

RT1060 Flexspi Qspinorflash Erase/Write/Read in Xip Mode

May Li/Calvin Ji 2020.10

North China CAS Team

RT1050/1060 在外部 QSPI Flash 启动并且 Code 在 QSPI Flash 执行（我们称之为 XIP 模式）是目前来讲比较常用的做法，而很多应用都会有用到一些常数参数的存储（比如字库，查找表以及图片等等），重要数据的备份与实时记录或者整个应用 firmware 的在线升级，这都涉及到对外部 Flash 的擦写操作，本文档针对实现 RT1050 在 QSPI Flash XIP 的同时还可以在线擦写 QSPI Flash 这个需求总结几点重要的注意事项并配套参考代码。

测试环境：

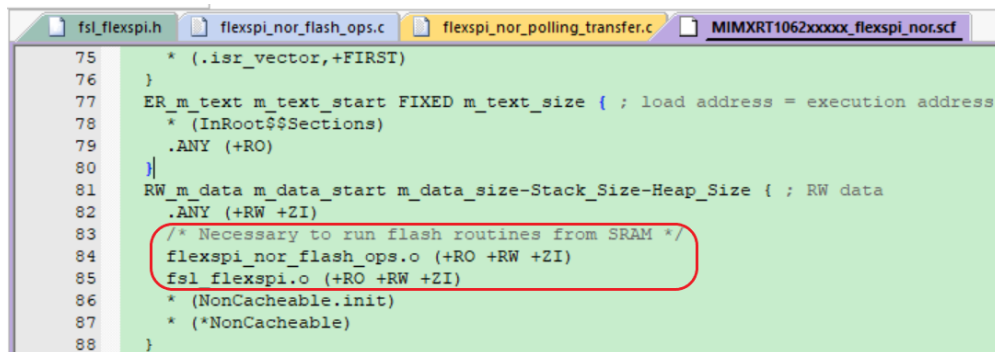
SDK 版本：SDK_v2.8.2

IDE 工具：Keil_v5.31

开发板：MIMXRT1060-EVK

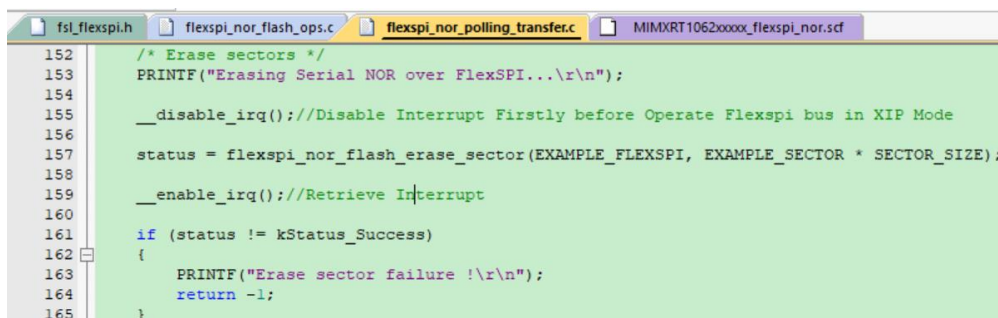
注意事项：

1. 由于需要在 XIP 模式下（eXecute in place）边执行边擦写自己，但现在市面上大部分 Flash（尤其是普通四线 QSPI）是不支持 RWW（Read-While-Write）特性的（只有较少型号支持 RWW，价格比较贵），所以首先要确保所有 Flexspi 操作相关的 API 要跑在 RAM 上而不能再跑在外部 QSPI Flash 上，下图所示方法可以把 flexspi_nor_flash_ops.c 和 fsl_flexspi.c 里的所有读擦写 API 在编译的时候直接链接到 ram 地址，另一种方法也可以通过在所选 API 函数前面添加 `__attribute__((section("NonCacheable")))` 预编译属性也可实现，只是下图方法更直接点，推荐尽量使用下图方法；



```
75  * (.isr_vector, +FIRST)
76  }
77  ER_m_text m_text_start FIXED m_text_size { ; load address = execution address
78  * (InRoot$$Sections)
79  .ANY (+RO)
80  }|
81  RW_m_data m_data_start m_data_size-Stack_Size-Heap_Size { ; RW data
82  .ANY (+RW +ZI)
83  /* Necessary to run flash routines from SRAM */
84  flexspi_nor_flash_ops.o (+RO +RW +ZI)
85  fsl_flexspi.o (+RO +RW +ZI)
86  * (NonCacheable.init)
87  * (*NonCacheable)
88  }
```

2. 在擦写外部 QSPI Flash 的时候需要关闭全局中断，因为无论是中断向量表还是中断服务函数都是默认放在外部 QSPI Flash 地址空间的（在 XIP 模式下），如果在擦写 Flash 期间来一个异步的中断事件同样会导致 hardfault 问题（擦写期间响应 QSPI Flash 里的中断），当然如果用户把中断向量表和中断服务函数都放到 RAM 里的话，这样可以不关闭中断；



```
152  /* Erase sectors */
153  PRINTF("Erasing Serial NOR over FlexSPI...\r\n");
154
155  __disable_irq();//Disable Interrupt Firstly before Operate Flexspi bus in XIP Mode
156
157  status = flexspi_nor_flash_erase_sector(EXAMPLE_FLEXSPI, EXAMPLE_SECTOR * SECTOR_SIZE);
158
159  __enable_irq();//Retrieve Interrupt
160
161  if (status != kStatus_Success)
162  {
163      PRINTF("Erase sector failure !\r\n");
164      return -1;
165  }
```

3. 在擦写完 QSPI Flash 的时候，要想回读 Flash 里的内容，需要清掉 DCache 里的内容（Invalidate 的意思就是告诉 CPU DCache 的内容不可信有可能是脏的，需要从实际物理空间读取），另外由于 Flexspi 有 1KB 的 AHB Pre-fetch buffer，在 Invalidate 仍然回读失败的情况下，可以尝试调用 FLEXSPI_SoftwareReset() 即清掉 AHB Buffer 的内容；

```

167 memset(s_nor_program_buffer, 0xFFU, sizeof(s_nor_program_buffer));
168
169 DCACHE_InvalidateByRange(EXAMPLE_FLEXSPI_AMBA_BASE + EXAMPLE_SECTOR * SECTOR_SIZE, FLASH_PAGE_SIZE);
170 //FLEXSPI_SoftwareReset(EXAMPLE_FLEXSPI); //Clear FlexSPI AHB Pre-Fetch buffer if DCache clean still
171 memcpy(s_nor_read_buffer, (void *) (EXAMPLE_FLEXSPI_AMBA_BASE + EXAMPLE_SECTOR * SECTOR_SIZE), sizeof
172
173 if (memcmp(s_nor_program_buffer, s_nor_read_buffer, sizeof(s_nor_program_buffer)))
174 {
175     PRINTF("Erase data - read out data value incorrect !\r\n ");
176     return -1;
177 }
178 else
179 {
180     PRINTF("Erase data - successfully. \r\n");
181 }

```

4. 对于待烧写到 QSPI Flash 里的 buffer，需要注意该 buffer 必须为 4 个字节对齐，否则会导致写失败；

```

26 /******
27 * Variables
28 *****/
29 /* Make sure the programing buffer is 4bytes aligned */
30 __attribute__((aligned (4))) static uint8_t s_nor_program_buffer[256];
31 __attribute__((aligned (4))) static uint8_t s_nor_read_buffer[256];
32

```

5. 对于不同厂家的 QSPI Flash，用户需要核对选择的 QSPI Flash 数据手册里读擦写的命令是否与代码中 customLUT 里列的一致，如果不一致则需要对应的修改，这里也说明下 LUT 查找表的作用，即把用户访问的指令（包括 AHB 指令或 IP Command 命令）转成外部 SPIflash 可以识别的 SPI 时序命令；

```

59
60 const uint32_t customLUT[CUSTOM_LUT_LENGTH] = {
61     /* Normal read mode -SDR */
62     [4 * NOR_CMD_LUT_SEQ_IDX_READ_NORMAL] =
63         FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x03, kFLEXSPI_Command_RADDR_SDR, kFLEXSPI_
64     [4 * NOR_CMD_LUT_SEQ_IDX_READ_NORMAL + 1] =
65         FLEXSPI_LUT_SEQ(kFLEXSPI_Command_READ_SDR, kFLEXSPI_1PAD, 0x04, kFLEXSPI_Command_STOP, kFLEXSPI_
66
67     /* Fast read mode - SDR */
68     [4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST] =
69         FLEXSPI_LUT_SEQ(kFLEXSPI_Command_SDR, kFLEXSPI_1PAD, 0x0B, kFLEXSPI_Command_RADDR_SDR, kFLEXSPI_
70     [4 * NOR_CMD_LUT_SEQ_IDX_READ_FAST + 1] = FLEXSPI_LUT_SEQ(
71         kFLEXSPI_Command_DUMMY_SDR, kFLEXSPI_1PAD, 0x08, kFLEXSPI_Command_READ_SDR, kFLEXSPI_1PAD, 0x04)
72

```

6. 对于 XIP 模式的代码，芯片在刚启动的时候 ROM 已经对 FlexSPI 模块进行了相关初始化（根据读取 QSPI Flash 前面的 FCB 的配置内容，已经在 evkmimxrt1060_flexspi_nor_config.c 定义了），所以不需要再重新对 Flexspi 进行时钟等相关配置，而是只需要更新下 LUT 表即可；

```

314 void flexspi_nor_flash_init(FLEXSPI_Type *base)
315 {
316     flexspi_config_t config;
317
318     #if defined(DCACHE_PRESENT) && (DCACHE_PRESENT == 1U)
319     #endif
320     #if !(defined(XIP_EXTERNAL_FLASH))
321     flexspi_clock_init();
322
323     /*Get FLEXSPI default settings and configure the flexspi. */
324     FLEXSPI_GetDefaultConfig(&config);
325
326     /*Set AHB buffer size for reading data through AHB bus. */
327     config.ahbConfig.enableAHBPrefetch = true;
328     config.ahbConfig.enableAHBBufferable = true;
329     config.ahbConfig.enableReadAddressOpt = true;
330     config.ahbConfig.enableAHBCachable = true;
331     config.rxSampleClock = kFLEXSPI_ReadSampleClkLoopbackFromDqsPad;
332     FLEXSPI_Init(base, &config);
333
334     /* Configure flash settings according to serial flash feature. */
335     FLEXSPI_SetFlashConfig(base, &deviceconfig, kFLEXSPI_PortA1);
336     #endif /*XIP_EXTERNAL_FLASH*/
337     /* Update LUT table. */
338     FLEXSPI_UpdateLUT(base, 0, customLUT, CUSTOM_LUT_LENGTH);
339 }

```

此外，除了上述方法之外，在 RT1060 以及以后的产品其内部的 ROM 里实际上已经预置了一整套 FlexSPI 操作外部 SPI Flash 的 API 接口函数，而且由于这些函数都是运行在 ROM 空间的，所以这就不需要

担心本文档提到的放到 RAM 中避免 RWW 冲突问题了。只是这个不是本文档介绍的内容，用户如果安装了 IAR 的话可以在 IAR 的安装路径下找到这套 API 的使用参考例程，见\IAR Systems\Embedded Workbench 8.4\arm\src\flashloader\NXP\FlexSPI\FlashIMXRT1060_FlexSPI。