AN12437　　i.MX RT 系列性能优化

Rev. 1 — Febuary 2020

URL：https://www.nxp.com/docs/en/application-note/AN12437.pdf

目录

1、**引言**

i.MX RT 系列采用了 Arm® Cortex® -M7 核心与 32K/32K L1 I/D-Cache，其运行速度高达 600 MHz 以提供高 CPU 性能和最佳实时响应。

•i.MX RT1050 处理器具有 512 KB 片上 RAM，可以灵活配置配置为 TCM 或通用片上 RAM。

•i.MX RT1060 处理器具有额外的 512 KB OCRAM，片上总共 1 MB 内存。

i.MX RT 系列提供各种内存接口，包括 SDRAM，RAW NAND 闪存，NOR 闪存，SD / eMMC 和 FlexSPI。这些丰富的功能帮助 i.MX RT 系列实现灵活的应用和高性能。系统性能取决于运行的系统和存储设备。本文档旨在介绍如何优化在不同存储设备上运行的系统性能。

2、**概述**

因为集成了 Cortex-M7 高性能内核，i.MX RT 可以：

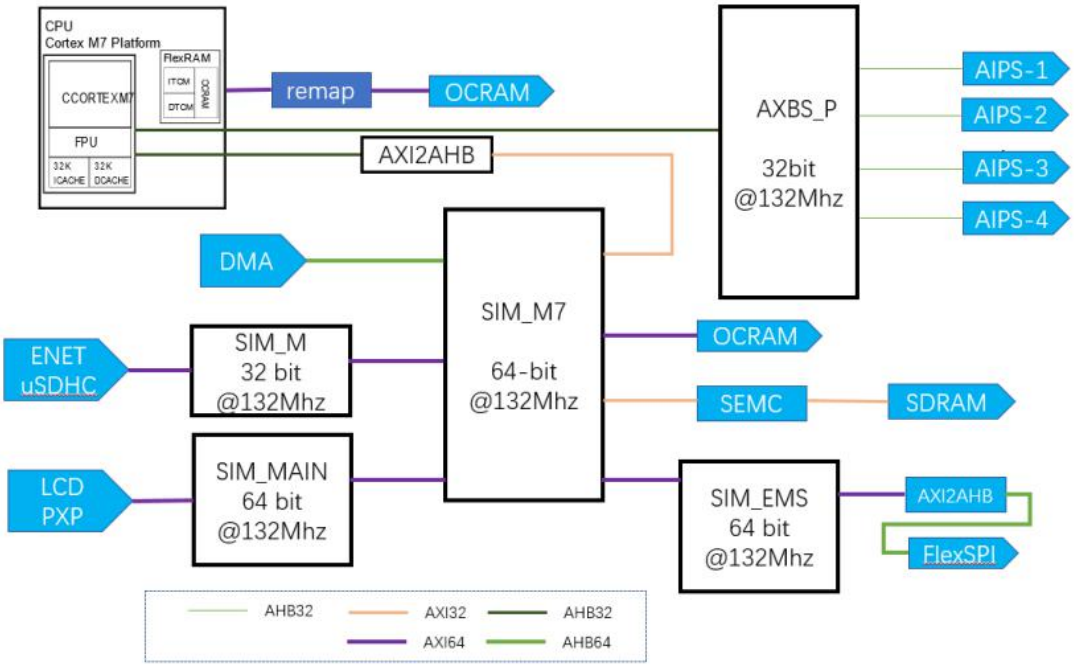
•最高运行 600 MHz。

•通过 32 K DCACHE 和 ICACHE 增强性能。

•基于应用，灵活且可配置的 FlexRAM 可分区为 DTCM / ITCM / OCRAM

有关如何配置 FlexRAM 的信息，请参考 AN12077。

i.MX RT 是不带闪存的。但是，它嵌入了高性能内部 SRAM，丰富的外设接口可以和很多存



储设备连接，例如 SDRAM，RAW NAND FLASH，NOR FLASH，SD / eMMC，Quad SPI Flash 和 Hyper flash。

根据工作模式，内存可以分为两种类型。

•XIP 存储器：直接执行代码

•非 XIP 存储器：不支持就地执行代码，而是将代码加载到可执行内存中。

下面列出了 i.MX RT 系列支持的可执行内存。

•ITCM / DTCM

•SDRAM

•OCRAM

•Hyper RAM

•Hyper / Octal NOR 闪存（支持 XIP）

•QSPI NOR 闪存（支持 XIP）

•并行 NOR 闪存（支持 XIP）

•并行 SRAM

基于总线体系结构和内存特性，不同的内存表现出不同的性能。图 1 显示

RT 系列的总线架构，以 i.MX RT1060 系统总线图为例。

图 1.i.MX RT1060 系统总线图

如图 1 所示，TCM 与 M7 内核紧密耦合，可运行在和内核相同的频率。OCRAM 和 SEMC 连接到 SIM_M7 结构，FlexSPI 连接到 SIM_EMS。相同的存储设备对于不同的主设备访问表现出不同的性能。例如，

•MCU 内核访问 TCM 表现出高性能。

•通过 DMA 访问时，OCRAM 显示出比 TCM 更高的性能，而通过 MCU 内核访问时，显示出更低的性能。原因是 OCRAM 和 DMA 位于同一总线结构中，访问期间的等待时间较短。

表 1 描述了总线结构摘要

表 1 总线结构摘要

Name	Bus width	Typical frequency	Comments
SIM_M7	64	132 MHz	All the fabric runs at the same clock frequency and it is always m:1 synchronous to M7 core clock. This table is based on the core frequency of 528 MHz.
SIM_MAIN	64		
SIM_EMS	64		
SIM_AXBS_P	32		
SIM_M	32		

表 2 描述了 i.MX RT 支持的每个存储设备的总线带宽

表 2 存储设备的总线带宽

Memory	Bus width (:bit)	Max speed (:MHz)	Comments
ITCM	64	600	It has two DTCM controllers with 32 bit, available to access odd or even address by different controller.
DTCM	2*32	600	
OCRAM	64	132	
SDRAM	16	166	
Hyper RAM	8 (DDR)	166	
Hyper flash	8 (DDR)	166	
QSPI flash	4	133	

总线带宽是影响存储设备性能的主要因素，但并非是全部因素。总线体系结构提供了一些增强功能以提高存储设备性能，例如 ICACHE / DCACHE。FlexSPI IP 支持额外的 1 KB RX AHB 预取缓冲区，并且可以将闪存数据预取到专用缓冲区，从而节省了读取访问延迟。但是，改善程度取决于应用程序。例如，当高速缓存命中率很高并且按顺序访问 QSPI 闪存时，它将得到更大的提升。系统性能与内存备和应用案例有关。在某些应用案例中，它可以获得类似的性能，如下所示表 7。但是，它在另一种情况下，在不同存储设备的运行差异很大。下面介绍差异的情况和如何改善。

3、**内存性能测试**

内存性能取决于内存特性，系统架构和其他方面，例如缓存，预取缓冲区和管道等等。

同样的内存在不同的主设备（CPU 内核，PXP，LCD，CSI，USB，eDMA 等等）下表现不同。例如，当 LCD 和 PXP 这两个主设备访问时，SDRAM 可以达到高吞吐量，因为这两个设备支持背对背访问。与其他主访问相比，它可以获得更好的性能，但在 CPU 内核某些访问时，它的性能下降很多。

以下性能讨论基于 CPU 内核的访问。

3.1 SDRAM 性能

i.MX RT 系列支持与 8/16 位 SDRAM 器件接口，并且最高可以运行 166 MHz。表 3 显示了测试结果通过读取/写入 4096 字节来进行传输，该字节通过系统时钟来测量 SDRAM 传输的持续时间。

表 3 SDRAM 性能

Items	Performance (:MB/s)		Comments
	DCache enabled	DCache disabled	
SDRAM read	111	25	SDRAM Working @ 166 MHz
SDRAM write	323	322	SDRAM Working @ 166 MHz

表 3 显示了 SDRAM 读写访问的良好性能。主要原因来自管道和 SEMC IP 高性能，缓存也提高了读取性能。

为了复现上面的测试，您可以从随附的软件包中获取测试代码。测试步骤如下。

•解压缩**性能测试**包，然后通过 AN12437SW\boards\evkbimxrt1050\demo_apps\performance_test\sdram_perforamnce_test\iar.打开 semc.eww t。请先安装 IAR 版本 8.40 或更高版本。

•生成调试子项目以生成 s-record 文件。宏 DCACHE_ENABLE 用于禁用或启用缓存。您根据测试要求修改它。

• 使用以下命令生成 sb 文件。

•通过基于 IMXRT1050-EVKB 板的 MFGTools 对闪存进行编程。

•代码在内部 ITCM 上运行。不要通过调试直接运行代码，这可能会影响性能。

将代码下载到闪存后，您可以运行并在串行终端中查看测试结果。

```
elftosb.exe -f imx -V -c imx-itcm-unsigned.bd -o ivt_flexspi_nor_normal.bin semc.srec
```

```
elftosb.exe -f kinetis -V -c program_flexspinor_image_hyperflash.bd -o boot_image.sb
```

```
ivt_flexspi_nor_normal_nopadding.bin
```

注意

最后的测试是针对超级闪存。配置闪存并使其以目标速度工作。有关详细信息，请参阅

FlexSPI 性能。

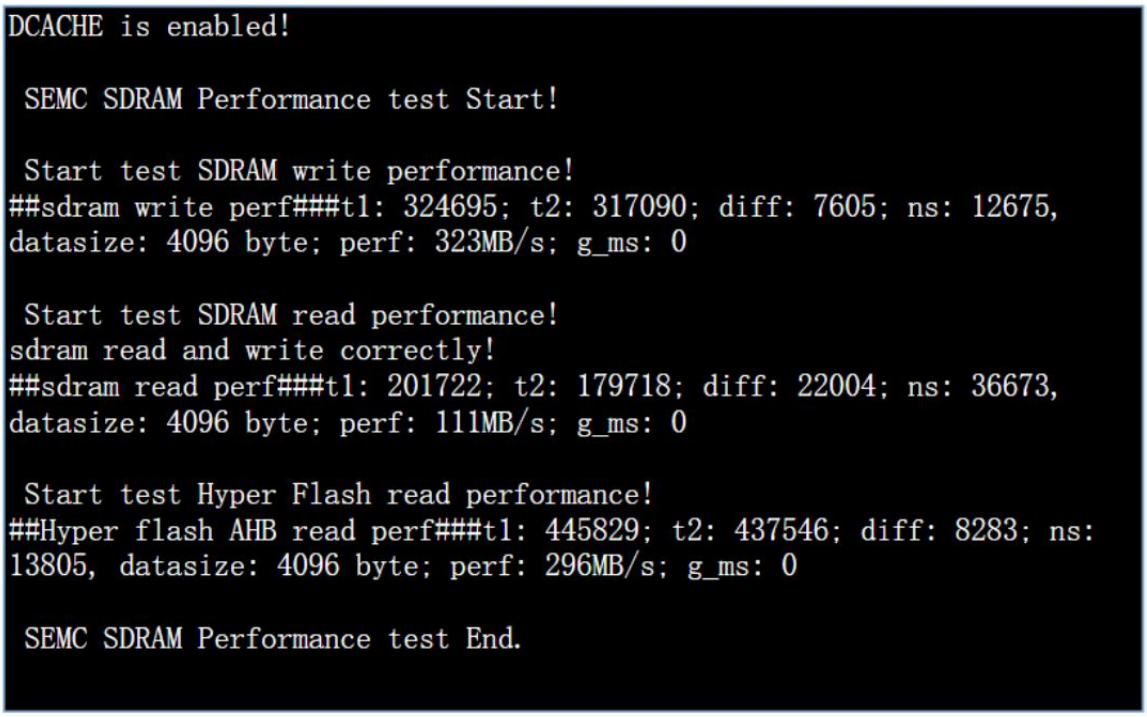


图 2 Sdram 性能测试

3.2 FlexSPI 性能

i.MX RT 支持 FlexSPI 接口。它提供灵活的配置来连接 QSPI 闪存，OCTAL 闪存，超级闪存和超级 RAM。它支持 AHB 和 IP 命令访问。AHB 访问有助于实现高性能，描述如下。

FlexSPI 支持 eXecute-In-the-Place (XIP)连接 NOR 闪存工作。连接到 FlexSPI 的 BEE 模块可实时解密固件。以下的 FlexSPI 的增强功能有助于提高性能。

•系统缓存（32 k DCACHE 和 32 K ICACHE）

•AHB 缓冲区，8 * 64 位 TX AHB 缓冲区和 128 * 64bit RX AHB 缓冲区

表 4 显示了性能评估，以 hyper / QSPI 闪存为例。

表 4 超级闪存性能

Items	Performance (:MB/s)				Comments
	DCache enabled, prefetch buffer enabled	DCache disabled, prefetch buffer enabled	DCache disabled, prefetch buffer disabled	DCache enabled, prefetch buffer disabled	
Hyper flash	296	70	15	92	Continuously reading 4 KB bytes @166 MHz DDR mode
QSPI flash	58	58	8	35	Continuously reading 4 KB bytes @133 MHz SDR mode

超级闪存比 QSPI 闪存具有更高的性能。它受益于总线带宽，工作速度和工作模式（DDR）。

通过启用缓存和预取缓冲区，可以使性能得到更大的提高。测试结果表明，无论禁用或者是启用预取缓冲区，它都能得到相似的性能。当预取缓冲区和高速缓存禁用时，性能下降约77%。

预取对 flexSPI 性能产生了重大影响。它可以为不同的主设备指定不同的缓冲区大小。那意味着某些主设备能具有专用的预取缓冲区，可以在某些应用程序中优化性能。例如，它可以为 eDMA 分配指定的缓冲区大小。如果需要从外部 QSPI 闪存频繁的传输数据到内部 SRAM，可以通过 eDAM 传输，其他主设备不会破坏用于 eDMA 的预取缓冲区内容。如果下次通过 DMA 访问同样的地址，则可以减少访问延迟。这样，它可以进一步提高性能。

FlexSPI 提供以下寄存器，以设置不同主设备的缓冲区大小。

•AHBRXBUF0CR0

•AHBRXBUF0CR1

•AHBRXBUF0CR2

•AHBRXBUF0CR3

用户可以修改这些寄存器并分配专用的缓冲区给某些主设备服务，如下显示在 表 5。

表 5 主设备 ID 号

Module	Master ID
Core platform	000b
eDMA	001b
DCP	010b
All others	011b

如所见 表 5，将单独的主设备 ID 号分配给内核，eDMA 和 DCP。其他主机共享一个 ID，例如 PXP，USB 等等。

图 3 显示了通用的预取方案。

启用预取缓冲区后，一旦从总线接收到请求，它将首先检查该请求是否与当前的 AHB 缓冲区地址范围匹配。如果是，它将直接返回数据。如果不是，它将读取新数据到 AHB 缓冲区。在将所需的数据传输到总线后，它将继续将紧接着的闪存数据预取到 AHB 缓冲区，直到缓冲区存满。

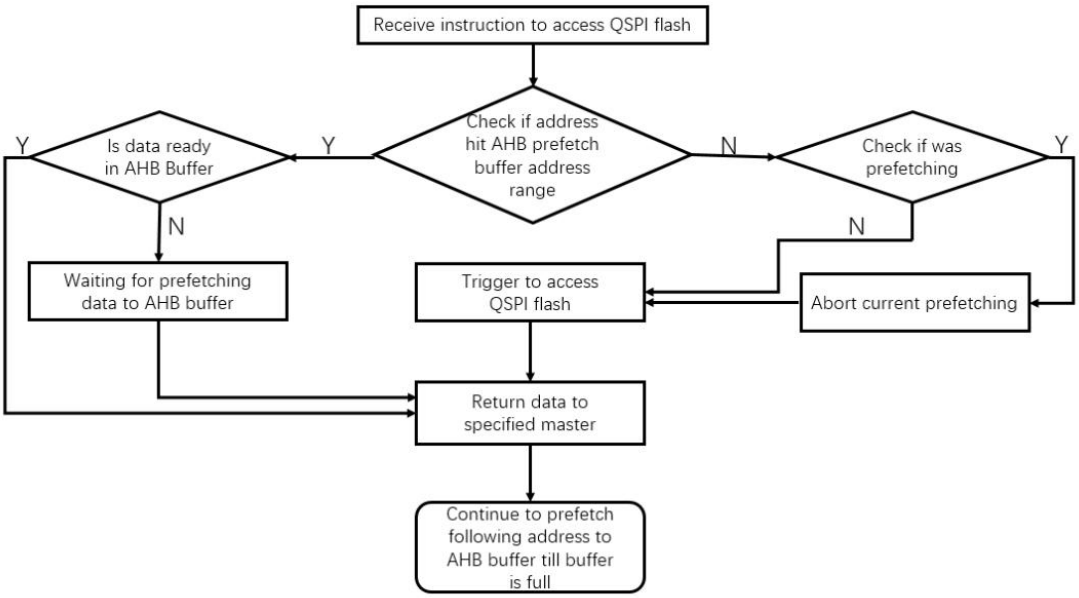


图 3 FlexSPI 预取流程

3.3 不同内存的性能比较

为了评估在不同内存中执行代码的性能，需要进行比较在不同存储设备里面运行相同代码的运行时间。

以一种常见的音频编码算法 OPUS 为例，它对保存在 SD 卡中的相同波形文件进行编码，并计算将波形编码为 OPUS 格式的持续时间。表 6 显示了测试结果。

表 6 性能比较

Test application	Code size (:Bytes)	Code location	Flash speed	Avergae speed (:μS)
Opus (encoder)	188 238	Hyper flash	166 MHz DDR	828364
	188 238	Hyper flash (encrypted image)	166 MHz DDR	847894
	188 238	QSPI flash	127 MHz SDR	1065541
	188 238	SDRAM	163 MHz SDR	826454
	188 238	ITCM	600 MHz	732964

表 6 显示了不同内存中的性能比较。它在 ITCM 上运行时获得最佳性能，而在 QSPI 闪存上获得最低的性能。QSPI 闪存比 TCM 慢约 45%，比 Hyper 闪存或 SDRAM 慢 28%。它在固件加密运行也略有下降，约为 2.6%。

性能取决于应用案例。在某些应用程序中，它在某些内存中运行代码的会下降很多（QSPI 闪存），而其他应用程序的下降幅度较小。在某些应用程序中，有可能获得相同的性能。例如，在表 7 中，在不同的内存中运行 CoreMark 几乎得到了相同的分数。

表 7 CoreMark 得分

Memory	Hyper flash	QSPI flash	SDRAM	ITCM
CoreMark Score(:/MHz)	5.059662	5.059659	5.059659	5.059659

表 7 显示性能是由应用程序和代码优化决定的。代码优化意味着优化代码以获得较高的缓存命中率，并将关键代码放置在 ITCM 上以提高性能。详情请参阅下面如何提高性能。

4、如何提高性能

改善性能的方法包括:

•保持较高的缓存命中率。

•将关键代码放置到内部 RAM。

影响缓存命中率的因素包括:

•查询表

•跳转

大多数应用都会用到查表。当经常查表时，他每次读取小部分数据，甚至是每次一个字节。每次读取访问都可能导致高速缓存未命中。并且从闪存读取新的数据，因为频繁的访问闪存导致性能下降。频繁跳转也会影响性能。因为这种情况会频繁导致缓存不命中。对于这些应用，提高性能的最佳方法是将代码分配到内部 SRAM（ITCM /DTCM）。

4.1 将部分代码分配给指定的内存

本节介绍如何将代码分配给内部 RAM 或其他指定的存储器。

有关 SDK 如何将部分代码分配到内部 SRAM 的信息，请参阅 power_mode_switch 的演示。它在 lpm.h 中定义了 MACRO **QUICKACCESS_SECTION_CODE** 为的是把函数分配到指定的存储器，如 图 4 所示。

```
/*! @name Time sensitive region */
/* @{ */
#if defined(XIP_EXTERNAL_FLASH) && (XIP_EXTERNAL_FLASH == 1)
#if (defined(__ICCARM__))
#define QUICKACCESS_SECTION_CODE(func) __ramfunc func
#elif(defined(__ARMCC_VERSION))
#define QUICKACCESS_SECTION_CODE(func) __attribute__((section("RamFunction"))) func
#elif defined(__MCUXPRESSO)
#define QUICKACCESS_SECTION_CODE(func) __attribute__((section(".ramfunc.$SRAM_ITC"))) func
#elif(defined(__GNUC__))
#define QUICKACCESS_SECTION_CODE(func) __attribute__((section("RamFunction"))) func
#else
#error Toolchain not supported.
#endif /* defined(__ICCARM__) */
#else
#if (defined(__ICCARM__))
#define QUICKACCESS_SECTION_CODE(func) func
#elif(defined(__ARMCC_VERSION))
#define QUICKACCESS_SECTION_CODE(func) func
#elif(defined(__MCUXPRESSO))
#define QUICKACCESS_SECTION_CODE(func) func
#elif(defined(__GNUC__))
#define QUICKACCESS_SECTION_CODE(func) func
#else
#error Toolchain not supported.
#endif
#endif /* __FSL_SDK_DRIVER_QUICK_ACCESS_ENABLE */
```

图 4 函数分配的宏定义

因此，它在链接文件 evkmimxrt1060_power_mode_switch_ca.scf 中定义了一个 RamFunction 内存段，如图 5 所示。

```
  RW_m_data m_data_start m_data_size-Stack_Size-Heap_Size { ; RW data
    .ANY (+RW +ZI)
    * (NonCacheable.init)
    * (NonCacheable)
  }
  ARM_LIB_HEAP +0 EMPTY Heap_Size {    ; Heap region growing up
  }
  ARM_LIB_STACK m_data_start+m_data_size EMPTY -Stack_Size { ; Stack region growing down
  }
  RW_m_ram_text m_text2_start UNINIT m_text2_size { ; load address = execution address
    * (RamFunction)
  }
}
```

图 5 “ RamFunction”的链接文件定义

要将一个函数分配给指定的 RAM，请参阅 图 6。

```
QUICKACCESS_SECTION_CODE(void LPM_SwitchBandgap(void));
QUICKACCESS_SECTION_CODE(void LPM_RestoreBandgap(void));
QUICKACCESS_SECTION_CODE(void LPM_SwitchToXtalOSC(void));
QUICKACCESS_SECTION_CODE(void LPM_SwitchToRcOSC(void));
QUICKACCESS_SECTION_CODE(void LPM_SwitchFlexspiClock(lpm_power_mode_t power_mode));
QUICKACCESS_SECTION_CODE(void LPM_RestoreFlexspiClock(void));
```

图 6 函数分配到指定 RAM 的示例

SDK 中的 fsl_common.h 提供了类似的宏定义便于使用。

5、如何在一个应用程序中识别关键代码

内部 RAM 的大小是有限的，即使 RT 已经提供了大容量 RAM（某些产品最多 2 M 字节），但还是不足以将某些应用的所有代码都放置到内部 RAM，并且很难知道哪个函数对性能的影响是最大的。一个复杂的应用程序可能包含数百个函数，因此函数分析很困难，并且需要付出更多的努力。下面介绍了一种通过 IDE 工具（IAR 和 MDK）进行函数分析的简单方法。

5.1 函数分析

许多 IDE 工具都支持通过串行线跟踪（SWO）或内嵌的跟踪模块（ETM）进行功能分析。i.MX RT 提供了 ETM 和 SWO TRACE 的全面支持。下面以 IAR 和 MDK 为例说明如何获取函数分析。

5.1.1 硬件设置

i.MX RT 评估板保留了 20 路 J-Link 连接器。可以用 J-link 通过 SWO 来进行跟踪，也可以修改评估版用 U-link 通过 ETM 来进行跟踪。

•SWO 跟踪

SWO 支持来自内核的单引脚输出信号。i.MX RT 系列支持 SWD 和 JTAG 调试。表 8 介绍 JTAG 和 SWD 的 J-link 定义。

表 8 J-link 连接器定义

Pin number	Pinout for JTAG	Pinout for SWD	Pin number	Pinout for JTAG	Pinout for SWD
1	V_{tref}	V_{tref}	2	NC	NC
3	n_{TRST}	NC	4	GND	GND
5	TDI	NC	6	GND	GND
7	TMS	SWDIO	8	GND	GND
9	TCK	SWCLK	10	GND	GND
11	RTCK	NC	12	GND	GND
13	TDO	SWO	14	GND	GND
15	RESET	RESET	16	GND	GND
17	QBGRQ	NC	18	GND	GND
19	5 V - supply	5 V - supply	20	GND	GND

i.MX RT1050 和 i.MX RT1020 将 SWO 跟踪信号映射到不同的引脚，该引脚与 JTAG_TDO 不复用。需要修改开发板将 TRACE_SWO 飞线连接到 J-link 连接器的针脚 13。对于 i.MX RT1060，TRACE_SWO 的 JTAG_TDO 连接到同一个管脚，并且连接到 J-link 连接器。SWO 跟踪可以正常工作而无需任何更改。

•ETM 跟踪

ETM 是硬件微单元。连接到内核时，ETM 在一个跟踪端口上输出指令和数据跟踪信息。ETM 通过符合 ATB 协议的跟踪端口来跟踪内核。

ULINK 支持 ETM 跟踪和相应的连接器，如下所示 图 7。

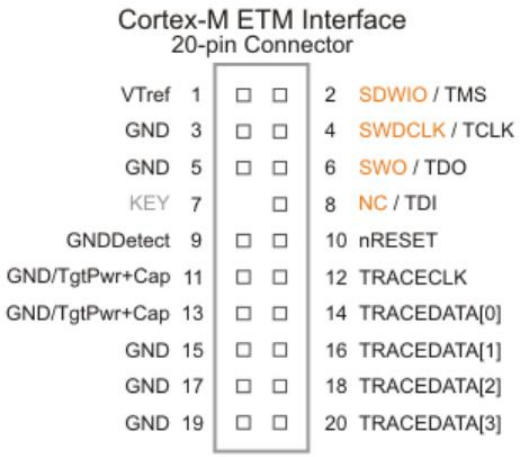


图 7. Cortex-M ETM 界面

要实施 ETM，请用线将 MCU 和 Cortex-M 连接器连接至以下信号。

— TRACECLK

— TRACEDATA [0]

— TRACEDATA [1]（可选）

— TRACEDATA [2]（可选）

— TRACEDATA [3]（可选）

5.1.2 软件设置

要启用 i.MX RT 跟踪功能，必须启用 TRACE 时钟并配置适当的 pinmux。

•配置 i.MX RT1060 以启用 SWO 功能的示例：

—跟踪时钟配置

```
CLOCK_EnableClock(kCLOCK_Trace)
```

```
CLOCK_SetDiv(kCLOCK_TraceDiv, 2)
```

```
CLOCK_SetMux(kCLOCK_TraceMux, 2)
```

—— PAD 配置

```
IOMUXC_SetPinMux(IOMUXC_GPIO_AD_B0_10_ARM_TRACE_SWO, 0U)
```

•配置 i.MX RT1060 以启用 ETM 功能的示例：

—— 跟踪时钟配置

```
CLOCK_EnableClock(kCLOCK_Trace);
CLOCK_SetDiv(kCLOCK_TraceDiv, 3);
CLOCK_SetMux(kCLOCK_TraceMux, 0);
```

—— Pad 配置

```
IOMUXC_SetPinMux(IOMUXC_GPIO_B0_12_ARM_TRACE_CLK,  0U);
IOMUXC_SetPinMux(IOMUXC_GPIO_B0_04_ARM_TRACE0,  0U);
IOMUXC_SetPinMux(IOMUXC_GPIO_B0_05_ARM_TRACE1,  0U);
IOMUXC_SetPinMux(IOMUXC_GPIO_B0_06_ARM_TRACE2,  0U);
IOMUXC_SetPinMux(IOMUXC_GPIO_B0_07_ARM_TRACE3,  0U);
```

5.1.3 IDE 配置

下面以 IAR 为例，介绍如何设置函数分析工具。

1.将 J-Link 连接到目标板 MIMXRT1060-EVK，然后单击 **J-Link** 以配置 SWO，如图 8 所示。

图 8 SWO 配置

2.单击 **Function Profiler** 以打开“**Function Profiler**”窗口。 图 9 显示了如何打开 **J-link** 窗口和 **Function Profile** 窗口。

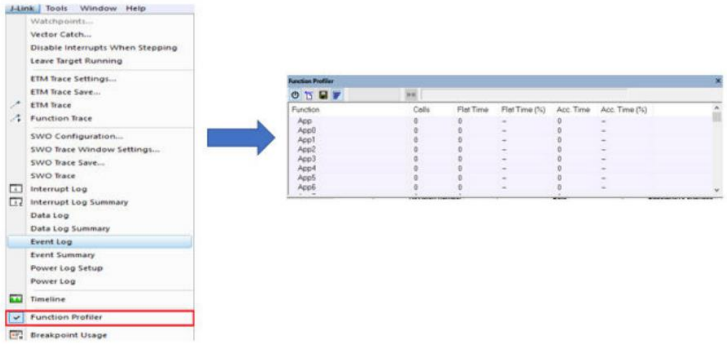


图 9 Functio Profile 窗口

3. 右键单击 **Function**，然后选择 **source:Sampling**，如图 10 所示。

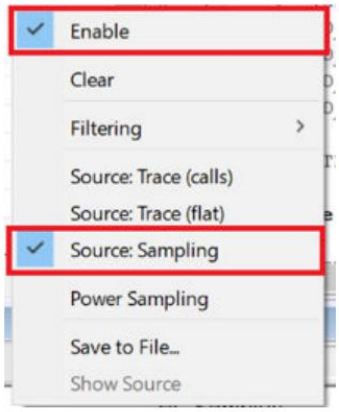


图 10 Function Profile 设置

运行一段时间的代码，然后停止以检查函数分析，如图 11 所示。

Function	PC Samples	PC Samples (%)	Address
App7	127637	25.05	0x6017c800-0x6...
App8	104876	20.58	0x6018c800-0x6...
App6	80883	15.88	0x60164400-0x6...
App5	76105	14.94	0x6015b000-0x6...
App	38493	7.56	0x60003e42-0x6...
App2	21685	4.26	0x60003d48-0x6...
App4	14601	2.87	0x60003df0-0x6...
App3	14516	2.85	0x60003d9c-0x6...
App1	14459	2.84	0x60003cf0-0x6...
App0	14458	2.84	0x60003c98-0x6...

图 11 Function Profile 结果

如图中所看到的图 11 中，App7 在此应用程序中占了很高的加载率，然后将该代码优化到 ITCM 来改进性能。

在对 App7 进行了优化之后，性能提高了 18.5%。 测试结果如图 12 所示。

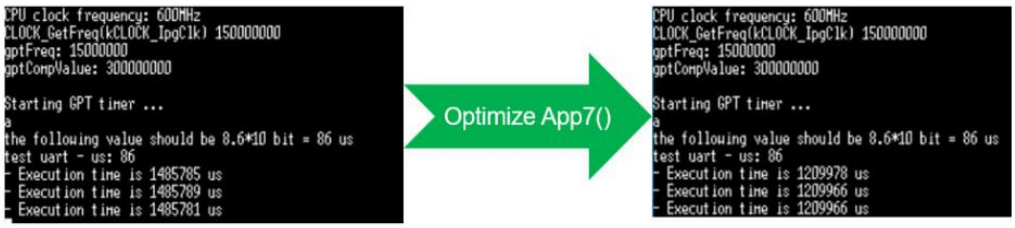


图 12 代码优化的结果

MDK IDE 不支持通过 SWO 跟踪进行函数分析。需要修改开发板以支持 ETM 跟踪。

有关详细信息，请参阅 硬体设定。

以下展示了如何执行 ULINK Pro 设置。

1. 将 ULINK Pro 连接到目标板，即修改过的 MIMXRT1060-EVK。选择正确的调试器，然后单击 **settings**，如图 13 所示 。

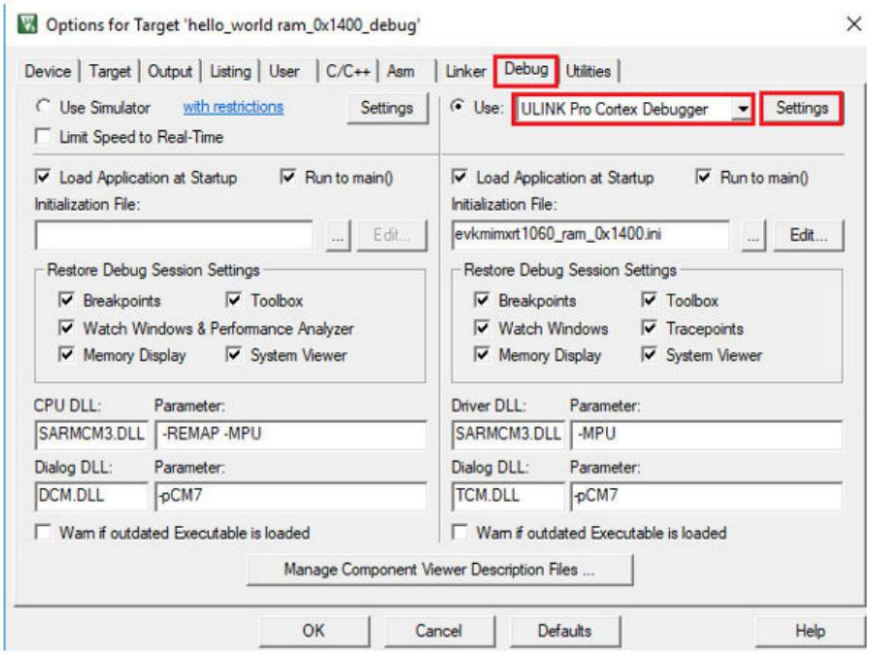


图 13 MDK 中调试器的设置

2. 单击 **Trace** 以设置 ETM，如图 14 所示。

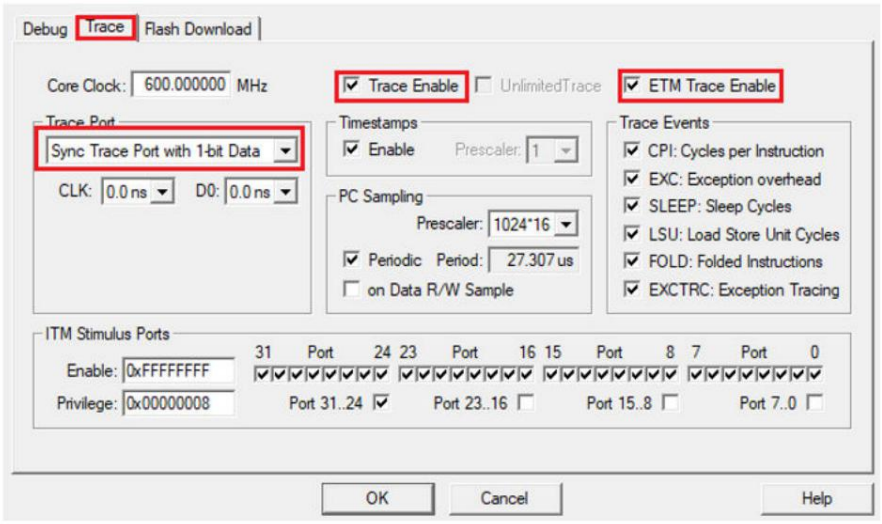


图 14 MDK 中的跟踪设置

1 位数据或者其他选项的跟踪端口的选择由硬件连接确定。

3. 图 15 显示了 **Performance Analyze** 窗口。

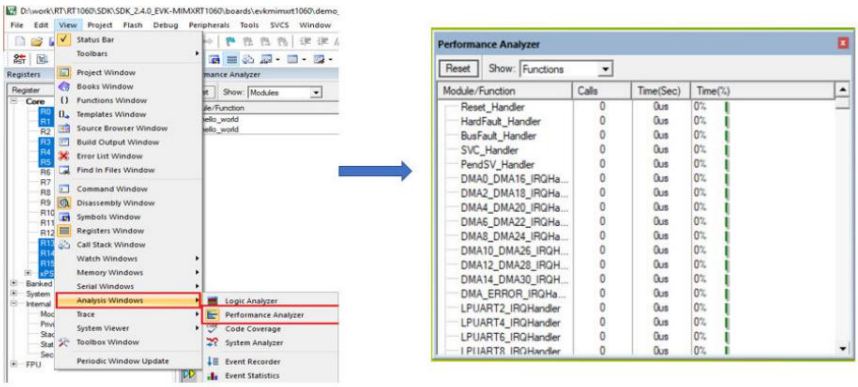


图 15 Performance Analyze 窗口

4.运行代码，然后停下来在 Performance Analyzer 窗口中看性能。

6.结论

本文档介绍了 i.MX RT10xx 系列支持的系统架构和内存性能，以及如何通过 IDE 工具进行函数分析并用来提高性能。该文档可帮助客户优化代码并在使用 i.MX RT 系列时获得良好的性能。

7. 修订历史

表 9 修订历史

Revision number	Date	Substantive changes
0	05/2019	Initial release
1	02/2020	Update Memory performance test and SDRAM performance

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

marks are trademarks and service marks licensed by Power.org.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: Febuary 2020

Document identifier: AN12437