AN0092
应用笔记

AT32F435/437 Performance Improve

前言

这篇应用笔记描述了如何通过软件方法改善 AT32F435_437 的运行效能。

支持型号列表：

支持型号	AT32F435 系列
	AT32F437 系列

目录

表目录

图目录

1 性能改善概述

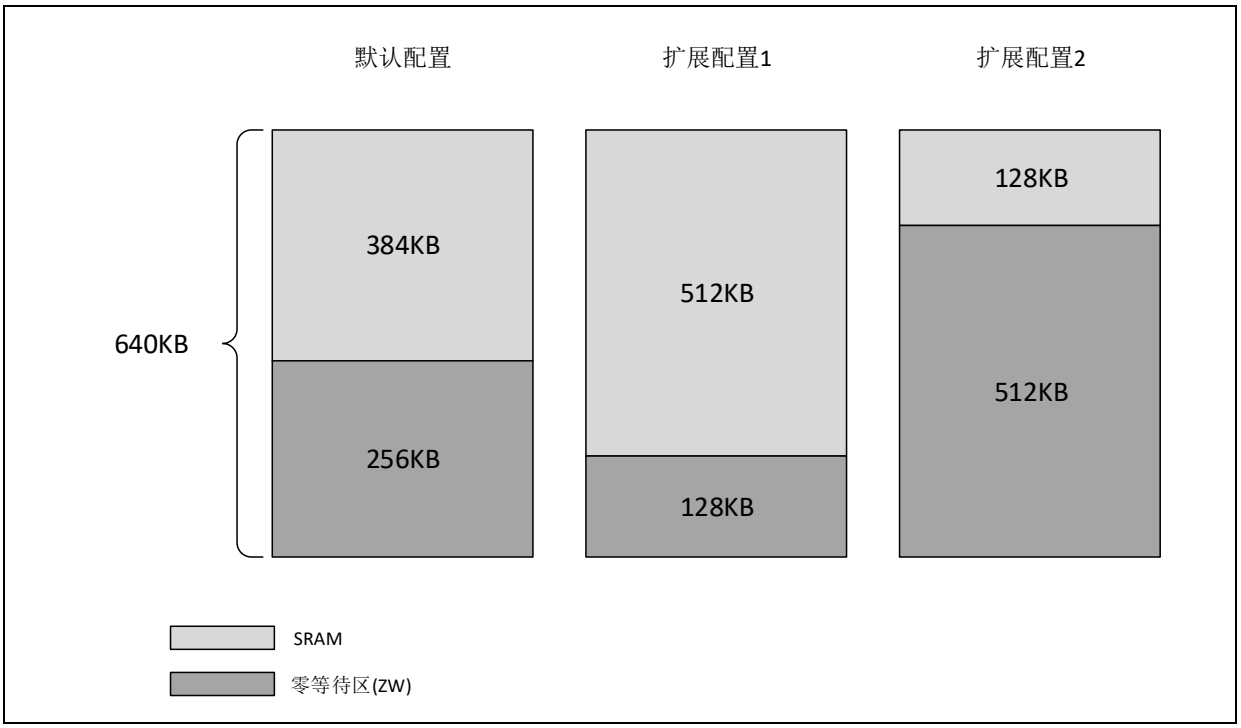
系统性能改善是多方面调优共同作用的结果。在着手改善之前需要对整个系统的软硬件结构和参数有深入的了解。硬件需要对如FLASH大小、SRAM大小、零等待区（ZW）和非零等待区（NZW）大小等参数有准确的认识，软件需要对整个流程熟悉，对代码、算法的执行时间和重要数据的访问频率等信息有个大致的判断。第二步再结合工程内容的实际情况具体分析，一步步进行系统优化，以达到改善性能的目的。本文主要从以下几个方面介绍如何改善AT32F435/437效能：

- SRAM和ZW的扩展配置。
- SRAM1和SRAM2的使用。
- FLASH连续读取使能开启。

2 SRAM 和零等待区（ZW）的扩展配置

对于AT32F435/437系列MCU，在默认情况下片内包含384KB SRAM和256KB ZW，总共640KB，根据各种不同需求，用户可以自行选择对SRAM和ZW的大小进行重新配置，下图展示的三种情况分别是默认配置、最大SRAM和最大ZW。

图 1. 扩展配置典型模式



2.1 需求分析与取舍

- ZW是芯片内部为实现高速运行而在FLASH与CPU之间做的一个预加载存储区，其本质也是一块RAM。在芯片上电后，硬件自动从FLASH的起始地址拷贝ZW大小的数据到ZW区，实现系统后期运行时对预加载数据的快速访问。相对于非零等待区，程序运行在ZW区能获得更佳的性能。
- SRAM是系统运行时的存储区域，其中存储的数据包括指令、数据、堆栈等信息，是程序正常运行所必需的区域。

ZW和SRAM的大小肯定都是越大越好，但因为两者总大小固定为640KB，所以为了做出合理的分配和取舍，用户需要根据实际应用进行扩展配置，以下是一些常用场景的推荐扩展配置：

1. 当系统编译完成，检查运行所需的RAM，如果大于芯片默认SRAM，则必须调整扩展配置，增加SRAM，减小ZW，使配置的SRAM大于系统正常运行所必需的大小。
2. 当系统编译完成，检查运行所需的ROM，如果大于芯片默认的ZW，并且其中有较多需频繁调用的数据或者对效能要求较高的函数：
   a) 首先可以将频繁调用的时间或对效能要求较高的函数，通过修改程序的分散加载文件、代码中直接指定地址等方法将其地址分配到ZW区内。
   b) 如果仍然不满足需求，再检查运行所需的RAM，如果小于芯片默认SRAM，并且即使调整减小SRAM后仍然够用，则可以调整扩展配置，增加ZW，减小SRAM，使ZW区域能尽量包含更多的代码。

2.2 扩展配置方法

扩展配置的控制字节EOPB0位于FLASH内的用户系统数据区（USD）。扩展配置支持以64KB为最小单位的调整，下图是EOPB0支持的配置模式

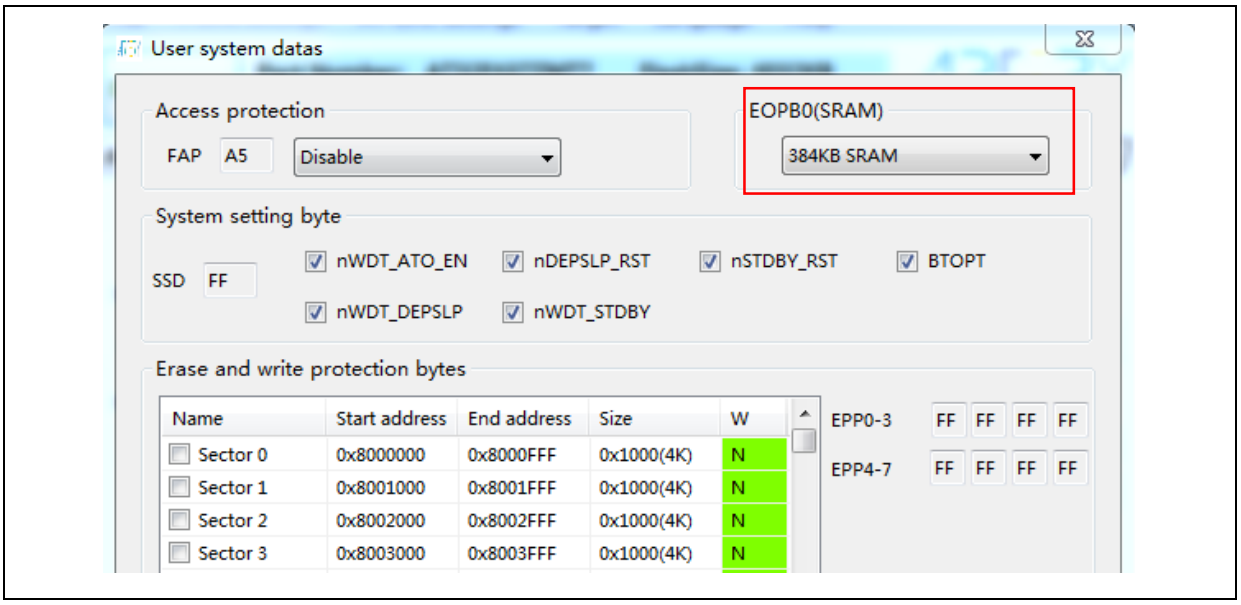
表 1. 扩展的选项系统内容

EOPB0[7：0]：扩充的系统选项		
256K 闪存容量	位1：0	00：片上SRAM 512K 字节 + 闪存零等待延迟区域 128K 字节 01：片上SRAM 448K 字节 + 闪存零等待延迟区域 192K 字节 10、11：片上SRAM 384K 字节 + 闪存零等待延迟区域 256K 字节 注意：位1~0 的改写只能在安全库区未启动的情况下实现
	位7：2	保留不用
1024K 及以上闪存容量	位2：0	000：片上SRAM 512K 字节 + 闪存零等待延迟区域128K 字节 001：片上SRAM 448K 字节 + 闪存零等待延迟区域192K 字节 010：片上SRAM 384K 字节 + 闪存零等待延迟区域 256K 字节 011：片上SRAM 320K 字节 + 闪存零等待延迟区域320K 字节 100：片上SRAM 256K 字节 + 闪存零等待延迟区域384K 字节 101：片上SRAM 192K 字节 + 闪存零等待延迟区域448K 字节 110、111：片上SRAM 128K 字节 + 闪存零等待延迟区域512K 字节 注意：位2~0 的改写只能在安全库区未启动的情况下实现
	位7：3	保留不用

要执行SRAM和ZW的扩展配置方法很多，在量产阶段，推荐用烧录工具，将配置好的用户系统数据（USD）和应用程序CODE一起烧录，在产品开发阶段，可以用ICP工具修改或者用户程序代码执行修改。

用户程序代码中执行扩展配置，详情请参考BSP里边的extend_sram例程和应用笔记AN0026_Extending_SRAM_in_User's_Program.pdf。下图是直接使用ICP工具进行扩展配置修改的界面

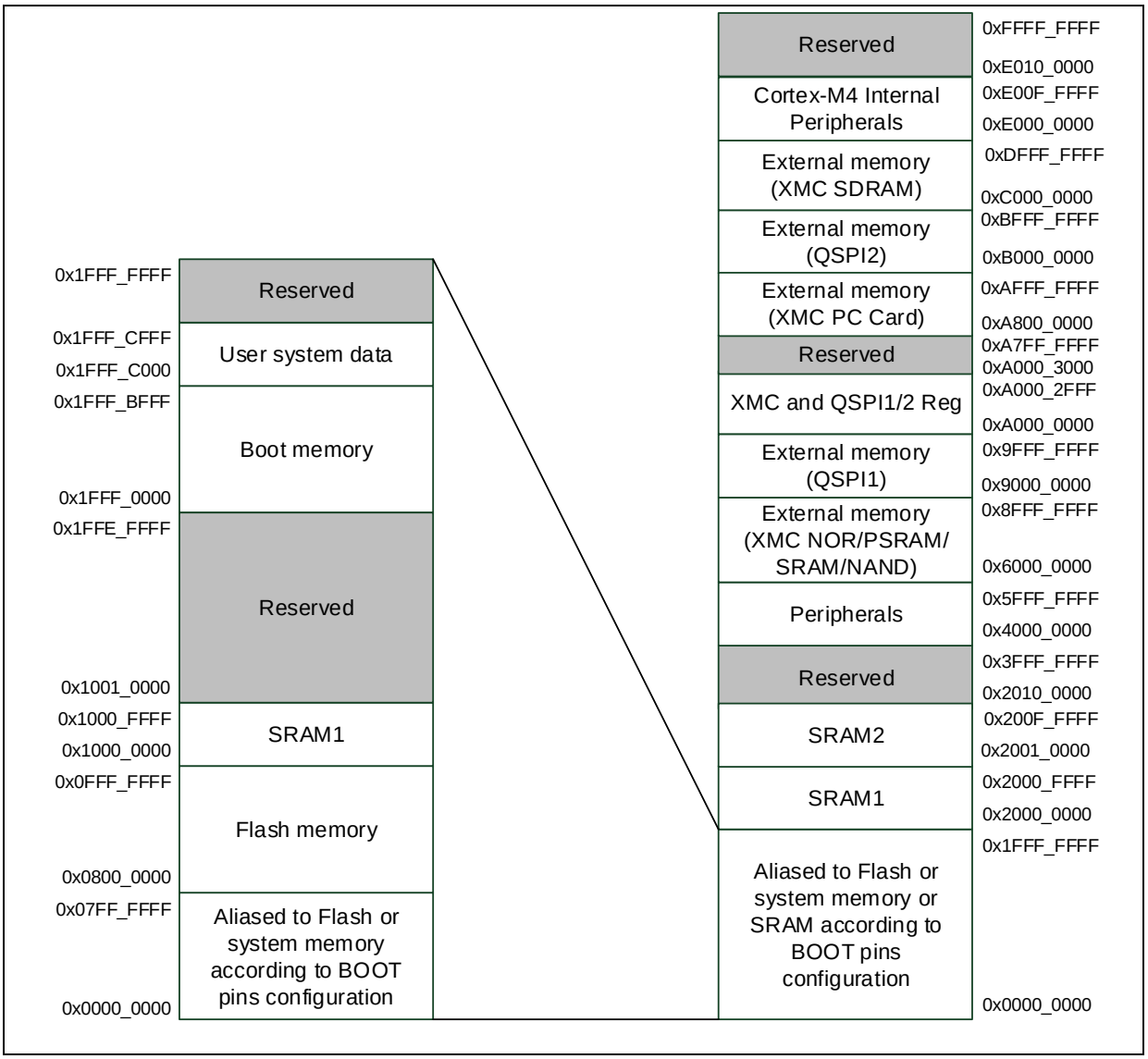
图 2. ICP 工具操作 SRAM 扩展配置



3 SRAM1 和 SRAM2 的使用

AT32F435/437系列的SRAM（128KB~512KB）是由SRAM1和SRAM2组成，其中SRAM1固定大小为64KB，地址从0x2000_0000~0x2000_FFFF，且可以映射到0x1000_0000~0x1000_FFFF。SRAM2会根据用户系统数据区中EOPB0的设定，范围从64KB~448KB，起始位置从0x2001_0000开始，地址结构如下图。本章节主要从SRAM1和SRAM2独立使用和SRAM1映射两个方面提出改善建议。

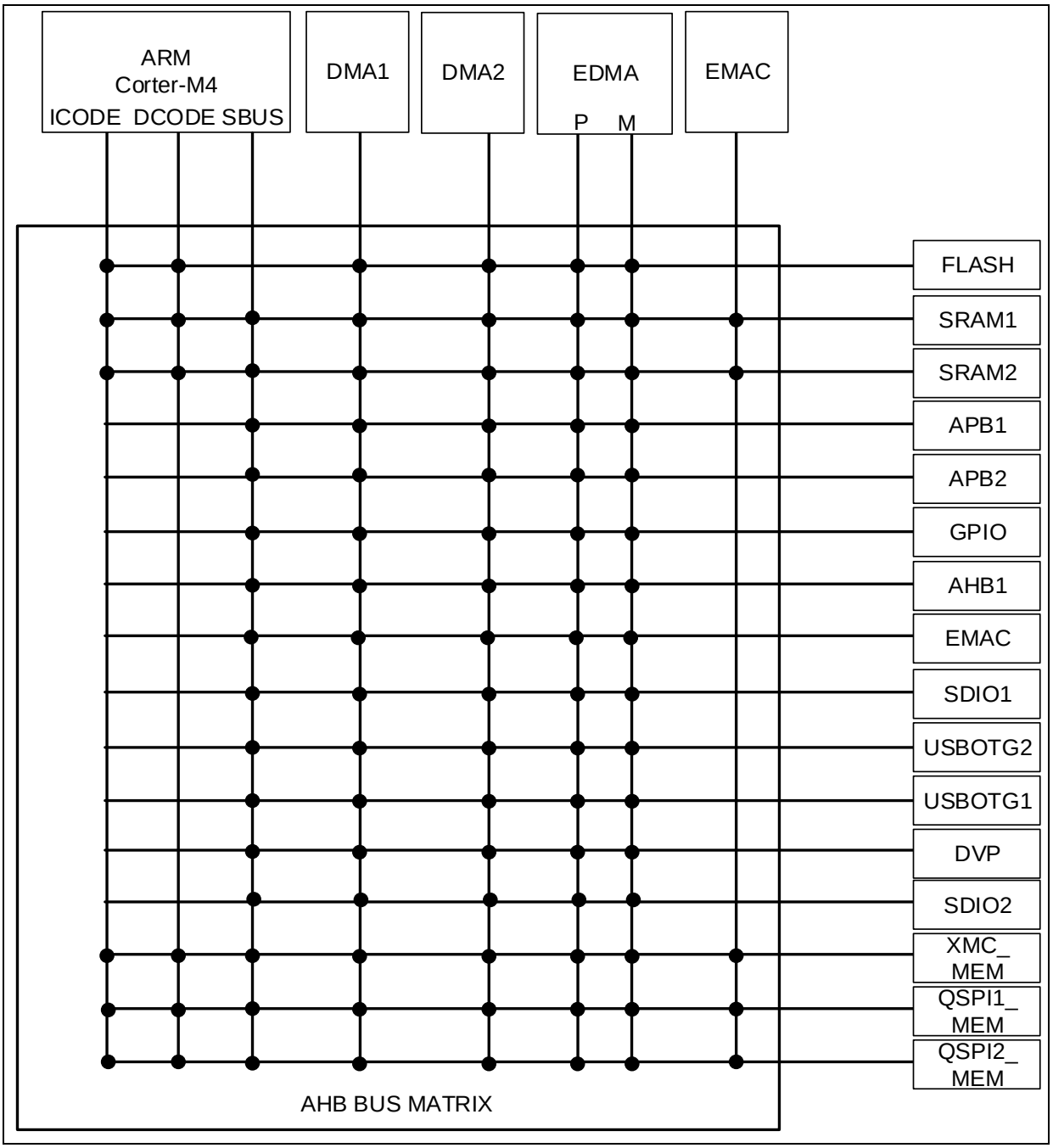
图 3. 地址映射配置图



3.1 SRAM1/SRAM2 独立使用

可以将SRAM1和SRAM2看成两个独立的RAM，从总线矩阵图可以看到SRAM1和SRAM2可以独立的访问，这可以使应用中某些场景下内存的访问速度得到提升。

图 4. AHB 总线矩阵图



举例说明其中一种应用场景：

系统运行中需要从FLASH加载数据，并且期望快速搬运到指定内存使用，程序设计用DMA从FLASH到SRAM进行搬运。

- 如果按照只有一块SRAM的常规设计，那么除了DMA，当程序运行中另外有内存的读写操作时，此时因为只有一块SRAM，访问SRAM的总线就会有冲突，导致效率下降。
- 而按照SRAM1和SRAM2独立的设计，则可以将DMA从FLASH搬运至的内存设置为SRAM1，程序运行中常规使用的内存设置为SRAM2，在两者同时执行时避开冲突提高效率。

设计一个简单的测试程序来进行验证，程序中循环执行IO翻转和变量（SRAM2）自加，同时DMA搬运FLASH数据到内存（SRAM1）。对比IO翻转的频率快慢可以看到，当DMA搬运的内存和程序变量分别来自SRAM1和SRAM2，IO翻转的频率为377KHz，大于DMA搬运的内存和程序变量都来自同一

块RAM（SRAM1或者SRAM2）的347KHz，下面两张图是测试程序中IO翻转的频率对比图。

图 5. 系统只用 SRAM1 或者 SRAM2

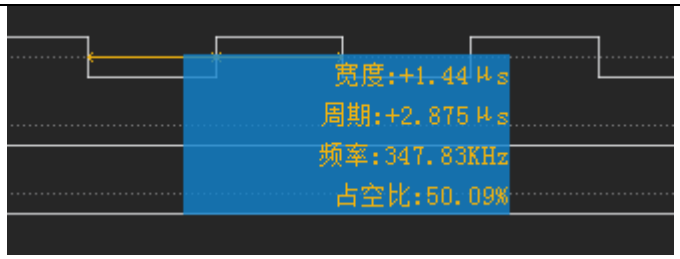
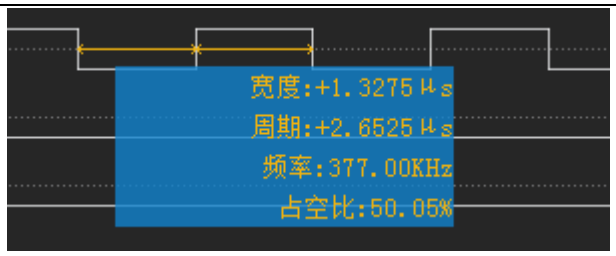


图 6. 系统同时用 SRAM1 和 SRAM2



3.2 SRAM1 映射

从Cortex-M4技术参考手册可知，系统中的I-BUS和D-BUS两条总线地址范围都是0x2000_0000以下，SRAM1地址范围从0x2000_0000~0x2000_FFFF，可以映射到0x1000_0000~0x1000_FFFF，也就是说SRAM1可以通过I-BUS、D-BUS或S-BUS访问，增加了弹性，可以使应用中某些场景下内存的访问速度得到提升。

图 7. 总线地址范围描述

The ICode memory interface. Instruction fetches from Code memory space (0x0000000 - 0x1FFFFFFF) are performed over this 32-bit *Advanced High-performance Bus* (AHB)-Lite bus. For more information, see *ICode bus interface* on page 14-4.

The DCode memory interface. Data and debug accesses to Code memory space (0x0000000 - 0x1FFFFFFF) are performed over this 32-bit AHB-Lite bus. For more information, see *DCode bus interface* on page 14-6.

The System interface. Instruction fetches, and data and debug accesses, to System space (0x20000000 - 0xDFFFFFFF, 0xE0100000 - 0xFFFFFFFF) are performed over this 32-bit AHB-Lite bus. For more information, see *System interface* on page 14-7.

举例说明其中一种应用场景：

系统运行中有多个变量的访问需求。

- 如果按照SRAM1只在0x2000_0000地址的常规设计，那么多个变量都只能通过S-BUS访问，系

统总线只能顺序执行。

- 而按照SRAM1可以映射到0x1000_0000地址，不同的变量可以分别通过S-BUS和D-BUS访问，提高效率。

设计一个简单的测试程序来进行验证，程序中循环执行IO翻转和两个变量A和B自加，变量A来自SRAM1，变量B来自SRAM2。对比IO翻转的频率快慢可以看到，当变量A用SRAM1地址映射0x1000_0000，IO翻转的执行频率2.06MHz，大于变量A用SRAM1地址映射0x2000_0000的2.00MHz，下面两张图是测试程序中IO翻转的频率对比图，

图 8. 变量 A 用 SRAM1 地址映射 0x2000_0000

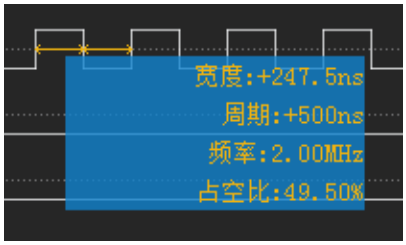
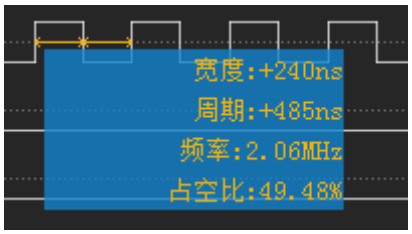


图 9. 变量 A 用 SRAM1 地址映射 0x1000_0000



4 FLASH 连续读取使能开启

FLASH_CONTR寄存器控制内部FLASH连续读取使能的开关，操作其中的FCONTR_EN位置'1'开启该功能。

该功能的开启对运行时需要从FLASH非零等待区读取大容量连续地址数据的应用效率有显著提高。

注意：开启该功能会使MCU在上述时间段内的运行总功耗增加数个毫安。

4.1 开启方法

在程序的初始化阶段，调用BSP里的库函数flash_continue_read_enable()，配置为TRUE即可。

设计一个简单的测试程序来进行验证，程序从非零等待区某个地址开始连续读取32KB的数据，并循环1000次，记录读取时间。对比开启和关闭的读取时间可以看到，关闭该功能耗时3.369s，开启该功能耗时2.231s，时间缩短了大概34%，下面两张图是测试程序的打印截图，打印信息1和2之间的时间即为执行FLASH读取的时间。

图 10. 关闭连续读取功能时间

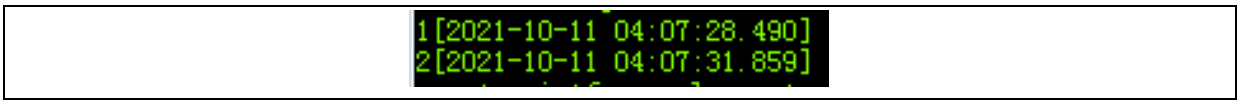
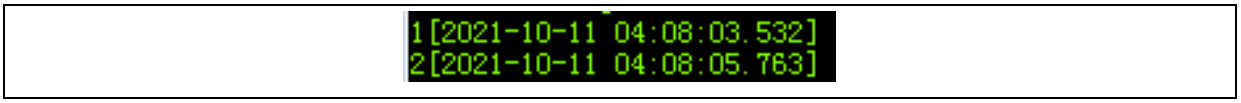


图 11. 开启连续读取功能时间



5 文档版本历史

表 2. 文档版本历史

日期	版本	变更
2021.10.8	2.0.0	最初版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：（A）对安全性有特别要求的应用，例如：生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）航天应用或航天环境；（D）武器，且/或（E）其他可能导致人身伤害、死亡及财产损害的应用。如果采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险及法律责任仍将由采购商单独承担，且采购商应独力负责在前述应用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。