

## 1 介绍

### 1.1 概览

LPC55xx/LPC55Sxx 是用于嵌入式开发的基于 Arm® Cortex®-M33 的微控制器。这些设备包括：

- 一个 Arm Cortex-M33 协处理器
- 一个 CASPER 加密/FFT 引擎
- 一个用于 DSP 函数的 PowerQuad 硬件加速器
- 多达 320 KB 的片上静态随机存储
- 多达 640 KB 的片上闪存
- PRINCE 模块，用于即时闪存加密/解密 内容
- 高速和全速 USB 主机和设备接口，无晶体操作可实现全速
- SDIO/MMC
- 五个通用计时器
- 一个 SCTimer/PWM
- 一个 RTC/alarm timer
- 一个 24 位多速率计时器 (MRT)
- 一个视窗化看门狗定时器 (WWDT)
- 一个高速 SPI (50 Mhz)
- 九个灵活的串行通信外设 (可以被配置为 USART, SPI, I2C, 或 I2S 接口)
- 可编程逻辑单元 (PLU)
- 一个 16 位 1.0 Msamples/sec ADC
- 一个比较器
- 一个温度传感器

TLPC55xx / LPC55Sxx 提供了两个具有以下功能的 Arm Cortex-M33 内核：

- Arm Cortex-M33 内核 (CPU0, r0p3)
  - 最高以 150 MHz 的频率运行 (仅设备版本 1B)
  - 信任区浮点单元 (FPU) 和内存保护单元 (MPU)
  - Arm Cortex M33 内置的嵌套矢量中断控制器 (NVIC)。
  - 具有各种源的不可屏蔽中断 (NMI) 输入。
  - 具有八个断点和四个监视点的串行线调试。包括用于增强调试功能的串行线输出。

#### 目录

1	介绍.....	1
2	双核实现.....	3
3	示例.....	6
4	多核 SDK.....	8
5	总结.....	9



- 系统节拍定时器。
- Arm Cortex-M33 协处理器 ( CPU1 , r0p3 ) —最高以 150 MHz 的频率运行 ( 仅限设备版本 1B )
  - 此实例的配置不包括 MPU , FPU , DSP , ETM 和 Trustzone。
  - 系统节拍定时器。

## 1.2 双核基本机制

LPC55xx / LPC55Sxx 中的双核是非对称架构, 这意味着一个核 ( CPU0 ) 是主核, 另一个核 ( CPU1 ) 是从核。CPU0 出厂设置为可以正常工作的主核, 而 CPU1 ( 从核 ) 处于保持状态, 并且其时钟在芯片启动时是禁用的。为了使其工作, 从核需要被释放, 并且可以通过主核内的寄存器启用其时钟。

在双核运行模式下, 它们需要相互通信。LPC55xx / LPC55Sxx 提供了一种称为“CPU 间邮箱”机制的简单方法, 该机制具有以下功能:

- 提供一种处理器间通信的方法, 允许多个 CPU 共享资源并以简单的方式相互通信。
- 每个 CPU 最多可以产生 32 个用户定义的中断给另一方。
- 如果有的话每个 CPU 都可以申请共享资源。
- 提供通信握手的互斥配置。

## 1.3 相关系统资源

Arm Cortex M33 包括三条 AHB-Lite 总线, 一条系统总线以及 I-code 和 D-code 总线。一条总线专用于指令提取 ( I-code ), 而一条总线专用于数据读取 ( D-code )。如果并发操作针对不同的设备, 则双核总线的使用允许同时操作。

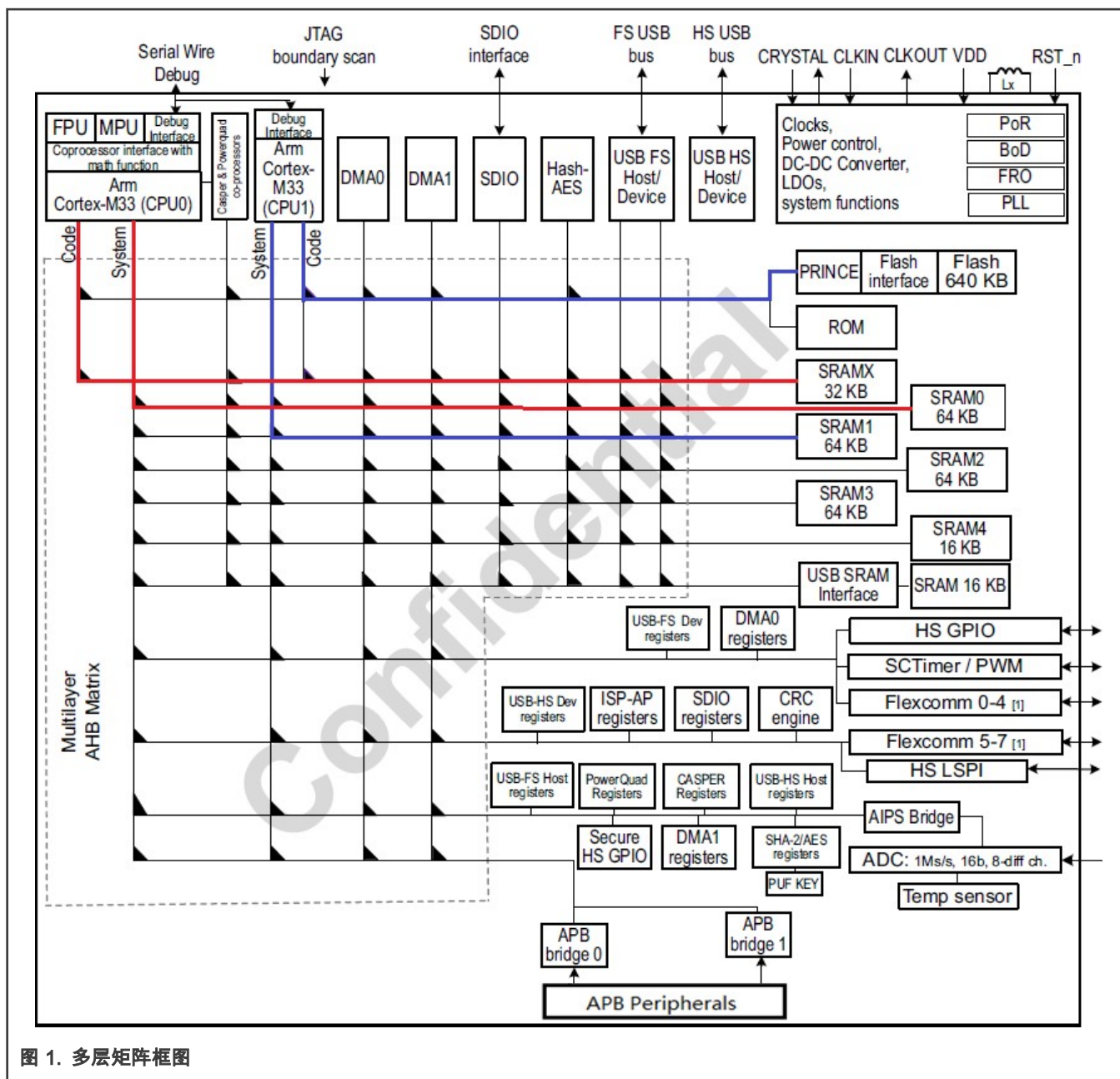
两个 CPU 共享 LPC55xx / LPC55Sxx 中的所有资源 ( 内存和外围设备 )。为了在双核使用上获得更好的性能, 支持以下机制。

### 1.3.1 多组 SRAM

LPC55xx/LPC55Sxx 支持 320 KB SRAM, 具有独立的总线主控器访问权限, 以实现更高的吞吐量, 以及对低功耗操作的独立电源控制功能。320 KB 的 SRAM 包括代码总线上的 32 KB SRAM, 系统总线上的 272 KB SRAM ( 其中 272 KB 是连续的 ) 以及附加的 16 KB USB SRAM。这使得两个 CPU 的代码和数据可以分开进行存储和访问。

### 1.3.2 AHB 多层矩阵

LPC55xx/LPC55Sxx 使用多层 AHB 矩阵, 以灵活的方式将 CPU 总线和其他总线主控设备连接到外围设备, 从而允许不同总线主控设备同时访问矩阵不同端口上的外围设备, 以实现性能优化。图 1 中的多层矩阵框图显示了可用矩阵连接的详细信息。突出显示的总线显示了在双核应用程序上以最佳性能进行的内存访问。



## 2 双核实现

典型的双核的实现包含以下过程：

- 如何在主核（CPU0）上加载从核（CPU1）映像；
- 如何在主核（CPU0）上启动从核（CPU1）；
- 从核（CPU1）与主核（CPU0）之间如何通信。

将基于 LPC55xx SDK 中的 mailbox\_mutex 驱动程序示例来介绍这些实现。

# 2.1 加载从核映像

为了获得良好的性能，从核映像需要被分配在与主核映像不同的总线矩阵层中运行。通常，从核映像分配在独立的 SRAM 组中，而主内核映像分配在 FLASH 中。从核的二进制映像可以包含在主内核的代码中，以一起构建到闪存上的集成映像中。开机后，主核映像将会被执行，同时从核保持出厂设置的状态。

在完成包括邮箱的系统初始化后，主内核将从闪存将核映像加载到其执行空间 SRAM 中。图 2 显示了加载从核二进制映像的过程。

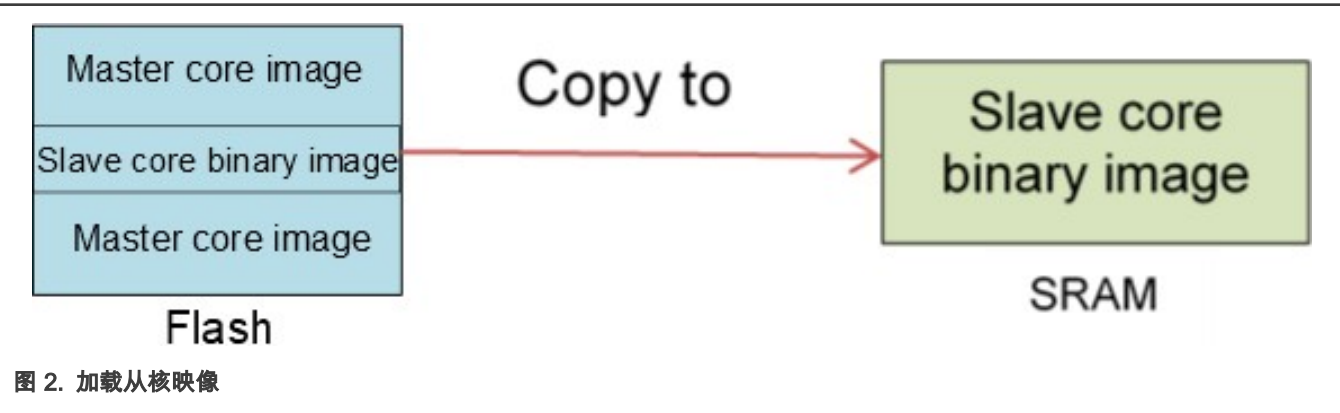


图 2. 加载从核映像

# 2.2 启动从核

在加载到一组 SRAM 中后，从核二进制映像将由主核启动运行。需要启用从核的时钟，并且需要释放从核以进行复位。需要设置从核映像的启动地址，以使从核正确启动。为此，LPC55xx/LPC55Sxx 提供了寄存器，如图 3 所示。

CPU Control for multiple processors (CPUCTRL, offset = 0x800) bit description					
Bit	Symbol	Access	Value	Description	Reset value
2:0	-	WO		Reserved. Read value is undefined, only zero should be written.	undefined
3	CPU1CLKEN	RW		CPU1 clock enable.	0x1
			0	The CPU1 clock is enabled.	
			1	The CPU1 clock is not enabled.	
4	-			Reserved. Read value is undefined, only zero should be written.	undefined
5	CPU1RSTEN	RW		CPU1 reset.	0x1
			0	The CPU1 is being reset.	
			1	The CPU1 is not being reset.	
15:6	-	WO	-	Reserved. Read value is undefined, only zero should be written.	-
31:16		RW		Must be written as 0xC0C4 for the write to have an effect.	-

Co-processor boot address (CPBOOT, offset = 0x804) bit description				
Bit	Symbol	Value	Description	Reset value
31:0	CPBOOT		Co-processor boot address.	0x0

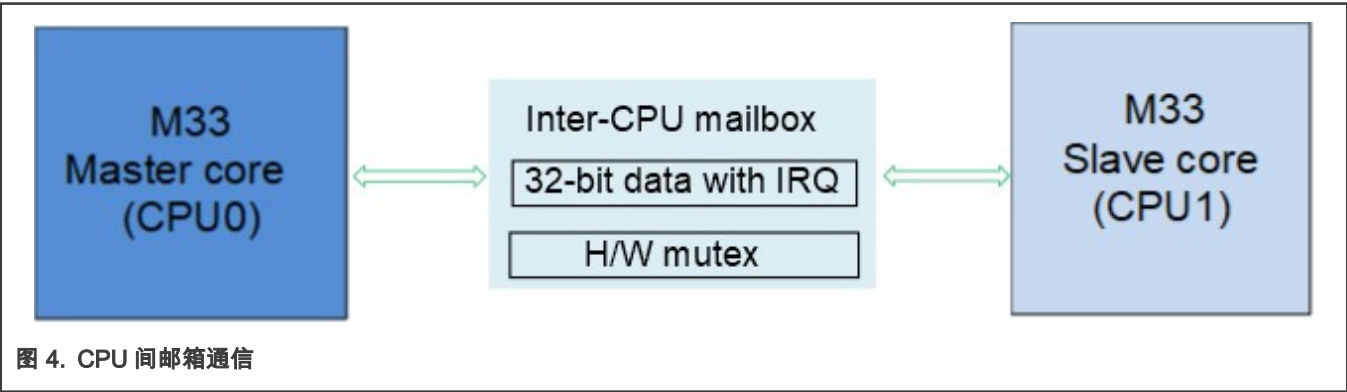
图 3. 用于从核启动运行的寄存器

# 2.3 双核通信

在从核也运行之后，两个内核需要相互通信以进行协作。按照前面提到的，LPC55xx / LPC55Sxx 提供了一种简单的硬件方法，称为“CPU 间邮箱机制”进行通信。在软件上，许多方法可以帮助实现两个内核的通信。软件方法可以简单也可以复杂。共享内存是这些方法中的共同基础。在这里，将介绍硬件 CPU 间邮箱通信机制。

2.3.1 请求通信

在使用之前，邮箱模块需要初始化。即使能该模块的时钟，并使模块复位以工作。为了触发通信，一个 CPU 将向另一个 CPU 发送中断请求。该模块提供寄存器来执行此操作，与此同时，它可以允许彼此发送 32 位非零数据。通信数据代表的内容可以是状态/事件或用户定义的数据。该功能的使用完全取决于用户。图 4 显示了 CPU 间邮箱通信。



2.3.2 同步通信

该机制还通过称为 MUTEX 的寄存器支持互斥控制，以实现通信同步。出于任何原因读取时，将返回寄存器中的当前值并将该位清除。在写入后将再次设置该位。这可以用作两个核之间的资源分配握手。每当一个核企图访问共享资源（例如，mailbox\_mutex 的驱动程序示例中的共享变量）时，它将读取 MUTEX 寄存器。如果是 1，则可以控制共享资源分配。进行任何必要的更改后，它将写入寄存器，使它的 EX 位再次被设置，并使共享资源分配的控制权可用于另一个内核。如果核读取到了 0，则它必须等待该位读取为 1 时才可以读取共享资源分配信息。互斥控制流程如图 5 所示。

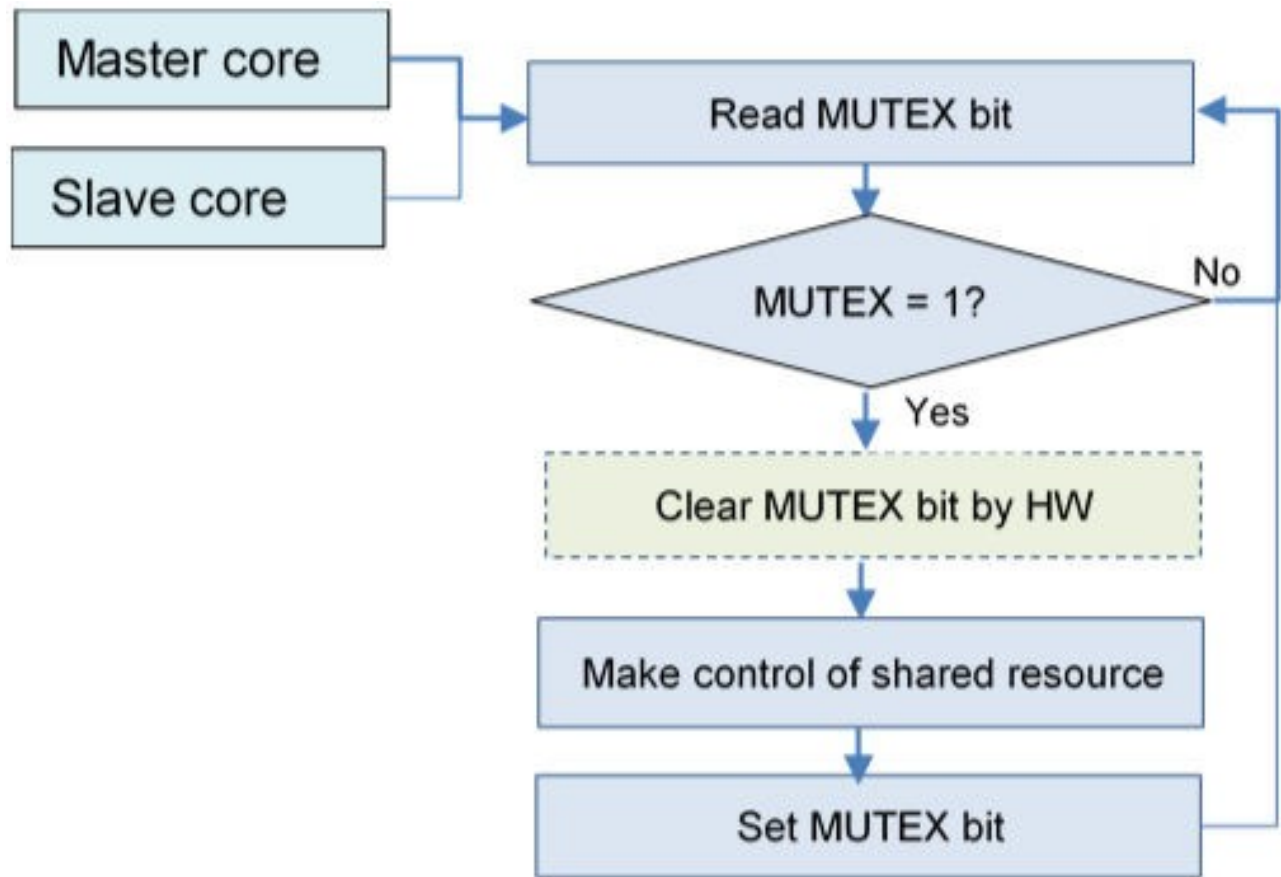


图 5. 互斥控制流程

### 3 示例

LPC55xx SDK 中 mailbox\_mutex 的驱动程序示例中显示了具有互斥控制的 CPU 间邮箱机制。这是一个简单的例子（路径是：boards\lpcxpresso55s69\driver\_examples\mailbox\mutex）。

主核（core0）和从核（core1）分别具有自己的项目。在示例中，内核 0 通过邮箱将共享变量 g\_shared 的地址发送到内核 1（参见图 6），内核 1 再在 ISR 里从邮箱中获取它（参见图 7）。两个内核都试图在 while 循环中获取互斥体。更新共享变量后，将互斥锁设置为允许另一个内核访问共享变量。

主核（core0）和从核（core1）中的主循环过程分别如图 8 和图 9 所示。

```

/* Send address of shared variable to CM33 core1 by Mailbox*/
MAILBOX_SetValue(MAILBOX, kMAILBOX_CM33_Core1, (uint32_t)&g_shared);

static inline void MAILBOX_SetValue(MAILBOX_Type *base, mailbox_cpu_id_t cpu_id, uint32_t mboxData)
{
    base->MBOXIRQ[cpu_id].IRQ = mboxData;
}
  
```

图 6. 发送共享变量地址的代码



```
void MAILBOX_IRQHandler()
{
    g_shared = (uint32_t *)MAILBOX_GetValue(MAILBOX, kMAILBOX_CM33_Core1);
    MAILBOX_ClearValueBits(MAILBOX, kMAILBOX_CM33_Core1, 0xffffffff);
}
```

图 7. 获取共享变量地址的代码

```
while (1)
{
    /* Get Mailbox mutex */
    while (MAILBOX_GetMutex(MAILBOX) == 0)
        ;

    /* The core1 has mutex, can change shared variable g_shared */
    if (g_shared != NULL)
    {
        (*g_shared)++;
        PRINTF("Core1 has mailbox mutex, update shared variable to: %d\r\n", *g_shared);
    }

    /* Set mutex to allow access other core to shared variable */
    MAILBOX_SetMutex(MAILBOX);
}
```

图 8. 从核 (core1) 上的主循环过程

```
while (1)
{
    /* Get Mailbox mutex */
    while (MAILBOX_GetMutex(MAILBOX) == 0)
        ;

    /* The core0 has mutex, can change shared variable g_shared */
    g_shared++;

    PRINTF("Core0 has mailbox mutex, update shared variable to: %d\r\n", g_shared);

    /* Set mutex to allow access other core to shared variable */
    MAILBOX_SetMutex(MAILBOX);
}
```

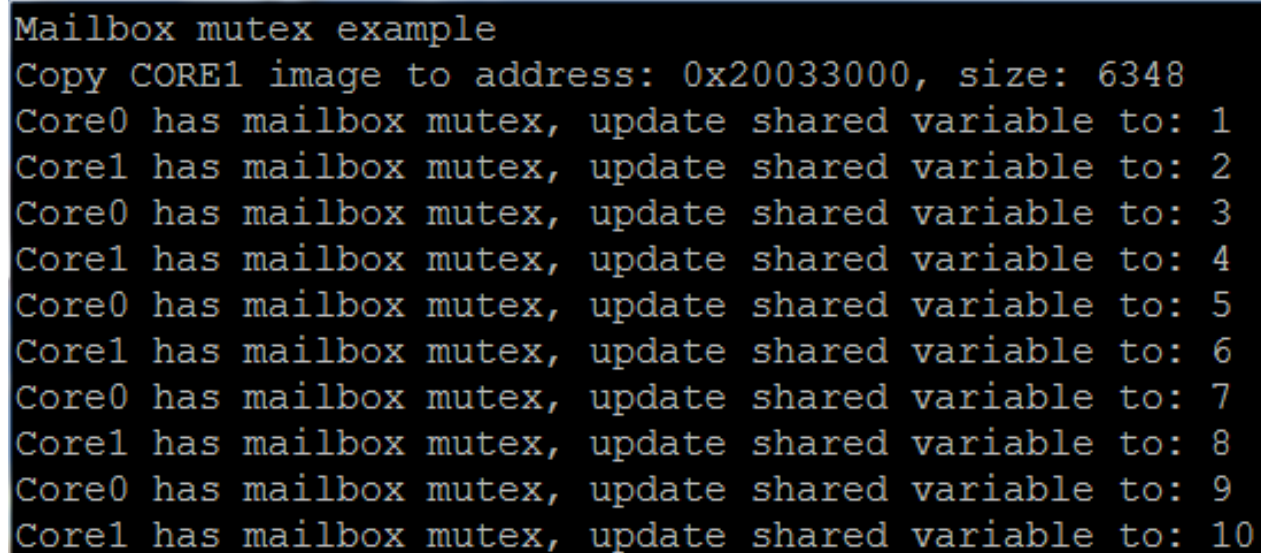
图 9. 主核 (core0) 上的主循环过程

结果可以在具有以下设置的串行终端窗口中显示：115200 波特率+ 8 个数据位+无奇偶校验+一个停止位+无流控制。

按照以下步骤准备并运行例程：

1. 构建从核 (core1) 的项目以生成二进制映像。
2. 使用内核 1 的二进制映像构建主核 (core0) 的项目。
3. 按以上设置打开串行终端。
4. 将内核 0 的映像下载到 LPCXpresso55s69 开发板上。
5. 按下板子上的复位按钮。

结果将在终端上打印，见 图 10.

A terminal window with a black background and yellow text. The text shows a sequence of operations between two cores (Core0 and Core1) using a mailbox mutex to update a shared variable. The output is as follows:

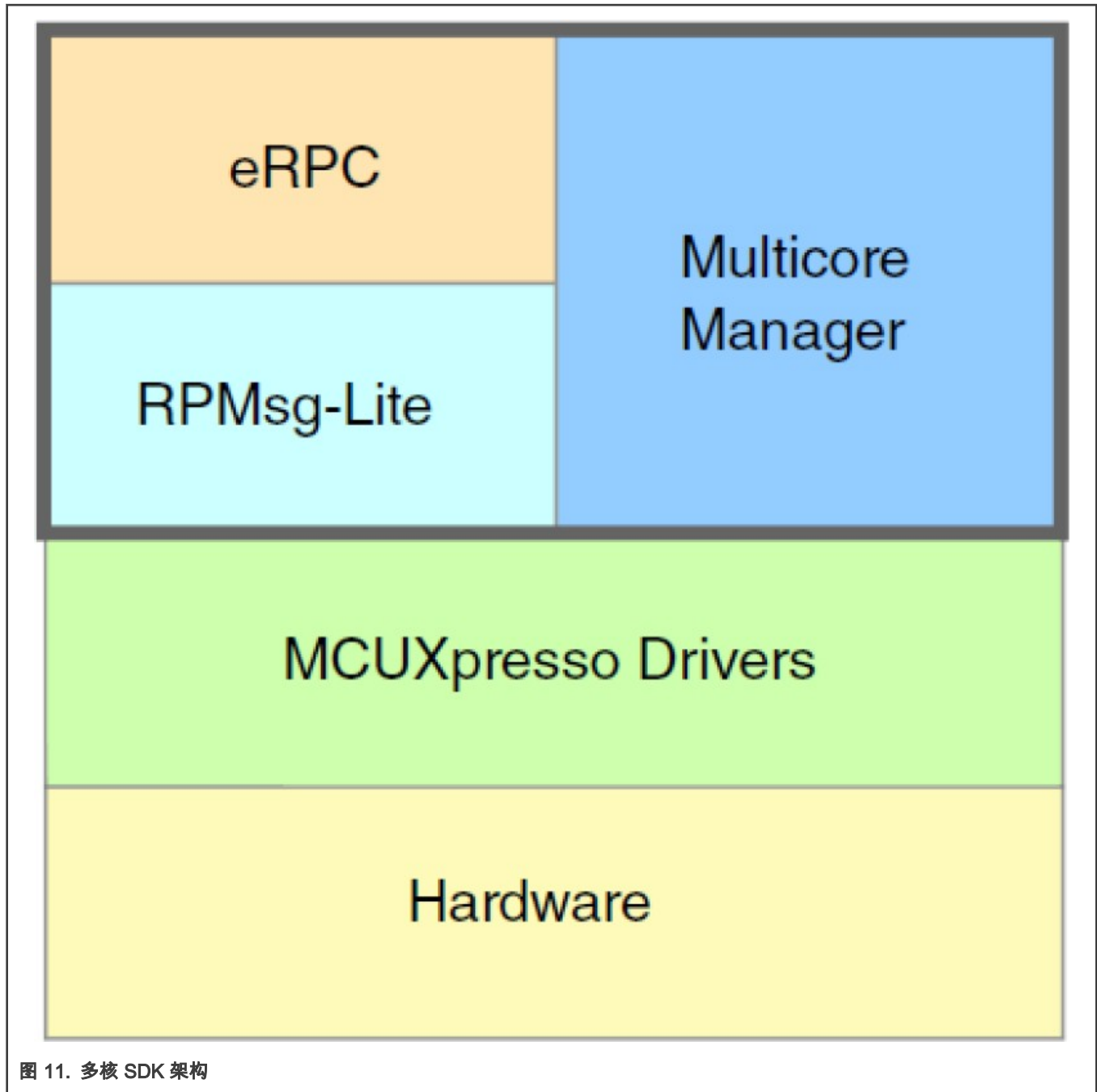
```
Mailbox mutex example
Copy CORE1 image to address: 0x20033000, size: 6348
Core0 has mailbox mutex, update shared variable to: 1
Core1 has mailbox mutex, update shared variable to: 2
Core0 has mailbox mutex, update shared variable to: 3
Core1 has mailbox mutex, update shared variable to: 4
Core0 has mailbox mutex, update shared variable to: 5
Core1 has mailbox mutex, update shared variable to: 6
Core0 has mailbox mutex, update shared variable to: 7
Core1 has mailbox mutex, update shared variable to: 8
Core0 has mailbox mutex, update shared variable to: 9
Core1 has mailbox mutex, update shared variable to: 10
```

图 10. 串行终端上的邮箱互斥体示例输出

## 4 多核 SDK

如上所述，带有 Mutex 模块的 CPU 间邮箱是一种用于双核通信的简单方法。在实践中有很多方法可以实现双核通信。恩智浦基于双核和多核架构，提供了一种称为多核 SDK 的更完整的多核开发套件，在 LPC55xx / LPC55Sxx SDK 软件包中包含了大量例程。可在以下路径中找到：boards\lpcpresso55s69\multicore\_examples. 多核 SDK 架构如 图 11 所示。





- **eRPC**: 嵌入式远程过程调用。这组 API 提供了调用另一个核上远程服务的功能。
- **RPMsg-Lite**: 远程处理器消息传递-轻量版。该功能库包含所有处理器通信。
- **Multicore Manager**: 这些 API 包含所有 CPU 内核信息和从核启动。

使用多核 SDK, 用户可以直接使用 API 实现多核应用程序通信, 而无需关注底层驱动程序。

## 5 总结

本应用笔记详细介绍了 LPC55xx/LPC55Sxx 支持的双核机制, 并基于 LPC5500 SDK 中的 mailbox\_mutex 驱动程序示例着重介绍了两个核如何进行通信。

LPC55xx/LPC55Sxx 中的双核是非对称架构。出厂时，一个内核设置为称为 CPU0/core0 的主核，另一个称为 CPU1/core1 的从核。当芯片启动时，主核可以工作而从核处于保持状态。从核需要由主核启动才工作。在此之前，通常应将从核映像加载到 SRAM 中，这是分配给它的执行空间。

LPC55xx/LPC55Sxx 提供了一种称为 Inter-CPU Mailbox 模块的简单方法用于双核通信。可以通过寄存器以 IRQ 触发发送非零的 32 位数据。该功能的使用非常灵活，完全取决于用户。在此模块中，支持互斥控件以实现共享资源访问的同步。

LPC5500 SDK 中有一个简单易懂的示例，用于演示具有邮箱和互斥功能的双核用法。本应用笔记对此进行了简要介绍。对于读者了解双核用法非常有帮助。

除了这种简单的双核通信机制外，在 LPC55xx/55Sxx SDK 中还打包了多核 SDK，具有丰富的例程，也可用于双核通信的开发。有关多核 SDK 的更多信息，请参考 <SDK installation directory>\middleware\multicore。

**How To Reach Us**

**Home Page:**

[nxp.com](http://nxp.com)

**Web Support:**

[nxp.com/support](http://nxp.com/support)

**Limited warranty and liability** — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

**Right to make changes** - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Security** — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at [PSIRT@nxp.com](mailto:PSIRT@nxp.com)) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE, VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. M, M Mobileye and other Mobileye trademarks or logos appearing herein are trademarks of Mobileye Vision Technologies Ltd. in the United States, the EU and/or other jurisdictions.

© NXP B.V. 2019-2021.

All rights reserved.

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

For sales office addresses, please send an email to: [salesaddresses@nxp.com](mailto:salesaddresses@nxp.com)

Date of release: 23 January 2019

Document identifier: AN12335

