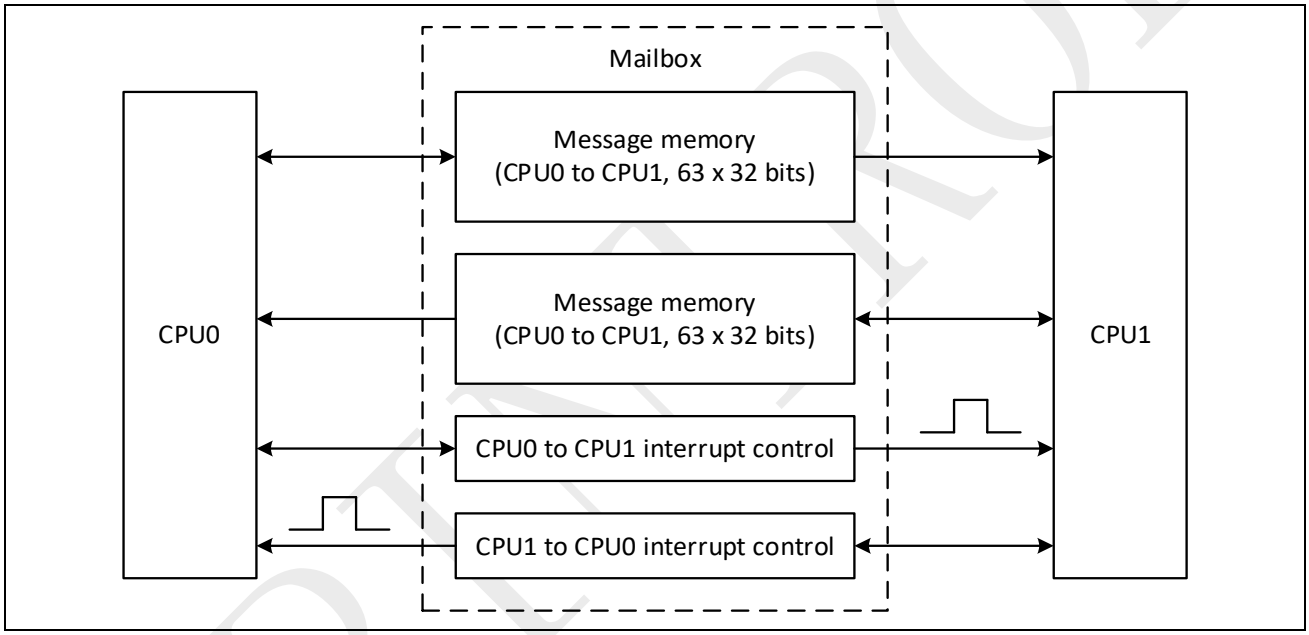
SPC2188 双核使用指南

版本 A/0 – 2023 年 8 月

概述

SPC2188 是双核 CPU，两颗 CPU 都是 ARM CM4 的核。CPU 之间通过 Mailbox 进行握手和共享数据（CPU0 和 CPU1 通过中断来相互请求，请求的消息内容通过 Message RAM 传递）。



目录

图片列表

版本历史

版本	日期	作者	状态	变更
A/0	2023-08-31	X.He	Released	首次发布。

术语或缩写

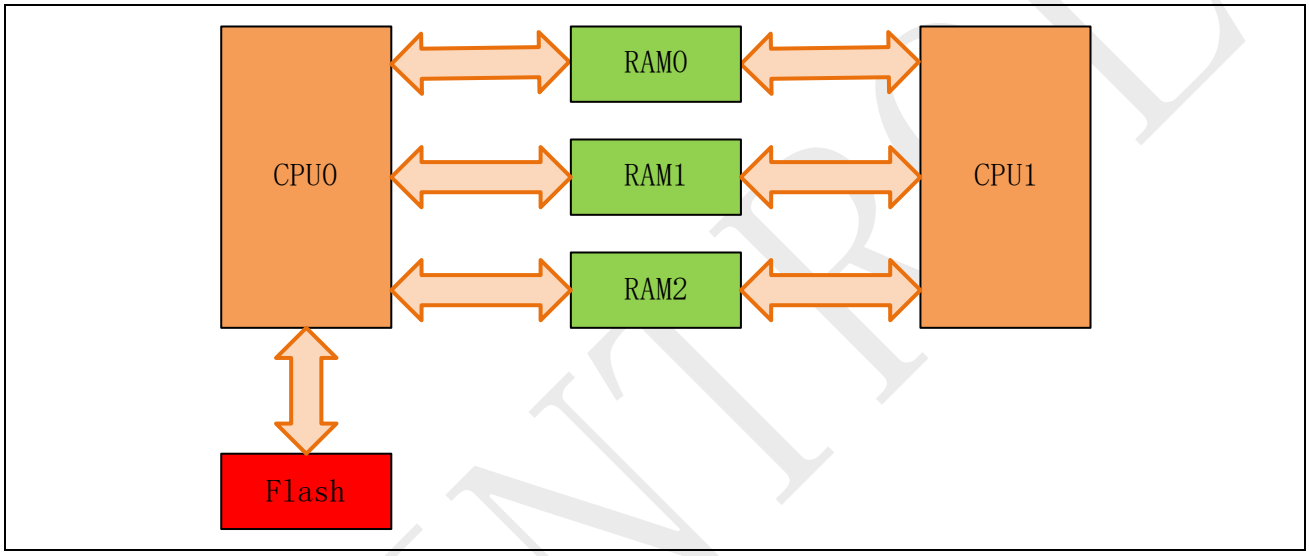
术语或缩写	描述

1 CPU 访问存储器权限

如图 1-1 所示，对于存储区域而言，CPU1 只能访问 RAM 空间，并没有访问 Flash 的能力，所以 CPU1 的代码在存储到 Flash 之后，只能依靠 CPU0 将其代码搬运到 CPU1 可以访问的 RAM 区域，然后有 CPU0 配置 CPU1 使能位，让 CPU1 开始执行其代码。

本手册的后续章节，将对如何烧录 CPU1 程序以及调试方法作详细说明。

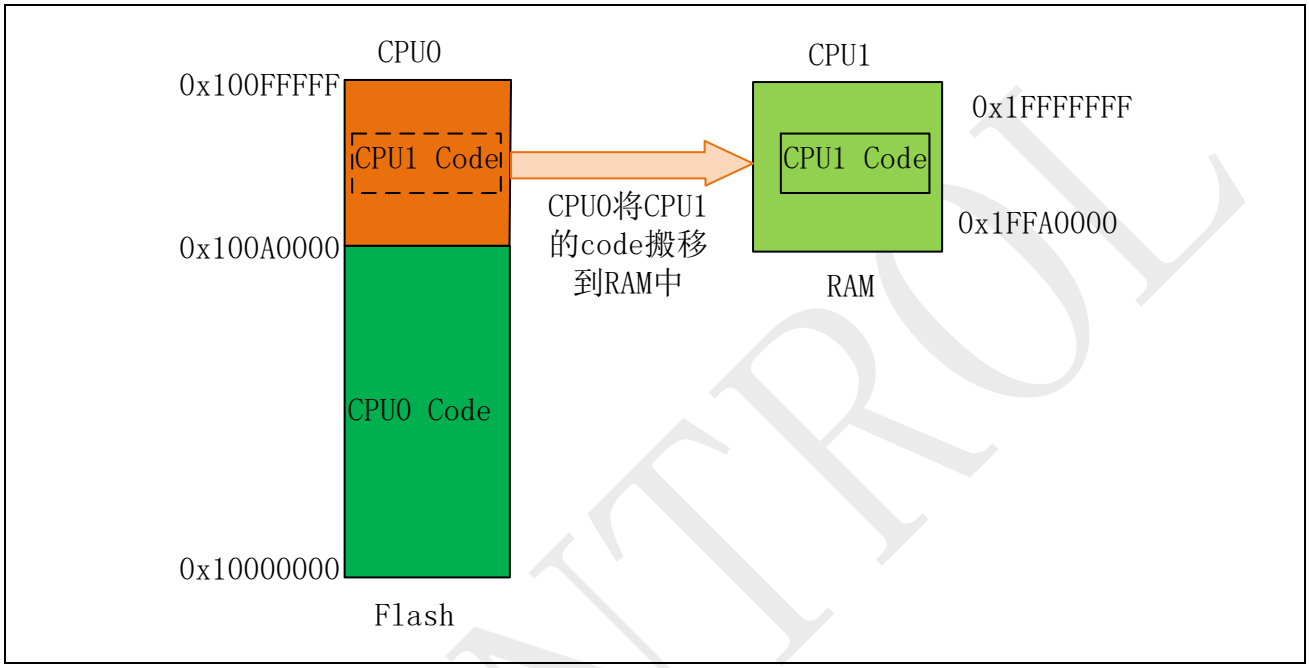
图 1-1：CPU0 和 CPU1 访问存储器权限



2 双核实例

如图 1-1 所示 CPU1 没有访问 Flash 的权限，但用户需要将 CPU1 的代码放在 Flash 中，此时只能通过 CPU0 进行下载到 flash，并且通过 CPU0 将 CPU1 的代码从 Flash 区域搬移到 RAM 区域给 CPU1 执行。

图 2-1：CPU1 代码搬运



2.1 CPU0 向 CPU1 发送数据

本示例演示 CPU0 向 CPU1 发送数据后并接收 CPU1 的数据进行校对。

CPU0 代码执行流程如下：

- 搬运 CPU1 代码，从 Falsh 搬运到指定的 RAM 地址；
- 使能 CPU1；
- 使能 CPU0 的邮箱中断，并往邮箱写入 0x11223344 数据；
- 触发 CPU1 的邮箱中断；
- 等待 CPU0 的中断触发，并在 CPU0 中断服务函数等待接收的数据；
- 校对发送和接收的数据；

Example Code

```
#include <stdio.h>

#include "spc2188.h"

#define CPU1_CODE_START_ADDR 0x100A0000
#define CPU1_CODE_DEST_ADDR  RAM_0_BASE
```

```
volatile uint32_t u32Flag;
volatile uint32_t u32Data;


uint32_t u32AddrSrc, u32AddrDst;
uint32_t *pu32DataSrc, *pu32DataDst;

void CPU1_CodeInit(uint32_t pu32SourceAddr, uint32_t pu32DestinationAddr,
uint32_t u32Size)
{
    uint32_t i;
    for(i = 0; i < u32Size; i++)
    {
        pu32DataSrc = (uint32_t*)pu32SourceAddr;
        pu32DataDst = (uint32_t*)pu32DestinationAddr;

        *pu32DataDst = *pu32DataSrc;

        pu32SourceAddr += 4;
        pu32DestinationAddr += 4;
    }
}


int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_MAX);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    u32Flag = 0;
    u32Data = 0;

    printf("Enter CPU0...\n");

    u32AddrSrc = CPU1_CODE_START_ADDR;
    u32AddrDst = CPU1_CODE_DEST_ADDR;
    /* Code is initially in flash. Need to copy to the CPU1 code-RAM before
running */
    CPU1_CodeInit(u32AddrSrc, u32AddrDst, 1024);

    /* Enable CPU1 */
    SYSTEM_RunCPU1(RAM_0_BASE);

    NVIC_EnableIRQ(MAILBOX_IRQn);

    /* Set the message that CPU0 to CPU1 */
    MAILBOX_CPU0Write32BitMessageToCPU1(0, 0x11223344U);

    /* Trigger an interrupt to CPU1 */
    MAILBOX_CPU0IssueRequestToCPU1();

    /* Wait for the interrupt triggered by CPU1 */
    while(u32Flag == 0)
    {
```

```
    }

    /* Wait CPU1 to print result */
    Delay_Ms(1000);

    /* Check message */
    if(u32Data == 0x12345678U)
    {
        printf("CPU0 check message PASS\n");
    }
    else
    {
        printf("CPU0 check message FAIL\n");
    }

    while(1)
    {

    }
}



void MAILBOX_IRQHandler(void)
{
    /* Read the message from CPU1 */
    u32Data = MAILBOX_CPU0Read32BitMessageFromCPU1(1);

    /* Set Flag */
    u32Flag = 1;

    /* Clear the interrupt from CPU1 to CPU0 */
    MAILBOX_CPU0ClearRequestFromCPU1();
}
```

2.2 CPU0 向 CPU1 发送数据

本示例演示 CPU0 向 CPU1 发送数据后并接收 CPU1 的数据进行校对。

CPU0 代码执行流程如下：

- 关闭 CPU1 的看门狗中断，防止 CPU1 进行复位；
- 使能 CPU1 的邮箱中断；
- 等待 CPU0 的中断触发；
- 在 CPU1 中断服务函数接收数据,然后往邮箱写 0x11223344 数据，再触发 CPU0 的邮箱中断；

Example Code

```
#include <stdio.h>
#include "spc2188.h"
```

```
volatile uint32_t u32Flag;
volatile uint32_t u32Data;


int main(void)
{
    WDT_WALLOW(WDT1);
    WDT_Stop(WDT1);
    WDT_WALLOW(WDT0);
    WDT_Stop(WDT0);

    printf("Enter CPU1...\n");

    u32Flag = 0;
    u32Data = 0;

    NVIC_EnableIRQ(MAILBOX_IRQn);

    /* Wait for the interrupt triggered by CPU0 */
    while(u32Flag == 0)
    {
    }

    /* Check message */
    if(u32Data != 0x11223344U)
    {
        printf("CPU1 check message FAIL\n");
    }
    else
    {
        printf("CPU1 check message PASS\n");
    }


    while(1)
    {

    }
}


void MAILBOX_IRQHandler(void)
{
    /* Read the message from CPU0 */
    u32Data = MAILBOX_CPU1Read32BitMessageFromCPU0(0);

    /* Set Flag */
    u32Flag = 1;

    /* Clear the interrupt from CPU0 to CPU1 */
    MAILBOX_CPU1ClearRequestFromCPU0();


    /* Set message from CPU1 to CPU0 */
    MAILBOX_CPU1Write32BitMessageToCPU0(1, 0x12345678U);

    /* Trigger an interrupt to CPU0 */
    MAILBOX_CPU1IssueRequestToCPU0();
}
```