STM32F429 使用外扩 SDRAM 运行程序的方法

一. SDRAM 运行程序基本原理

STM32 的高端产品 429/439 添加了新的外设，SDRAM 控制器（FMC 总线）。不少客户都使用外扩的 SDRAM 作为变量的存储区，也有可能作为 C stack 和 heap 的存放区，因此，需要在 IAR 链接前（数据的拷贝）完成 SDRAM 控制器的初始化工作，ST 已经提供相关代码供客户参考（初始化程序在 system_stm32f4.c 中）。

若对函数使用 IAR 链接器关键字 （例：__ramfunc void func1(void) ），IAR 在链接的时候会将程序放在 SDRAM 的区域，并将函数的入口地址传给调用者。

二. 客户问题描述及原因

客户反映使用外扩 SDRAM 运行程序（使用链接器将 code 存放在 SDRAM 中，与编译器无关，采用 GCC 或者 IAR 都有这个问题）出错，Hard Fault 发生。

单步运行，使能 Usage, Memory, Bus Fault，发现 Hard Fault 是由 Memory Fault（bit 0 IACCVIOL）引起的。

参考 ARM 关于 Cortex-M4 内核 IACCVIOL 位的说明

Instruction access violation flag:

0 = no instruction access violation fault

1 = the processor attempted an instruction fetch from a location that does not permit execution. The PC value stacked for the exception return points to the faulting instruction. The processor has not written a fault address to the MMAR. This fault condition occurs on any access to an XN (eXecute Never) region, **even when the MPU is disabled or not present**. Potential reasons:

a) Branch to regions that are not defined in the MPU or defined as non-executable.

b) Invalid return due to corrupted stack content.

c) Incorrect entry in the exception vector table

在 M4 内核中，0x2000 0000 以上的地址是通过 System Bus 来访问的，system bus 是可以对数据和指令读取的，但是内核的默认设置（即 MPU 处于 Disabled 的状况下），部分地址禁止执行指令，见下图（STM32F429 的 SDRAM 取值范围设置在 0xC000 0000 以上），因此会发生该错误。

Address	Name	Device Type	XN?	Cache	Description
0x00000000-0x1FFFFFFF	Code	Normal	-	WT	Typically ROM or flash memory. Memory required from address 0x0 to support the vector table for system boot code on reset.
0x20000000-0x3FFFFFFF	SRAM	Normal	-	WBWA	SRAM region typically used for on-chip RAM.
0x40000000-0x5FFFFFFF	Peripheral	Device	XN	-	On-chip peripheral address space.
0x60000000-0x7FFFFFFF	RAM	Normal	-	WBWA	Memory with write-back, write allocate cache attribute for L2/L3 cache support.
0x80000000-0x9FFFFFFF	RAM	Normal	-	WT	Memory with write-through cache attribute.
0xA0000000-0xBFFFFFFF	Device	Device, shareable	XN	-	Shared device space.
0xC0000000-0xDFFFFFFF	Device	Device, non-shareable	XN	-	Non-shared device space.
0xE0000000-0xFFFFFFFF	System	See *Description*	XN	-	System segment for the PPB and vendor system peripherals, see Table B3-2 on page B3-706.

因此有两种解决方案：

1. 更改 MPU (Memory Protect Unit) 设置（通用性上来讲此方法更好，本文将介绍这种实现方式）。
2. Memory Remap （将 SDRAM 调整到 I-Code 总线上）。

三. MPU 设置方法

关于 MPU 的讲解，强烈建议大家阅读《ARM Cortex-M3 权威指南》或者 ARM Cortex-M4 的官方文档。在与本文对应的例程中已给出了基本的实现方法（请参考 stm32_mpu.c）。同时，本文也给出了如何将数据存放在 SRAM 及 CCMRAM 的方法（请参考 IAR_DevelopmentGuide）。

四.结语

MPU 的主要作用是实施存储器的保护，它能够在系统或程序出现异常而非正常地访问不应该访问的存储空间时，通过触发异常中断而达到提高系统可靠性的目的。

所谓非正常地访问不应该访问的存储空间，最常见的现象是存储器访问越界，更具体的表现可以是数组溢出、堆栈溢出、动态存储器分配失败等情况；另一种常见的现象是程序跑飞了。

事实上，MPU 可以 RTOS 中有着更广泛的应用（参考 FreeRTOS 相关资料）。