市场。

瑞萨处理器主要有采用 Arm Cortex-M 内核的 RA 系列 32 位 MCU、基于 32 位和 64 位 Arm 的 RZ 系列高性能微处理器、RL78 低功耗 8 位和 16 位 MCU、RX 系列 32 位高功效 MCU、RISC-V 32 位与 64 位 MCU 和 MPU、RH850 车用 MCU 等。

RH850 车用 MCU 分为 RH850 F1KM-S1/F1KM-S2/F1KM-S4/F1KH-D8/U2A/C1M-Ax 等多个系列，涵盖 G3MH/G3KH/G4MH 等多个内核架构。

本文以 RH850 G3MH&G3KH 作为移植的来源平台。

1.2 ARMv8-R 架构介绍

早期的 Cortex-R 处理器，如 Cortex-R5，建立在 Armv7-R 架构上。Cortex-R52 和 Cortex-R52+处理器实现了 Armv8-R 架构，有助于解决汽车实时软件日益复杂的问题，以及从离散专用控制器到功能集中和组合的控制器的过渡。Armv8-R 体系结构增加了支持，使其能够在单个处理器中更好地控制软件，提供代码的隔离，并支持可重复和可理解的行为，包括在实时性处理器中的虚拟化能力。

Armv8-R 架构的特点如下：

- 支持 T32 和 A32 指令集
- 提供了对 Armv7-R 指令集的向下兼容
- 使基于 Armv7-R 构建的程序可直接在 Armv8-R 环境下进行编译和构建
- 支持 EL0/EL1/EL2 三级异常级别，支持两段式 MPU。EL1 MPU 通常由操作系统管理，实现操作系统与应用的隔离，以及应用内部运行单元的隔离。EL2 MPU 通过运行在 EL2 上的代码进行编程，实现虚拟机级别的隔离控制

- 支持 GIC V3.0，提供了高效的中断响应和核间分发机制，支持多达 960 个核间共享中断（SPI）

1.3 RH850 G3MH&G3KH 架构介绍

瑞萨 RH850 G3MH/G3KH 采用 32 位体系结构，具备 32 位内部数据总线。瑞萨 RH850 G3MH/G3KH 采用 RISC 指令集，有 32 个 32 位通用寄存器。瑞萨 RH850 G3MH/G3KH 具备丰富的外设接口，包括 CAN、LIN、FlexRay 等常用的汽车总线接口，满足不同应用场景下的接口需求。

瑞萨 RH850 G3MH/G3KH 的主要特点包括:

- 支持用户模式和特权模式两级异常级别
- 具备内存保护单元
- 用于数据和指令的 4 GB 线性空间
- 中断控制器支持 16 个中断优先级
- 中断控制器支持快速中断上下文保存和恢复机制
- 具备电源管理机制，支持深度睡眠模式

第 2 章 内核架构对比

为了更清晰了解 ARM Cortex-R52/R52+和瑞萨 RH850 G3MH/G3KH 的区别，本文通过以下几个方面进行分析：

- Programmer's model

介绍两个架构的编程模式区别。编程模式是从开发者角度介绍了芯片的特性，主要从数据类型格式、运行模式等方面进行分析。

- Instruction set

介绍指令集区别。

- General Purpose Registers

介绍通用寄存器区别。

- Exceptions and interrupts

介绍中断和异常处理机制的区别，包括中断优先级、中断异常处理方式以及中断异常向量表等。

- Memory

介绍内存模型的区别，包括寻址空间、寻址方式、内存保护单元等。

- Debug

介绍调试系统区别。

2.1.1 Data Type

ARM Cortex-R52/R52+支持数据类型包括：Byte (8 bits)、Halfword (16 bits)、Word (32 bits)、Doubleword (64 bits)。ARM Cortex-R52/R52+同时支持 half-precision、single-precision、double-precision floating-point 数据类型。

瑞萨 RH850 G3MH/G3KH 支持数据类型包括：Byte (8 bits)、Halfword (16 bits)、Word (32 bits)、Double-word (64 bits)、Bit (1 bit)。

移植提示 1：每种数据类型的长度和所使用的编译器有关，在移植时需要确认在特定编译器下数据类型的长度。对于使用 C 语言的开发者，在编程时建议使用 sizeof 关键字获取数据类型的长度，而非直接写入固定的长度值。

移植提示 2：尽量减少使用位域（Bit 类型），这会降低代码的可移植性。

2.1.2 Byte ordering and Alignment

ARM Cortex-R52/R52+的数据存储方式可以使用小端格式（数据的低字节存储在内存的低地址），也可以使用大端格式（数据的高字节存储在内存的低地址）。

- ARM Cortex-R52/R52+的小端格式

一个字(4 字节)数据和半字(2 字节)数据在内存中存储方式如下图：

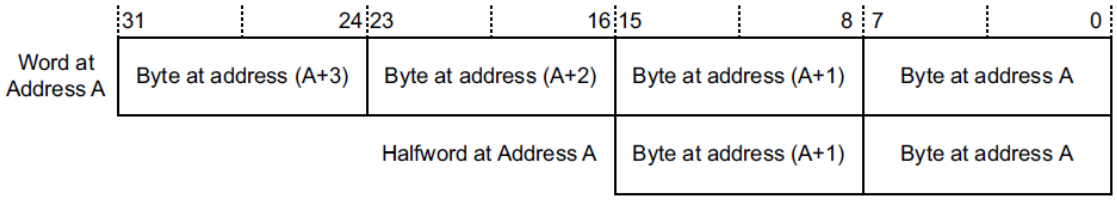


图 2.1.2-1 ARM Cortex-R52/R52+小端格式

- ARM 架构的大端格式

一个字(4 字节)数据和半字(2 字节)数据在内存中存储方式如下图：

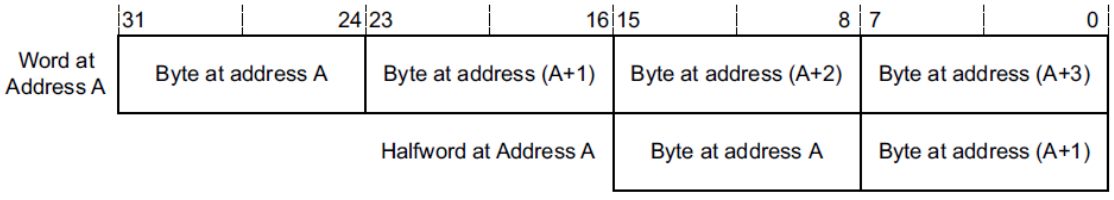


图 2.1.2-2 ARM Cortex-R52/R52+大端格式

ARM Cortex-R52/R52+的字节序列可以根据复位时的控制输入选择大小端模式，默认是小端模式。

瑞萨 RH850 G3MH/G3KH 采用小端格式，数据存储方式与 ARM Cortex-R52/R52+的小端格式类似。

在数据对齐方面，ARM Cortex-R52/R52+最常见的对齐要求是 4 字节对齐和 8 字节对齐。这意味着一个 4 字节的数据，它的存储地址必须是 4 的倍数；对于 8 字节的数据，它的存储地址必须是 8 的倍数。ARM Cortex-R52/R52+支持使能 4 字节对齐检测机制，可通过配置系统寄存器 SCTLR.A 字段实现。

瑞萨 RH850 G3MH/G3KH 同样支持 4 字节对齐和 8 字节对齐。而瑞萨 RH850 G3MH/G3KH 支持 4 字节和 8 字节对齐检测机制，可通过配置系统寄存器 MCTL.MA 字段实现。

移植提示：需要注意数据的存储方式。在移植时，如果数据的存储方式发生了变化，可能需要进行字节序的转换。

2.1.3 Operating modes

ARM Cortex-R52/R52+可以在多种模式下执行。每种模式与一个异常级别相关联。异常会导致处理器切换到对应的模式。User 模式关联异常级别 EL0。System, FIQ, IRQ, Supervisor, Abort 和 Undefined 模式关联异常级别 EL1。虚拟机模式关联异常级别 EL2。在启动后，默认进入 EL2 模式。

ARM Cortex-R52/R52+可通过 SVC 指令将模式由 User 切换至 EL1，通过 HVC 指令切换至 EL2。

RH850 G3MH/G3KH 具有特权模式（SV）和用户模式（UM）操作模式，由 PSW 寄存器中的 UM 状态位进行指示。在特权模式下，程序可访问所有硬件寄存器，具有最高硬件访问权限。在启动后，默认进入特权模式。在用户模式下，仅可以访问指定的寄存器，并且不能执行权限敏感指令，否则会引发异常。

RH850 G3MH/G3KH 可通过 SYSCALL 指令完成模式从 UM 至 SV 的切换。

ARM Cortex-R52/R52+操作模式常见的使用模型如下：

- EL0：应用程序
- EL1：操作系统内核态
- EL2：虚拟机（Hypervisor）

RH850 G3MH/G3KH 操作模式常见的使用模型如下：

- UM：应用程序
- SV：操作系统内核态

移植提示：RH850 G3MH/G3KH 的 UM 模式类似于 ARM Cortex-R52/R52+的 EL0。在该模式下运行的程序受限访问硬件寄存器等资源。RH850 G3MH/G3KH 的 SV 模式类似于 ARM Cortex-R52/R52+的 EL1 模式，程序可访问所有寄存器。

2.2 Instruction set

ARM Cortex-R52/R52+符合 Armv8-R AArch32 架构，有两种指令集状态：

- A32：执行 32 位、字对齐的 A32 指令
- T32：执行 16 位和 32 位、半字对齐的 T32 指令

瑞萨 RH850 G3MH/G3KH 指令集以 16 位和 32 位格式表示，并由少数指令可配置为 48 位和 64 位指令。RH850 G3MH/G3KH 进一步将指令集划分为基本指令集、协处理器指令集和保留指令集。协处理器指令集可由具体的处理器实现进行定义，保留指令集主要为将来的功能扩展而做的预留。

移植提示 1：ARM Cortex-R52/R52+和瑞萨 RH850 G3MH/G3KH 的指令集差距较大，汇编代码不兼容。如果程序中使用了汇编代码，则在迁移过程中需要重写相应的汇编代码。

移植提示 2：汇编代码的格式与所使用的编译器有关。在迁移过程中，请参考对应的编译器手册。下图给出了 ARM Cortex-R52/R52+基于 ARM 编译器的汇编代码格式示例。

```
Loop        MUL R5, R5, R1
            SUBS R1, R1, #1
            BNE Loop
```

图 2.2-1 ARM Cortex-R52/R52+基于 ARM 编译器的汇编代码格式示例

2.3 General Purpose Registers

ARM Cortex-R52/R52+基于 Armv8-R 架构。Armv8-R 架构的通用寄存器如下图所示：

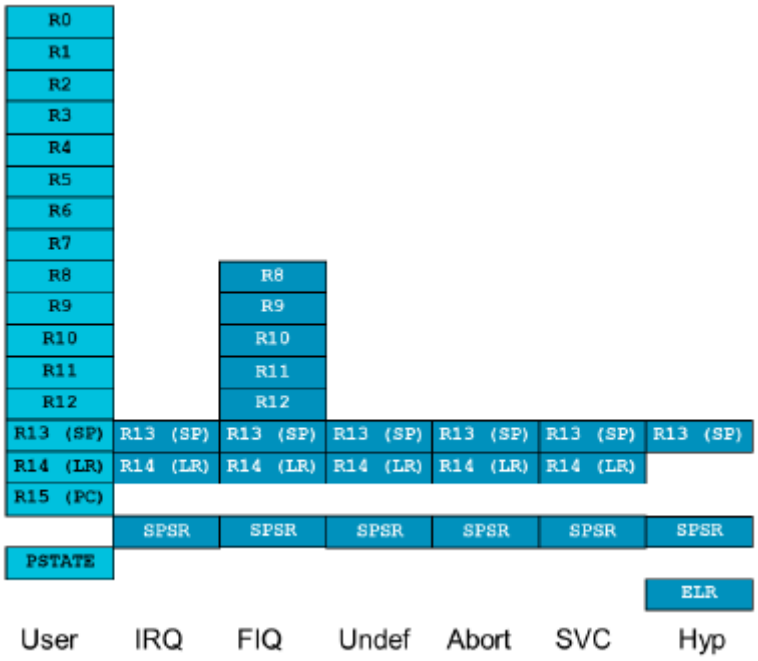


图 2.3-1 Armv8-R 架构通用寄存器

在 Armv8-R 架构引入了多种模式，包括 User、IRQ、FIQ、Undef、Abort、SVC、Hyp。寄存器 R0-R7 为各个模式所共享的通用寄存器。除 User 模式以外，其他模式均具备独立的 R13（SP）和 R14（LR）寄存器。其中，R13 存储了当前运行的栈地址（SP），R14 存储了当前的返回地址（LR）。FIQ 具备独立的 R8-R12 寄存器。

ARM Cortex-R52/R52+运行时，如果当前的运行模式发生了切换，SP 和 LR 将使用对应模式下的 R13 和 R14 作为当前的实际运行值。

瑞萨 RH850 G3MH/G3KH 具备 33 个通用寄存器。其中，R3 为当前运行的堆栈地址（SP），R31 为当前运行的返回地址（LR）。除 R0-R31 之外，瑞萨 RH850 G3MH/G3KH 提供了 PC 寄存器指示当前的指令地址。瑞萨 RH850 G3MH/G3KH 的通用寄存器如下图所示：

Program Register	Name	Function	Description
General-purpose registers	r0	Zero register	Always retains 0
	r1	Assembler reserved register	Used as working register for generating addresses
	r2	Register for address and data variables (used when the real-time OS used does not use this register)	
	r3	Stack pointer (SP)	Used for generating a stack frame when a function is called
	r4	Global pointer (GP)	Used for accessing a global variable in the data area
	r5	Text pointer (TP)	Used as a register that indicates the start of the text area (area where program code is placed)
	r6 to r29	Register for addresses and data variables	
	r30	Element pointer (EP)	Used as a base pointer for generating addresses when accessing memory
	r31	Link pointer (LP)	Used when the compiler calls a function
Program counter	PC	Retains instruction addresses during execution of programs	

图 2.3-2 瑞萨 RH850 G3MH/G3KH 通用寄存器

移植提示：栈用于在函数调用时生成一个栈帧，以保存函数调用的上下文和返回地址。栈还用于中断和异常中的上下文保存和恢复。瑞萨 RH850 G3MH/G3KH 将 R3 寄存器作为栈指针（SP）。Cortex-R52/R52+将 R13 寄存器作为栈指针（SP）。栈增长方向取决于特定的处理器实现。在迁移期间，需要注意栈指针的相关操作，如入栈、出栈等。

2.4.1 中断控制器

ARM Cortex-R52/R52+的中断控制器基于 ARM GIC-V34 架构。ARM Cortex-R52/R52+支持以下中断类型:

表 2.4.1-1 ARM Cortex-R52/R52+支持的中断类型

中断类型	含义	支持数量
Private Peripheral Interrupts (PPIs)	每个处理器核心的私有外设中断	16
Shared Peripheral Interrupts (SPIs)	共享外设中断。由外设产生，可通过软件配置路由到指定的处理器核心	960
Software Generated Interrupts (SGIs)	软中断。通过软件写入 SGI generation system register	16

ARM Cortex-R52/R52+具有 1 个处理器级别的 GIC Distributor，以及多个核心级的 GIC Redistributor（GIC CPU Interface Per Core）。GIC 逻辑框图入下图所示:

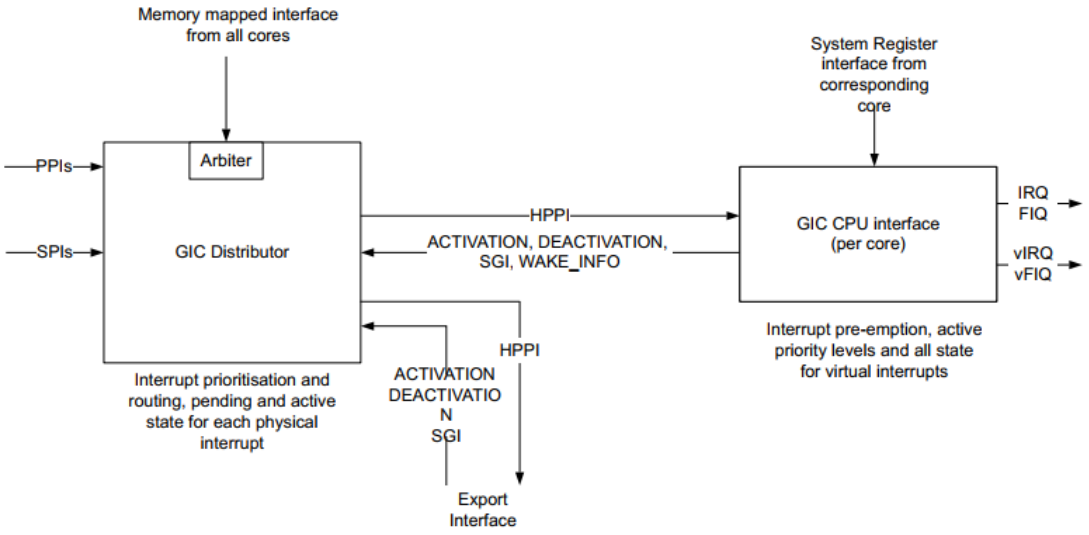


图 2.4.1-1 GIC 逻辑框图

ARM Cortex-R52/R52+的中断控制器支持分组（Group），例如：Group 0 用于接收 FIQ 中断，而 Group 1 用于接收 IRQ 中断。中断控制器寄存器通过内存进行映射，不同处理器的物理基地址由 CFGPERIPHBASE 寄存器指定。

ARM Cortex-R52/R52+具有两级中断向量表（Vector Table），EL1 模式和 EL2 模式分别有对应的中断向量表。中断向量表的结构如下表所示:

表 2.4.1-2 EL1 模式和 EL2 模式的中断向量表

地址偏移	EL1 Vector Table	EL2 Vector Table
0x00	Reset	Reset
0x04	Undefined Instruction	Undefined Instruction (From Hypervisor Mode)
0x08	Software Interrupt	HVC (from Hypervisor Mode)
0x0C	Prefetch Abort	Prefetch Abort (from Hypervisor Mode)
0x10	Data Abort	Data Abort (from Hypervisor Mode)
0x14	Reserved	Hypervisor Trap/Hypervisor mode entry
0x18	IRQ	IRQ
0x1C	FIQ	FIQ

瑞萨 RH850 G3MH/G3KH 不支持虚拟化，只具备一级中断向量表。中断控制器具有 1 个寄存器用于为每个中断源设置优先级。中断控制器根据寄存器配置的优先级，将中断请求按优先级进行处理，并将高优先级中断反馈给处理器核心。

瑞萨 RH850 G3MH/G3KH 提供了直接向量法和表引用法两种中断向量安装机制：

- **直接向量法**

在这种方法中，使用 RBASE 和 EBASE 相加的地址作为中断的响应地址。RBASE 是中断向量表的地址，EBASE 是每种异常类型的偏移量。在这种方法中，每个中断优先级都对应一个用户中断异常处理程序，并且具有相同优先级的中断会跳转到相同的中断处理程序。直接向量法示例如下图所示：

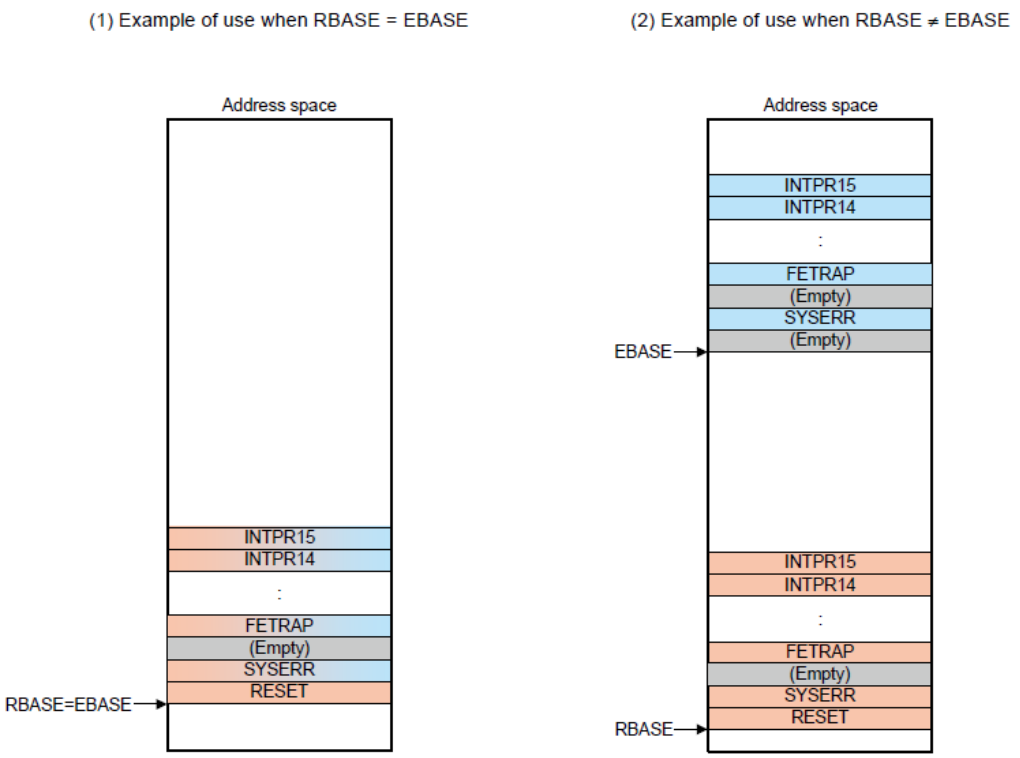


图 2.4.1-2 瑞萨 RH850 G3MH/G3KH 直接向量法

- **表引用法**

在这种方法中，为每个中断提供了一个单独的中断处理程序，示例如下图所示：

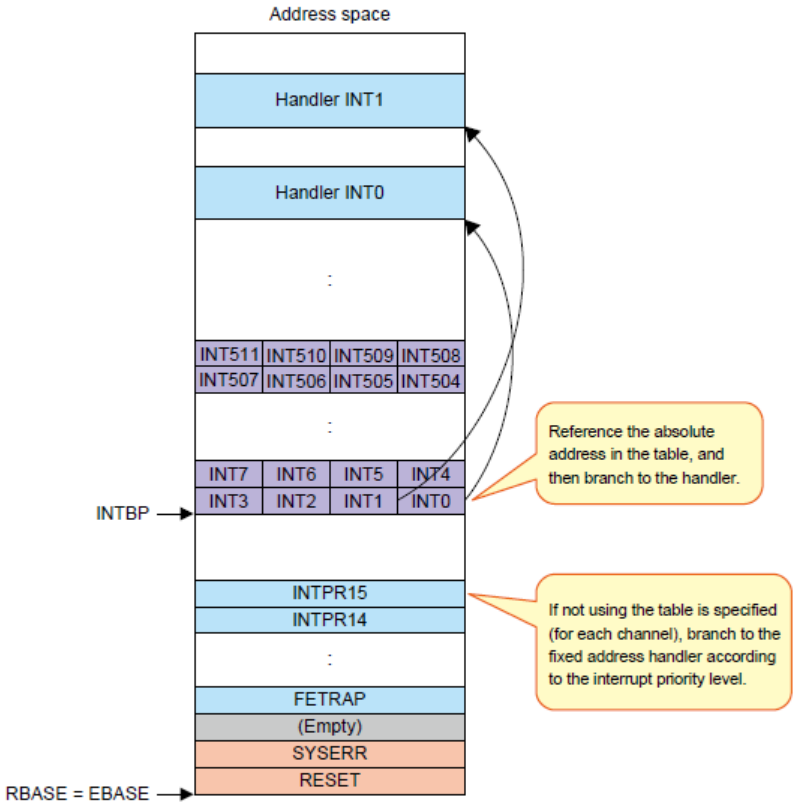


图 2.4.1-3 瑞萨 RH850 G3MH/G3KH 表引用法

移植提示：ARM Cortex-R52/R52+和瑞萨 RH850 G3MH/G3KH 的中断系统原理不同。在迁移时，需要重新初始化中断控制器，设置中断向量表。ARM Cortex-R52/R52+启动后默认使用 EL2 模式对应的中断向量表。可在 EL2 模式中断向量表 Reset Handler 处理完成后，配置 EL1 模式中断向量表，并跳转至 EL1 模式中断向量表 Reset Handler 执行。参考过程示例如下：

- EL2 模式中断向量表示例：

```
EL2_Vectors:
        LDR PC, EL2_Reset_Addr
        LDR PC, EL2_Undefined_Addr
        LDR PC, EL2_HVC_Addr
        LDR PC, EL2_Prefetch_Addr
        LDR PC, EL2_Abort_Addr
        LDR PC, EL2_HypModeEntry_Addr
        LDR PC, EL2_IRQ_Addr
        LDR PC, EL2_FIQ_Addr
```

图 2.4.1-4 ARM Cortex-R52/R52+ EL2 模式中断向量表

- EL2 模式 Reset Handler 示例：

```
EL2_Reset_Handler:
        LDR r0, =EL2_Vectors
        MCR p15, 4, r0, c12, c0, 0      //  Write to HVBAR

        // Init HSCTLR
        LDR r0, =0x30C5180C             // See TRM for decodi
        MCR p15, 4, r0, c1, c0, 0       // Write to HSCTLR

        // Enable EL1 access to all IMP DEF registers
        LDR r0, =0x7F81
        MCR p15, 4, r0, c1, c0, 1       // Write to HACTLR

        // Change EL1 exception base address
        LDR r0, =EL1_Vectors
        MCR p15, 0, r0, c12, c0, 0      // Write to VBAR

        // Go to SVC mode
        MRS r0, cpsr
        MOV r1, #Mode_SVC
        BFI r0, r1, #0, #5
#ifdef __THUMB__
        ORR r0, r0, #(0x1 << 5)         // Set T bit
#endif
        MSR spsr_cxsf, r0
        LDR r0, =EL1_Reset_Handler
        MSR elr_hyp, r0
        DSB
        ISB
        ERET
```

图 2.4.1-5 ARM Cortex-R52/R52+ EL2 模式 Reset Handler 示例

将 EL1 模式中断向量表基地址（EL1_Vectors）设置到 VBAR 寄存器，并通过 ERET 指令跳转至 EL1 模式中断向量表 Reset Handler，即可实现向 EL1 模式中断向量表的切换。

- EL1 模式中断向量表示例：

```
EL1_Vectors:
        LDR PC, EL1_Reset_Addr
        LDR PC, EL1_Undefined_Addr
        LDR PC, EL1_SVC_Addr
        LDR PC, EL1_Prefetch_Addr
        LDR PC, EL1_Abort_Addr
        LDR PC, EL1_Reserved
        LDR PC, EL1_IRQ_Addr
        LDR PC, EL1_FIQ_Addr
```

图 2.4.1-6 ARM Cortex-R52/R52+ EL1 模式中断向量表

2.4.2 异常控制器

ARM Cortex-R52/R52+异常模型定义了 EL0-EL2 三个异常级别，其中：

EL0：最低的软件执行特权等级，即用户态；

EL1：增强的异常级别，即特权态；

EL2：提供虚拟化支持。

ARM Cortex-R52/R52+在捕获到特定事件后，进入异常处理过程，并通过异常返回指令将程序执行过程返回到异常发生时的状态。

瑞萨 RH850 G3MH/G3KH 支持两种异常级别：EI 级异常和 FE 级异常。FE 级异常用于对系统具有高度紧急性的异常或可能发生的内存管理功能的异常。EI 级异常用于常规的用户处理，其中包含了外设中断。RH850 G3MH/G3KH 将中断也视为一种异常。

2.5.1 内存视图

ARM Cortex R52/R52+ 所使用的 Armv8-R AArch32 架构定义了 PMSAv8 内存模型。内存访问权限和属性由内存保护单元（MPU）确定。在 ARM Cortex R52/R52+中，物理地址始终与虚拟地址相同。Armv8-R 的缺省内存视图如下图所示：

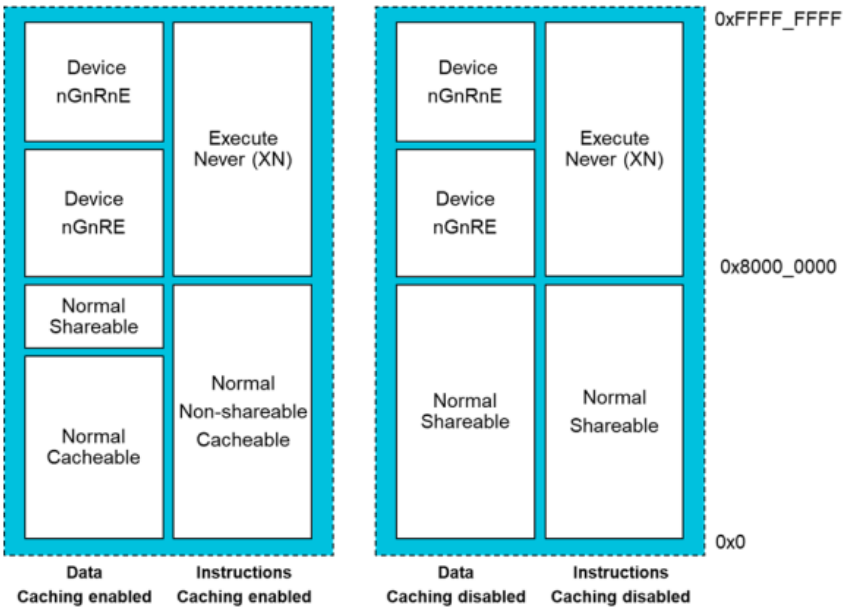


图 2.5.1-1 ARM Cortex-R52/R52+ 缺省内存视图

Armv8-R 包含以下内存类型：

- Normal memory

Normal memory 是常规的内存区域，适用于不同类型的存储器，如 ROM、RAM、Flash 和 SDRAM。该区域允许程序读取和写入。

- Device memory

Device memory 适用于外设和 I/O 访问。该区域不支持高速缓存（Cache），但仍支持通过缓冲区的方式读写数据。

- Strongly ordered

Strongly-ordered memory 对指令执行的顺序有严格的要求，保证指令按顺序执行。

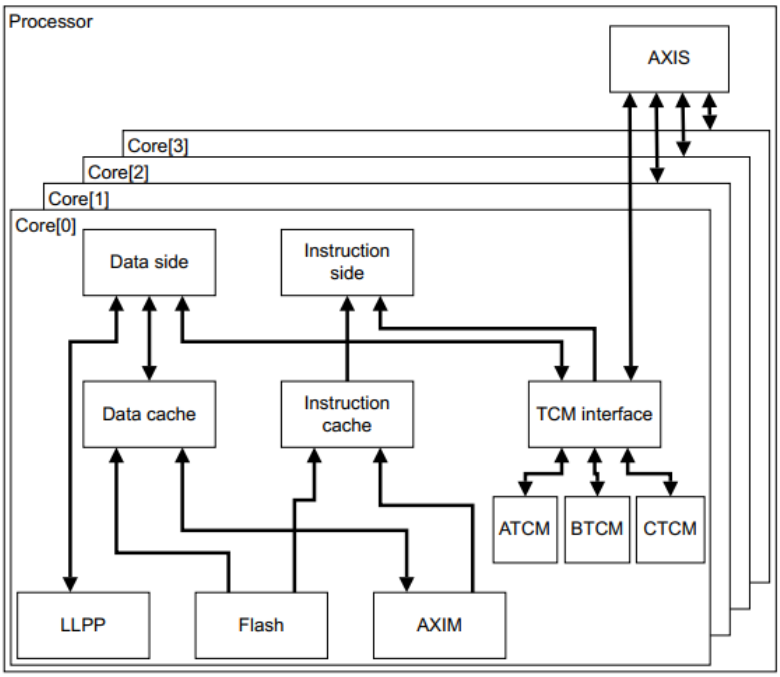


图 2.5.1-2 ARM Cortex-R52/R52+ 内存系统框图

ARM Cortex R52/R52+ 具备多级内存系统。每个核具备独立的数据缓存（Data cache）和指令缓存（Instruction cache）。每个核具备独立的 TCM（Tightly Coupled Memory），可用于软件实现快速数据读写和指令执行。AXIM interface 是访问外部存储器的主要界面。Flash interface 用于访问外部只读存储器，如 Flash。LLPP interface 用于访问外设和特定的外部存储器。

瑞萨 RH850 G3MH/G3KH 支持 4 GB 内存地址空间，无论是指令寻址（指令访问）还是操作数寻址（数据访问），都可以直接映射到整个地址空间中。

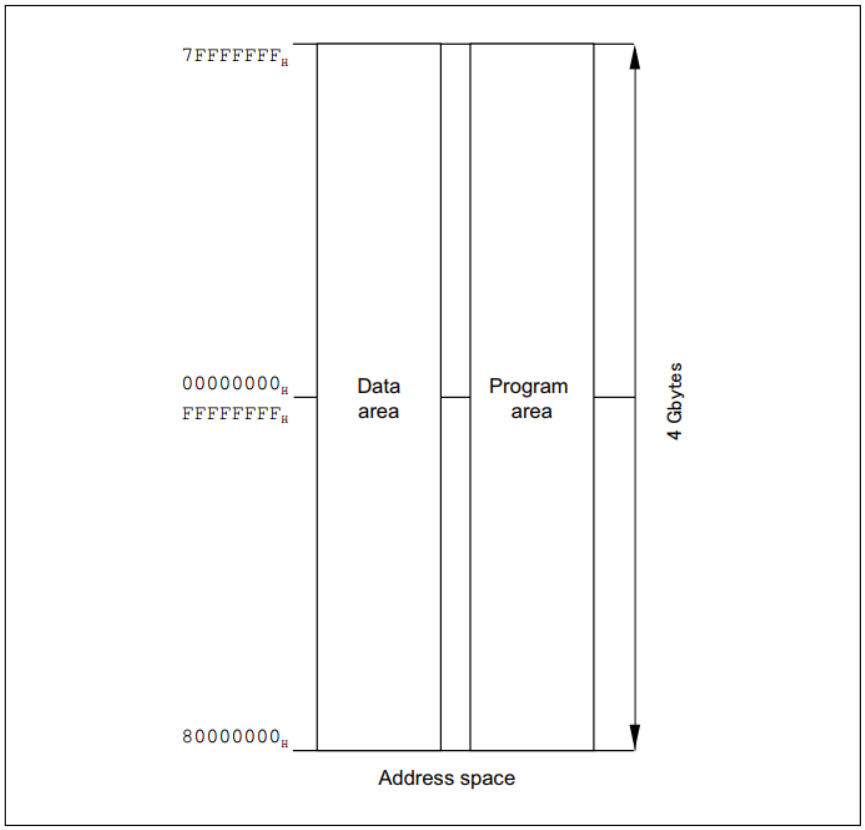


图 2.5.1-3 瑞萨 RH850 G3MH/G3KH 内存视图

2.5.2 Memory Protection Unit

ARM Cortex R52/R52+提供了 2 个可编程的 MPU。EL1 模式和 EL2 模式分别对应于各自的 MPU。每个 MPU 均能覆盖 4G 地址空间。每个内存保护区域（Memory Region）包括以下配置元素：起始地址、结束地址、访问权限和内存属性。

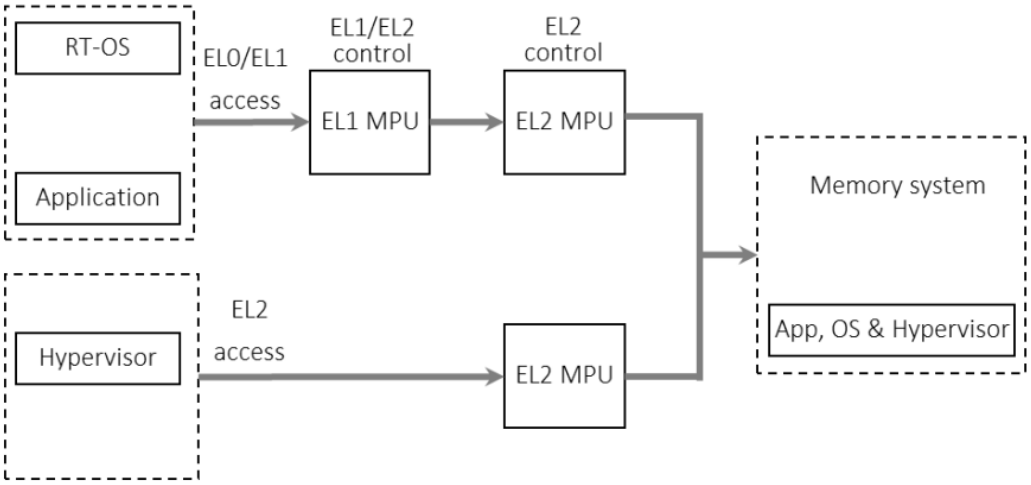


图 2.5.2-1 ARM Cortex R52/R52+ MPU 框图

ARM Cortex R52/R52+支持虚拟化。应用程序和 RTOS 访问 EL1 MPU，实现应用之间，以及应用和 RTOS 之间的隔离。虚拟机控制 EL2 MPU，实现虚拟机之前的隔离。ARM Cortex R52/R52+ MPU 使用 2bit 来控制访问权限，如下表所示：

表 2.5.2-1 ARM Cortex R52/R52+ MPU 的访问权限控制

权限 Bit 位	Unprivileged (EL0)	Privileged (EL1)
00	No access	Read/write
01	Read/write	Read/write
10	No access	Read-only
11	Read-only	Read-only

瑞萨 RH850 G3MH/G3KH 不支持虚拟化，只具备 1 个 MPU。瑞萨 RH850 G3MH/G3KH 的 MPU 支持分别在特权模式（SV）和用户模式（UM）下设置保护区域和相应的权限。 瑞萨 RH850 G3MH/G3KH 使用 6bit 来控制访问权限，如下图所示：

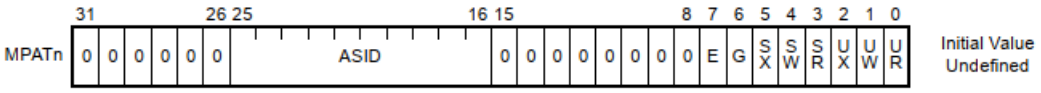


图 2.5.2-2 瑞萨 RH850 G3MH/G3KH MPU 权限位

其中，SX、SW 和 SR 对应于特权模式（SV）下的访问权限，UX、UW、UR 对应于用户模式（UM）下的访问权限。

2.6 Debug

ARM Cortex R52/R52+支持通过 JTAG 或者 SWD 协议进行调试。可通过调试设备，如 ARM DSTREAM 连接至目标设备。

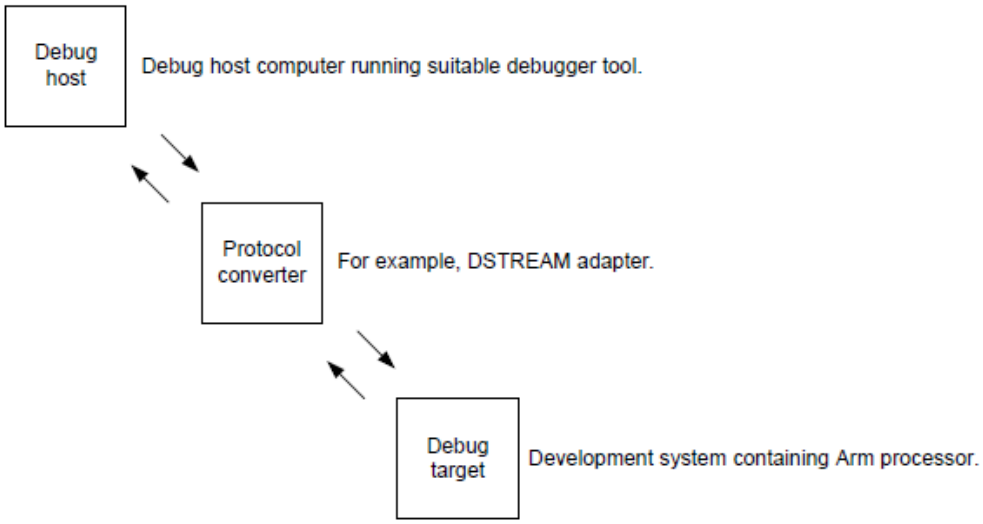


图 2.6-1 ARM Cortex R52/R52+调试模型

瑞萨 RH850 G3MH/G3KH 支持 JTAG 协议，调试模型与 ARM Cortex R52/R52+类似。可通过调试设备，如 Renesas E1 simulator 连接至目标设备。

第 3 章 软件开发移植

此章节介绍移植的软件默认是用高级语言开发的，例如 C 语言，C 语言具有编译简易、跨平台等特性，在嵌入式中应用广泛。在实际的移植过程中，会有依赖于平台架构的汇编代码，特别是存在于启动代码、异常处理程序中，但每个平台架构都有各自的例程供用户参考，这也为用户提供了便利。

本文主要从以下几个方面来介绍瑞萨 RH850 G3MH/G3KH 的软件移植到 ARM Cortex R52/R52+中：

- 开发工具链
- 芯片启动过程
- 异常、中断处理
- 外设访问

3.1 开发工具链

嵌入式软件开发环境和运行时环境位于不同平台之上。通常，开发环境部署在 Windows/Linux 计算机上，而运行时环境部署在目标芯片上。通过交叉编译环境实现运行时环境程序的编译和构建。嵌入式开发环境工具链通常包括编译器、汇编器、链接器、调试器等。随着工具链的发展，编辑软件、编译软件、汇编软件、链接软件、调试软件和功能库都已整合到一个环境中，称为集成开发环境（Integrated Development Environment，IDE）。

推荐的 ARM 平台工具链包括：Arm Development Studio、Green Hills MULTI Integrated Development Environment。

将瑞萨 RH850 G3MH/G3KH 的软件移植到 ARM Cortex R52/R52+，需要考虑不同编译环境的差异，主要从以下几个方面进行分析：

- 汇编代码

为了提升程序的运行效率，程序的启动代码、中断和异常处理程序会用汇编代码编写。在移植时，需要考虑不同平台汇编指令的区别，可以参考章节 2.2 Instruction set。集成开发环境通常会给用户提供不同平台的例程，可以将移植平台的启动代码、异常处理等程序替换成目标平台的对应例程。

- 数据类型

不同编译器支持不同平台的数据类型不同，具体参考编译器的类型定义，此部分可以参考章节 2.1.1 Data Type。

- 编译、链接选项

在移植过程中，需要考虑不同编译器选项，比如编译器支持的 C 语言的不同规范(C90/C99 等)、支持不同的 ANSI 标准以及编译或链接的优化选项等。不同的编译、链接选项会影响生成的可执行文件运行效果。

- 链接文件

在编译器的链接过程中，需要根据链接文件将程序分配到不同的地址空间，比如分配到 SRAM 区、FLASH 区、堆栈区等。不同的编译器的链接文件格式不同。集成开发环境通常会给用户提供不同平台的链接文件模板。

3.2 芯片启动

芯片上电时，在执行用户开发的应用程序之前，需要对芯片内部寄存器等进行初始化，建立必要的运行环境。在芯片初始化完成之后，才能执行应用程序。

ARM Cortex-R52/R52+的启动过程从复位向量处开始执行，通常先进入 EL2 模式，并执行 EL2 模式中断向量表的 Reset Handler。若不使用虚拟机，可在 EL2 模式的 Reset Handler 中，设置 EL1 模式的中断向量表，并跳转至 EL1 模式的 Reset Handler，可参考章节 2.4.1 中断控制器。

在 Reset Handler 中，通常执行以下初始化过程：

- 配置缓存机制以启用或禁用数据缓存和程序缓存功能
- 初始化 CPU 内核寄存器
- 初始化堆栈指针
- 初始化诸如 MPU 等模块
- 初始化.bss 和.data 段中的全局变量
- 初始化时钟等外设模块
- 进入用户应用程序的入口点，如 main 函数

在进行移植时，需要考虑启动过程中的差异。工具链为特定处理器提供了启动代码样例。该样例可以直接用于项目中。一些特殊的模块需要进行配置，例如 MPU 内存保护单元的配置，需要根据链接文件配置堆栈寄存器的值，需要根据链接文件中的分区进行全局变量的初始化等。

3.3 异常、中断处理

ARM Cortex-R52/R52+的中断和异常可参考章节 2.4 Exceptions and interrupts。

在 ARM Cortex-R52/R52+中，中断和异常共用一个中断向量表，下面主要介绍异常的处理流程：

1、处理器状态会自动保存到 SPSR 寄存器中，并更新当前处理器状态。如果在 EL1 模式，则修改链接寄存器（LR）以供后续返回中断使用。

2、从中断向量表的相应条目执行。该条目是一条跳转指令，跳转至 top-level handler。

3、保存上下文，然后调用 second-level exception handler。

4、Second-level exception handler 通常在 C 语言环境中执行。执行结束后，返回到 top-level handler。

5、Top-level handler 恢复上下文，然后返回到链接寄存器（LR）保存的地址。

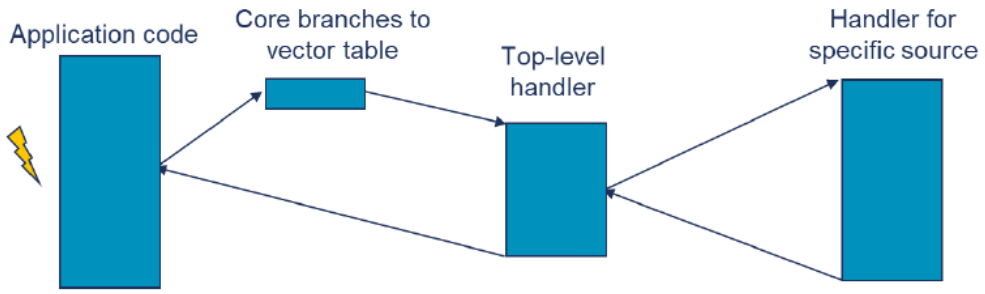


图 3.3-1 ARM Cortex R52/R52+异常处理过程

3.4 外设访问

ARM Cortex-R52/R52+的外设取决于不同芯片厂商的实现方式。通常，芯片厂商会为用户提供外设驱动。在移植过程中，可以基于芯片厂商提供的外设驱动进行开发。

第 4 章 附录

4.1 参考资料

ID	参考资料	版本	日期
1	Arm® Cortex®-R52 Processor Technical Reference Manual	Revision: r1p4	20 July 2021
2	Arm® Cortex®-R52+ Processor Technical Reference Manual	Revision: r0p1	25 August 2022
3	Arm® Architecture Reference Manual Supplement: Armv8, for the Armv8-R AArch32 architecture profile	Updated EAC release	06 November 2020
4	RH850 G3MH User's Manual: Software	Rev.1.30	Dec 2016
5	RH850 G3KH User's Manual: Software	Rev.1.20	Dec 2016

4.2 术语和缩写词

术语/缩写词	描述
CPU	Central Processing Unit
MCU	Microcontroller Uni
MMU	Memory Management Unit
IDE	Integrated Development Environment