

RT1060 动态修改内部 FlexRAM 分配

Ji Cheng 2020.10

North China CAS Team

RT1050/RT1060 内部的 FlexRAM 默认配置为 128KB ITCM, 128KB DTCM 和 256KB OCRAM (RT1060 还有额外固定的 512KB OCRAM2 空间), 而该默认配置在很多情况下并不能满足用户的实际需求, 此时就涉及到对 FlexRAM 的重分配, 本篇文档会先说明 FlexRAM 的工作机制, 然后再给出实际操作步骤, 希望能解决大部分用户的真正诉求, 更详细的内容请在 NXP 官网下载参阅 AN12077 文档。

测试环境:

SDK 版本: SDK v2.8.2

IDE 工具: Keil v5.31

开发板: MIMXRT1060-EVK

FlexRAM 分配机制:

1. 首先需要看下 RT 内部 512KB (这里以 RT1050/RT1060 为例) FlexRAM 的内部结构如下图 1, 这 512KB 的 RAM 实际上是堆放在一块统一的物理位置的, 具体分别分配多少给 ITCM、DTCM 和 OCRAM 只是通过内部配置开放给不同的总线接口多少物理空间 (OCRAM 总线接口为 AXI64, ITCM 为 TCM_ITF_64, DTCM 为 TCM_IF_32), 而最小分配单位为 $16KB \times 2 = 32KB$ 一个 Bank, 所以 RT1050/1060 一共 16 个 Bank 而 RT1020 为 8 个 Bank;

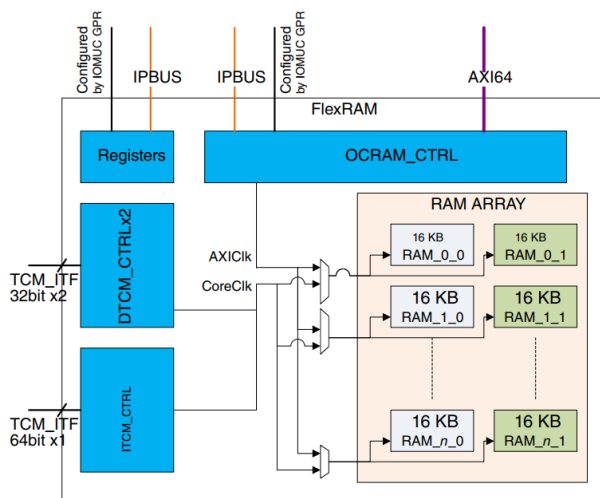


Figure 30-1. FLEXRAM block diagram

2. 其次需要了解不同 Memory 总线的寻址空间如下图，即无论给各个 memory 接口分配多少空间，他们的起始空间是固定的，这样保证地址的继承性；

<u>FlexRAM</u>	Memory Map
ITCM	0x0000_0000~
DTCM	0x2000_0000~
OCRAM	0x2020 0000~

3. 修改 FlexRAM 的空间分配理论上有两种，第一种是通过修改 RT 内部熔丝位的配置（RT 上电启动时默认使用内部熔丝位的配置，而内部熔丝的默认配置是 128KB ITCM，128KB DTCM 和 256KB OCRM），第二种是保持熔丝位不变而是上电后在 CPU 配置堆栈之前修改内部寄存器改变 FlexRAM 分配并使其生效。

第一种方式由于内部熔丝位在芯片出厂之后只能配置一次就固定了，所以大多数应用场景使用第二种方式可灵活满足用户要求，为此 RT 提供了下面三个寄存器实现该功能：

IOMUXC_GPR_GPR16 field descriptions

Field	Description
31-7 CM7_INIT_VTOR	Vector table offset register out of reset. See the ARM v7-M Architecture Reference Manual for more information about the vector table offset register (VTOR).
6-3 -	This field is reserved. Reserved
2 FLEXRAM_BANK_CFG_SEL	FlexRAM bank config source select 0 use fuse value to config 1 use FLEXRAM_BANK_CFG to config
1 INIT_DTCM_EN	DTCM enable initialization out of reset. NOTE: When a TCM is enabled, the corresponding CFG*TCMSZ register must NOT be set to 0'b0000 0 DTCM is disabled 1 DTCM is enabled
0 INIT_ITCM_EN	ITCM enable initialization out of reset. NOTE: When a TCM is enabled, the corresponding CFG*TCMSZ register must NOT be set to 0'b0000 0 ITCM is disabled 1 ITCM is enabled

IOMUXC_GPR_GPR16 field descriptions

Field	Description
31-7 CM7_INIT_VTOR	Vector table offset register out of reset. See the ARM v7-M Architecture Reference Manual for more information about the vector table offset register (VTOR).
6-3 -	This field is reserved. Reserved
2 FLEXRAM_BANK_CFG_SEL	FlexRAM bank config source select 0 use fuse value to config 1 use FLEXRAM_BANK_CFG to config
1 INIT_DTCM_EN	DTCM enable initialization out of reset. NOTE: When a TCM is enabled, the corresponding CFG*TCMSZ register must NOT be set to 0'b0000 0 DTCM is disabled 1 DTCM is enabled
0 INIT_ITCM_EN	ITCM enable initialization out of reset. NOTE: When a TCM is enabled, the corresponding CFG*TCMSZ register must NOT be set to 0'b0000 0 ITCM is disabled 1 ITCM is enabled

34.4.18 GPR17 General Purpose Register (IOMUXC_GPR_GPR17)

GPR Register

Address: 400A_C000h base + 44h offset = 400A_C044h

R	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR17 field descriptions

Field	Description
FLEXRAM_BANK_CFG	FlexRAM bank config value GPR_FLEXRAM_BANK_CFG[2n+1 : 2n], where n = 0, 1, ..., 15 • 00: RAM bank n is not used • 01: RAM bank n is OCRAM • 10: RAM bank n is DTCM • 11: RAM bank n is ITCM

修改步骤：

1. 双击打开 SDK 任意 keil 工程（这里以\boards\evkmimxrt1060\driver_examples\gpio\led_output\mdk 为例），选择 flexspi_nor_debug 工程配置，然后右键工程->Options->Linker 并 Edit 当前配置的 linker 文件，修改如下图，这里以 64KB ITCM，384KB DTCM 和 576KB OCRAM（只针对 RT106x，RT105x 没有 OCRAM2）为例，并把 ramfunction section 放到 itcm 空间里以最优优化 ramfunction 的执行速度；

```

gpio_led_output.c  MIMXRT1062xxxxx_flexspi_nor_flexramchange.scf
38 #define m_ivt_start 0x60001000
39 #define m_ivt_size 0x00001000
40
41 #define m_interrupts_start 0x60002000
42 #define m_interrupts_size 0x00000400
43
44 #define m_text_start 0x60002400
45 #define m_text_size 0x007FDC00
46
47 /* 64KB ITCM for interrupt and high efficiency code */
48 #define m_itcm_start 0x00000000
49 #define m_itcm_size 0x00010000
50 /* 384KB DTCM For Data, heap and stack */
51 #define m_interrupts_ram_start 0x20000000
52 #define m_interrupts_ram_size __ram_vector_table_size__
53 #define m_data_start (m_interrupts_ram_start + m_interrupts_ram_size)
54 #define m_data_size (0x00060000 - m_interrupts_ram_size)
55 /* 64KB OCRAM + 512KB OCRAM2 (only in RT106x) */
56 #define m_data2_start 0x20200000
57 #define m_data2_size 0x00090000
58

```

```

98 }
99 #endif
100 RW_m_data m_data_start m_data_size-Stack_Size-Heap_Size { ; RW data
101 .ANY (+RW +ZI)
102 * (NonCacheable.init)
103 * (*NonCacheable)
104 }
105 ARM_LIB_HEAP +0 EMPTY Heap_Size { ; Heap region growing up
106 }
107 ARM_LIB_STACK m_data_start+m_data_size EMPTY -Stack_Size { ; Stack region growing down
108 }
109 RW_m_ncache m_data2_start EMPTY 0 {
110 }
111 RW_m_ncache_unused +0 EMPTY m_data2_size-ImageLength(RW_m_ncache) { ; Empty region added fo
112 }
113
114 ITCM m_itcm_start m_itcm_size {
115 * (RamFunction)
116 ;gpio_led_output.o(+RO) ;user can take this format to make API code running onto ITCM
117 }
118 }

```

3. 上一步的改动是告诉编译器地址重新分配了，而这一步则要告诉 CPU 内部 RAM 重新分配了，具体修改如下，建议在 ResetHandler 最开始的时候进行重分配即 CPU 对堆栈进行初始化之前；

```

290 /* Reset Handler */
291 .equ __iomux_gpr14_adr, 0x400AC038
292 .equ __iomux_gpr16_adr, 0x400AC040
293 .equ __iomux_gpr17_adr, 0x400AC044
294 /* ITCM=64KB, DTCM=384KB, OCRAM=576KB */
295 .equ __flexram_bank_cfg, 0x5AAAAAA5
296
297 .thumb_func
298 .align 2
299 .globl Reset_Handler
300 .weak Reset_Handler
301 .type Reset_Handler, %function
302 Reset_Handler:
303 cpsid i /* Mask interrupts */
304 /* Flexram config enable */
305 ldr R0, =__iomux_gpr17_adr
306 ldr R1, =__flexram_bank_cfg
307 str R1, [R0]
308 ldr R0, =__iomux_gpr16_adr
309 ldr R1, [R0]
310 orr R1, R1, #4
311 str R1, [R0]
312 /* end of flexram config */
313 .equ VTOR, 0xE000ED08
314 ldr r0, =VTOR

```

4. FlexRAM 分配好之后既告诉编译器也告诉了 CPU，最后还要记得告诉 MPU（M7 内核的 MPU 管理各个地址空间的属性），否则会导致地址对齐问题。参考我们前面的 FlexRAM 大小分配修改如下（在 board.c 里），其中 DTCM 虽然实际分配了 384KB 但是由于 MPU 宏里没有这个大小配置可以配成大于该值，另外下图为 RT1060 的分配，对 RT1050 来说没有 OCRAM2；

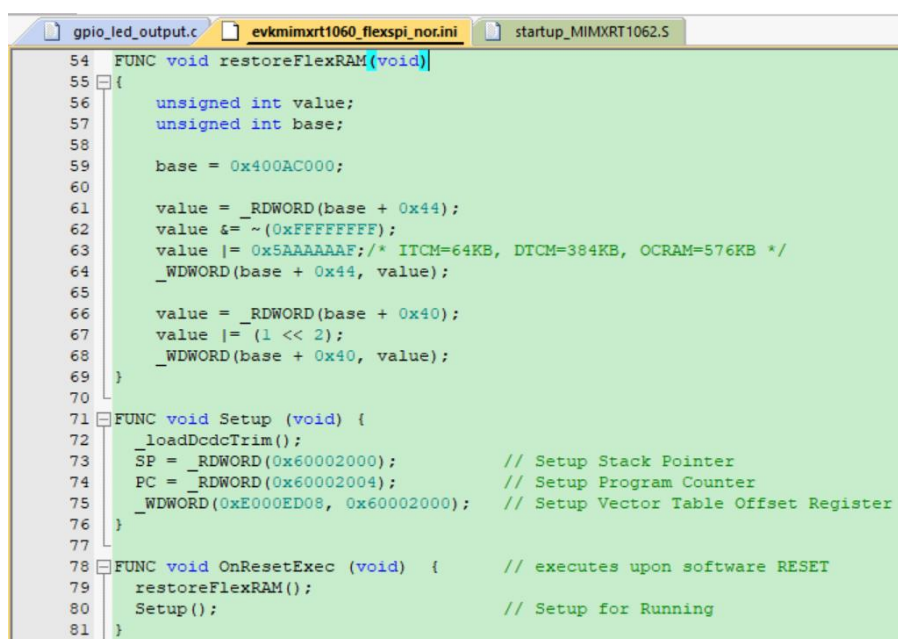
```

337
338 /* Region 5 setting: Memory with Normal type, not shareable, outer/inner write back */
339 MPU->RBAR = ARM_MPU_RBAR(5, 0x00000000U);
340 MPU->RASR = ARM_MPU_RASR(0, ARM_MPU_AP_FULL, 0, 0, 1, 1, 0, ARM_MPU_REGION_SIZE_64KB);
341
342 /* Region 6 setting: Memory with Normal type, not shareable, outer/inner write back */
343 MPU->RBAR = ARM_MPU_RBAR(6, 0x20000000U);
344 MPU->RASR = ARM_MPU_RASR(0, ARM_MPU_AP_FULL, 0, 0, 1, 1, 0, ARM_MPU_REGION_SIZE_512KB);
345
346 /* Region 7 setting: Memory with Normal type, not shareable, outer/inner write back */
347 MPU->RBAR = ARM_MPU_RBAR(7, 0x20200000U);
348 MPU->RASR = ARM_MPU_RASR(0, ARM_MPU_AP_FULL, 0, 0, 1, 1, 0, ARM_MPU_REGION_SIZE_512KB);
349
350 /* Region 8 setting: Memory with Normal type, not shareable, outer/inner write back */
351 MPU->RBAR = ARM_MPU_RBAR(8, 0x20280000U);
352 MPU->RASR = ARM_MPU_RASR(0, ARM_MPU_AP_FULL, 0, 0, 1, 1, 0, ARM_MPU_REGION_SIZE_64KB);
353
354 /* Region 9 setting: Memory with Normal type, not shareable, outer/inner write back */
355 MPU->RBAR = ARM_MPU_RBAR(9, 0x80000000U);
356 MPU->RASR = ARM_MPU_RASR(0, ARM_MPU_AP_FULL, 0, 0, 1, 1, 0, ARM_MPU_REGION_SIZE_32MB);

```

5. 最后右键工程 Options->Debug，编辑.ini 初始化文件，该文件在进入 debug 模式的时候会优先调用执行，在这里面也建议同步修改 FlexRAM 的配置，即保证进入 debug 模式之前把 FlexRAM 配置好。不过此文件修改在 RAM 里 debug 时必须修改，因为 RAM debug 时是要 download 代码到 RAM 里的，因此必须

事先把 RAM 分配好 (ResetHandler 里已经来不及了), 其他模式 Debug 则可以不修改;



```
54 FUNC void restoreFlexRAM(void)
55 {
56     unsigned int value;
57     unsigned int base;
58
59     base = 0x400AC000;
60
61     value = _RDWORD(base + 0x44);
62     value &= ~(0xFFFFFFFF);
63     value |= 0x5AAAAAAF; /* ITCM=64KB, DTCM=384KB, OCRM=576KB */
64     _WDWORD(base + 0x44, value);
65
66     value = _RDWORD(base + 0x40);
67     value |= (1 << 2);
68     _WDWORD(base + 0x40, value);
69 }
70
71 FUNC void Setup (void) {
72     _loadDcdcTrim();
73     SP = _RDWORD(0x60002000); // Setup Stack Pointer
74     PC = _RDWORD(0x60002004); // Setup Program Counter
75     _WDWORD(0xE000ED08, 0x60002000); // Setup Vector Table Offset Register
76 }
77
78 FUNC void OnResetExec (void) { // executes upon software RESET
79     restoreFlexRAM();
80     Setup(); // Setup for Running
81 }
```

至此, 按照上述步骤即可完成 RT1060 内部 FlexRAM 的动态重分配, 用户也可以参考上述方法根据自己实际应用需求进行相应的配置。