# 第 3 章 存储器构成

## 目录

本章包括下列主题：

**注：** 本系列参考手册章节旨在作为器件数据手册的补充资料，并非适用于所有的 PIC32MX 器件，适用与否取决于具体的器件型号。

请查询具体器件数据手册中 **“存储器”** 章节开始处的注释，以查看本文档是否支持您当前使用的器件。

器件数据手册和系列参考手册的各章节均可从 Microchip 网站 http://www.microchip.com 下载。

## 3.1 简介

PIC32MX 单片机提供 4 GB 的统一虚拟存储地址空间。所有存储区 （包括程序存储器、数据存储器、SFR 和配置寄存器）都位于该地址空间中各自的唯一地址范围内。程序存储器和数据存储器可以选择划分为用户存储器和内核存储器。此外，数据存储器可以是可执行存储器，允许 PIC32MX 器件从数据存储器执行。

PIC32MX 存储器构成的主要特性包括：

- 32 位固有数据宽度
- 独立的用户模式地址空间和内核模式地址空间
- 灵活的程序闪存存储器分区
- 数据 RAM 可灵活地分为数据空间和程序空间
- 受保护代码的独立引导闪存
- 强大的总线异常处理功能，阻止代码跑飞
- 简单的存储器映射 （通过使用固定映射转换 （Fixed Mapping Translation， FMT）单元）
- 可高速缓存的地址区和不可高速缓存的地址区

## 3.2 控制寄存器

本节列出了用于为数据和代码设置 RAM 和闪存分区的特殊功能寄存器 （Special Function Register， SFR）（对于用户模式和内核模式）。

以下是可用 SFR 的列表：

- BMXCON：配置寄存器
- BMXxxxBA：存储器分区基址寄存器
- BMXDRMSZ：数据 RAM 大小寄存器
- BMXPFMSZ：程序闪存大小寄存器
- BMXBOOTSZ：引导闪存大小寄存器

### 3.2.1 BMXCON 寄存器

该寄存器用于配置 DMA 访问的程序闪存高速缓存功能、总线错误异常、数据 RAM 等待状态和仲裁模式。

### 3.2.2 BMXxxxBA 寄存器

这些寄存器用于配置内核模式、用户模式数据和用户模式程序空间在 RAM 中的相对基址。

### 3.2.3 BMXDRMSZ 寄存器

该只读寄存器用于标识数据 RAM 的大小 （以字节为单位）。

### 3.2.4 BMXPFMSZ 寄存器

该只读寄存器用于标识程序闪存存储器的大小 （以字节为单位）。

### 3.2.5 BMXBOOTSZ 寄存器

该只读寄存器用于标识引导闪存存储器的大小 （以字节为单位）。

表 3-1 简要汇总了所有与存储器构成相关的寄存器。该汇总表之后列出了相应的寄存器，并且每个寄存器均附有详细的说明。

**表 3-1：　存储器构成 SFR 汇总**

| 地址偏移 | 名称 | 位范围 | Bit 31/23/15/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x2000 | BMXCON(1,2,3) | 31:24 | — | — | — | — | — | BMXCHEDMA | — | — |
| | | 23:16 | — | — | — | BMXERRIXI | BMXERRICD | BMXERRDMA | BMXERRDS | BMXERRIS |
| | | 15:8 | — | — | — | — | — | — | — | — |
| | | 7:0 | — | BMXWSDRM | — | — | — | BMXARB<2:0> | | |
| 0x2010 | BMXDKPBA(1,2,3) | 31:24 | — | — | — | — | — | — | — | — |
| | | 23:16 | — | — | — | — | — | — | — | — |
| | | 15:8 | BMXDKPBA<15:8> | | | | | | | |
| | | 7:0 | BMXDKPBA<7:0> | | | | | | | |
| 0x2020 | BMXDUDBA(1,2,3) | 31:24 | — | — | — | — | — | — | — | — |
| | | 23:16 | — | — | — | — | — | — | — | — |
| | | 15:8 | BMXDUDBA<15:8> | | | | | | | |
| | | 7:0 | BMXDUDBA<7:0> | | | | | | | |
| 0x2030 | BMXDUPBA(1,2,3) | 31:24 | — | — | — | — | — | — | — | — |
| | | 23:16 | — | — | — | — | — | — | — | — |
| | | 15:8 | BMXDUPBA<15:8> | | | | | | | |
| | | 7:0 | BMXDUPBA<7:0> | | | | | | | |
| 0x2040 | BMXDRMSZ | 31:24 | BMXDRMSZ<31:24> | | | | | | | |
| | | 23:16 | BMXDRMSZ<23:16> | | | | | | | |
| | | 15:8 | BMXDRMSZ<15:8> | | | | | | | |
| | | 7:0 | BMXDRMSZ<7:0> | | | | | | | |
| 0x2050 | BMXPUPBA(1,2,3) | 31:24 | — | — | — | — | — | — | — | — |
| | | 23:16 | — | — | — | — | BMXPUPBA<19:16> | | | |
| | | 15:8 | BMXPUPBA<15:8> | | | | | | | |
| | | 7:0 | BMXPUPBA<7:0> | | | | | | | |
| 0x2060 | BMXPFMSZ | 31:24 | BMXPFMSZ<31:24> | | | | | | | |
| | | 23:16 | BMXPFMSZ<23:16> | | | | | | | |
| | | 15:8 | BMXPFMSZ<15:8> | | | | | | | |
| | | 7:0 | BMXPFMSZ<7:0> | | | | | | | |
| 0x2070 | BMXBOOTSZ | 31:24 | BMXBOOTSZ<31:24> | | | | | | | |
| | | 23:16 | BMXBOOTSZ<23:16> | | | | | | | |
| | | 15:8 | BMXBOOTSZ<15:8> | | | | | | | |
| | | 7:0 | BMXBOOTSZ<7:0> | | | | | | | |

**图注：**　— = 未实现，读为 0。地址偏移值以十六进制显示。

**注　1：** 该寄存器具有关联的清零寄存器，位于 0x4 字节偏移处。这些清零寄存器的命名方式是在关联寄存器的名称末尾附加 CLR（例如，BMXCONCLR）。向清零寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器，位于 0x8 字节偏移处。这些置 1 寄存器的命名方式是在关联寄存器的名称末尾附加 SET（例如，BMXCONSET）。向置 1 寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器，位于 0xC 字节偏移处。这些取反寄存器的命名方式是在关联寄存器的名称末尾附加 INV（例如，BMXCONINV）。向取反寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**寄存器 3-1： BMXCON：总线矩阵配置寄存器 (1,2,3)**

| r-x | r-x | r-x | r-x | r-x | R/W-0 | r-x | r-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | BMXCHEDMA | — | — |
| bit 31 | | | | | | | bit 24 |

| r-x | r-x | r-x | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | BMXERRIXI | BMXERRICD | BMXERRDMA | BMXERRDS | BMXERRIS |
| bit 23 | | | | | | | bit 16 |

| r-x | r-x | r-x | r-x | r-x | r-x | r-x | r-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| r-x | R/W-1 | r-x | r-x | r-x | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | BMXWSDRM | — | — | — | BMXARB<2:0> | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | P = 可编程位 | r = 保留位 |
| U = 未实现位 | -n = POR 时的值：（0， 1， x = 未知） | | |

bit 31-27 **保留：**写入 0；忽略读操作

bit 26 **BMXCHEDMA：**DMA 访问 BMX PFM 高速缓存功能位

1 = 对 DMA 访问使能程序 （数据）闪存存储器高速缓存功能 （要求高速缓存使能数据高速缓存）
0 = 对 DMA 访问禁止程序 （数据）闪存存储器高速缓存功能
（命中时仍然从高速缓存进行读取，但未命中时将不更新高速缓存）

bit 25-21 **保留：**写入 0；忽略读操作

bit 20 **BMXERRIXI：**IXI 总线错误使能位

1 = 对于从 IXI 共用总线启动的非映射地址访问使能总线错误异常
0 = 对于从 IXI 共用总线启动的非映射地址访问禁止总线错误异常

bit 19 **BMXERRICD：**ICD 调试单元总线错误使能位

1 = 对于从 ICD 启动的非映射地址访问使能总线错误异常
0 = 对于从 ICD 启动的非映射地址访问禁止总线错误异常

bit 18 **BMXERRDMA：**DMA 总线错误位

1 = 对于从 DMA 启动的非映射地址访问使能总线错误异常
0 = 对于从 DMA 启动的非映射地址访问禁止总线错误异常

bit 17 **BMXERRDS：**CPU 数据访问总线错误位 （在 DEBUG （调试）模式下禁止）

1 = 对于从 CPU 数据访问启动的非映射地址访问使能总线错误异常
0 = 对于从 CPU 数据访问启动的非映射地址访问禁止总线错误异常

bit 16 **BMXERRIS：**CPU 指令访问总线错误位 （在 DEBUG （调试）模式下禁止）

1 = 对于从 CPU 指令访问启动的非映射地址访问使能总线错误异常
0 = 对于从 CPU 指令访问启动的非映射地址访问禁止总线错误异常

**注 1：** 该寄存器具有关联的清零寄存器 （BMXCONCLR），位于 0x4 字节偏移处。向清零寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器 （BMXCONSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器 （BMXCONINV），位于 0xC 字节偏移处。向取反寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**寄存器 3-1：　BMXCON：总线矩阵配置寄存器 (1,2,3) （续）**

| | |
|---|---|
| bit 15-7 | **保留：**写入 0；忽略读操作 |
| bit 6 | **BMXWSDRM：**数据 RAM 的 CPU 指令或数据访问等待状态位<br>1 = 来自 CPU 的数据 RAM 访问具有 1 个等待状态用于建立地址<br>0 = 来自 CPU 的数据 RAM 访问具有 0 个等待状态用于建立地址 |
| bit 5-3 | **保留：**写入 0；忽略读操作 |
| bit 2-0 | **BMXARB<2:0>：**总线矩阵仲裁模式位<br>111...011 = 保留 （使用这些配置模式将产生未定义的行为）<br>010 = 仲裁模式 2<br>001 = 仲裁模式 1 （默认值）<br>000 = 仲裁模式 0 |

**注　1：**该寄存器具有关联的清零寄存器（BMXCONCLR），位于 0x4 字节偏移处。向清零寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：**该寄存器具有关联的置 1 寄存器（BMXCONSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：**该寄存器具有关联的取反寄存器 （BMXCONINV），位于 0xC 字节偏移处。向取反寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**寄存器 3-2： BMXDKPBA：数据 RAM 内核程序基址寄存器 (1,2,3,4,5)**

| r-x | r-x | r-x | r-x | r-x | r-x | r-x | r-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 31 | | | | | | | bit 24 |

| r-x | r-x | r-x | r-x | r-x | r-x | r-x | r-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| BMXDKPBA<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| BMXDKPBA<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | P = 可编程位 | r = 保留位 |
| U = 未实现位 | -n = POR 时的值：（0， 1， x = 未知） | | |

bit 31-16 **保留：**写入 0；忽略读操作

bit 15-11 **BMXDKPBA<15:11>：**DRM 内核程序基址位
非零时，该值选择内核程序空间在 RAM 中的相对基址

bit 10-0 **BMXDKPBA<10:0>：**只读位
值总是为 0，这会强制设置 2 KB 的递增量

**注 1：** 该寄存器具有关联的清零寄存器（BMXDKPBACLR），位于 0x4 字节偏移处。向清零寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器（BMXDKPBASET），位于 0x8 字节偏移处。向置 1 寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（BMXDKPBAINV），位于 0xC 字节偏移处。向取反寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 复位时，该寄存器中的值被强制为 0，从而使整个 RAM 分配为内核模式数据使用。

**5：** 该寄存器中的值必须小于或等于 BMXDRMSZ。

**寄存器 3-3： BMXDUDBA：数据 RAM 用户数据基址寄存器 (1,2,3,4,5)**

| r-x | r-x | r-x | r-x | r-x | r-x | r-x | r-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 31 | | | | | | | bit 24 |

| r-x | r-x | r-x | r-x | r-x | r-x | r-x | r-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| BMXDUDBA<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| BMXDUDBA<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **图注：** | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | P = 可编程位 | r = 保留位 |
| U = 未实现位 | -n = POR 时的值：（0， 1， x = 未知） | | |

bit 31-16 **保留：**写入 0；忽略读操作

bit 15-11 **BMXDUDBA<15:11>：**DRM 用户数据基址位
非零时，该值选择用户模式数据空间在 RAM 中的相对基址
**注：** 如果非零，该值必须大于 BMXDKPBA。

bit 10-0 **BMXDUDBA<10:0>：**只读位
值总是为 0，这会强制设置 2 KB 的递增量

**注 1：** 该寄存器具有关联的清零寄存器（BMXDUDBACLR），位于 0x4 字节偏移处。向清零寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器（BMXDUDBASET），位于 0x8 字节偏移处。向置 1 寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（BMXDUDBAINV），位于 0xC 字节偏移处。向取反寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 复位时，该寄存器中的值被强制为 0，从而使整个 RAM 分配为内核模式数据使用。

**5：** 该寄存器中的值必须小于或等于 BMXDRMSZ。

**寄存器 3-4： BMXDUPBA：数据 RAM 用户程序基址寄存器 (1,2,3,4,5)**

| r-x | r-x | r-x | r-x | r-x | r-x | r-x | r-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 31 | | | | | | | bit 24 |

| r-x | r-x | r-x | r-x | r-x | r-x | r-x | r-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| BMXDUPBA<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| sBMXDUPBA<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | P = 可编程位 | r = 保留位 |
| U = 未实现位 | -n = POR 时的值：（0， 1， x = 未知） | | |

bit 31-16 **保留**：写入 0；忽略读操作

bit 15-11 **BMXDUPBA<15:11>**：DRM 用户程序基址位
非零时，该值选择用户模式程序空间在 RAM 中的相对基址

**注：** 如果非零， BMXDUPBA 必须大于 BMXDUDBA。

bit 10-0 **BMXDUPBA<10:0>**：只读位
值总是为 0，这会强制设置 2 KB 的递增量

**注 1：** 该寄存器具有关联的清零寄存器 （BMXDUPBACLR），位于 0x4 字节偏移处。向清零寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器 （BMXDUPBASET），位于 0x8 字节偏移处。向置 1 寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（BMXDUPBAINV），位于 0xC 字节偏移处。向取反寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 复位时，该寄存器中的值被强制为 0，从而使整个 RAM 分配为内核模式数据使用。

**5：** 该寄存器中的值必须小于或等于 BMXDRMSZ。

**寄存器 3-5： BMXDRMSZ：数据 RAM 大小寄存器**

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXDRMSZ<31:24> | | | | | | | |
| bit 31 | | | | | | | bit 24 |

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXDRMSZ<23:16> | | | | | | | |
| bit 23 | | | | | | | bit 16 |

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXDRMSZ<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXDRMSZ<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **图注：** | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | P = 可编程位 | r = 保留位 |
| U = 未实现位 | -n = POR 时的值：（0， 1， x = 未知） | | |

bit 31-0 **BMXDRMSZ<31:0>：**数据 RAM 存储器 （DRM）大小位
以字节为单位指示数据 RAM 大小的静态值：
0x00002000 = 器件具有 8 KB 的 RAM
0x00004000 = 器件具有 16 KB 的 RAM
0x00008000 = 器件具有 32 KB 的 RAM
0x00010000 = 器件具有 64 KB 的 RAM
0x00020000 = 器件具有 128 KB 的 RAM

**寄存器 3-6： BMXPUPBA：程序闪存 （PFM）用户程序基址寄存器 (1,2,3,4,5)**

| r-x | r-x | r-x | r-x | r-x | r-x | r-x | r-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 31 | | | | | | | bit 24 |

| r-x | r-x | r-x | r-x | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | BMXPUPBA<19:16> | | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| BMXPUPBA<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| BMXPUPBA<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | P = 可编程位 | r = 保留位 |
| U = 未实现位 | -n = POR 时的值：（0， 1， x = 未知） | | |

bit 31-20 **保留：**写入 0；忽略读操作

bit 19-11 **BMXPUPBA<19:11>：**程序闪存 （PFM）用户程序基址位

bit 10-0 **BMXPUPBA<10:0>：**只读位
值总是为 0，这会强制设置 2 KB 的递增量

**注 1：** 该寄存器具有关联的清零寄存器（BMXPUPBACLR），位于 0x4 字节偏移处。向清零寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器（BMXPUPPBASET），位于 0x8 字节偏移处。向置 1 寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（BMXPUPBAINV），位于 0xC 字节偏移处。向取反寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 复位时，该寄存器中的值被强制为 0，从而使整个 RAM 分配为内核模式程序使用。

**5：** 该寄存器中的值必须小于或等于 BMXPFMSZ。

**寄存器 3-7： BMXPFMSZ：程序闪存 （PFM）大小寄存器**

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXPFMSZ<31:24> | | | | | | | |
| bit 31 | | | | | | | bit 24 |

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXPFMSZ<23:16> | | | | | | | |
| bit 23 | | | | | | | bit 16 |

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXPFMSZ<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXPFMSZ<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **图注：** | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | P = 可编程位 | r = 保留位 |
| U = 未实现位 | -n = POR 时的值：（0， 1， x = 未知） | | |

bit 31-0 **BMXPFMSZ<31:0>：**程序闪存存储器 （PFM）大小位
以字节为单位指示 PFM 大小的静态值：
0x00008000 = 器件具有 32 KB 的闪存
0x00010000 = 器件具有 64 KB 的闪存
0x00020000 = 器件具有 128 KB 的闪存
0x00040000 = 器件具有 256 KB 的闪存
0x00080000 = 器件具有 512 KB 的闪存

**寄存器 3-8： BMXBOOTSZ：引导闪存 （IFM）大小寄存器**

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXBOOTSZ<31:24> | | | | | | | |
| bit 31 | | | | | | | bit 24 |

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXBOOTSZ<23:16> | | | | | | | |
| bit 23 | | | | | | | bit 16 |

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXBOOTSZ<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| BMXBOOTSZ<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

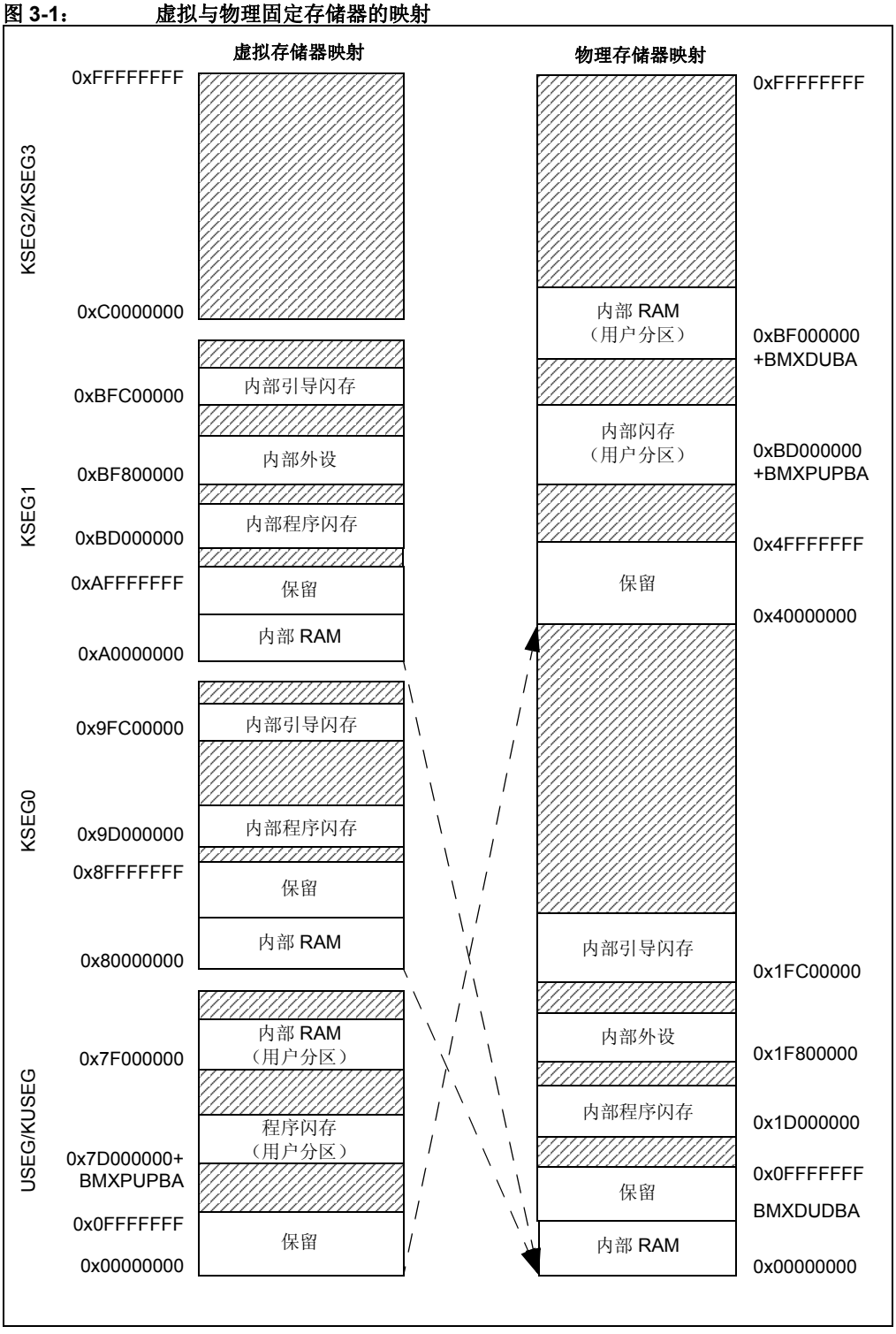
| **图注：** | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | P = 可编程位 | r = 保留位 |
| U = 未实现位 | -n = POR 时的值：（0， 1， x = 未知） | | |

bit 31-0 **BMXBOOTSZ<31:0>：**引导闪存存储器 （BFM）大小位
以字节为单位指示引导 PFM 大小的静态值：
0x00003000 = 器件具有 12 KB 的引导闪存

## 3.3 PIC32MX 存储器布局

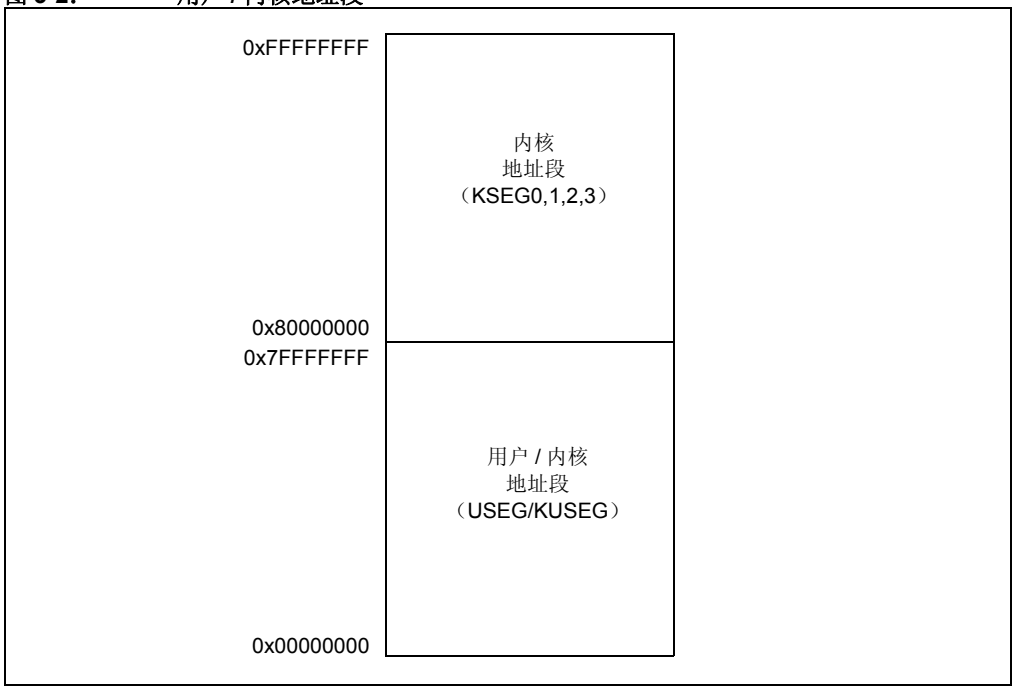
PIC32MX 单片机实现了两个地址空间：虚拟地址空间和物理地址空间。所有硬件资源 （例如程序存储器、数据存储器和外设）都位于各自相关的物理地址范围内。虚拟地址专供 CPU 取指和执行指令使用。物理地址供外设（例如不通过 CPU，独立访问存储器的 DMA 和闪存控制器）使用。

**图 3-1： 虚拟与物理固定存储器的映射**



整个 4 GB 虚拟地址空间分为两个基本区域：用户空间和内核空间。来自用户模式地址段的低 2 GB 空间称为 USEG/KUSEG。用户模式应用程序必须驻留在 USEG 段，并在其中执行。USEG 段也可供所有内核模式应用程序使用，所以它也称为 KUSEG——指示它可同时用于用户模式和内核模式。在用户模式下工作时，必须配置总线矩阵，使部分闪存和数据存储器在 USEG/KUSEG 段中可用。更多详细信息，请参见**第 3.4 节 “PIC32MX 地址映射”**。

**图 3-2： 用户 / 内核地址段**




虚拟地址空间的高 2 GB 构成仅限内核模式使用的空间。内核空间分为 4 个各为 512 MB 的地址段：KSEG0、KSEG1、KSEG2 和 KSEG3。仅内核模式应用程序可以访问内核存储空间。内核空间包括所有外设寄存器。因此，仅内核模式应用程序可以监视和操作外设。只有 KSEG0 和 KSEG1 段指向真正的存储器资源。如 MIPS 文档所介绍，地址段 KSEG2 可供 EJTAG 探针调试器使用 （请参见 EJTAG 规范）。 PIC32MX 仅使用 KSEG0 和 KSEG1 段。可通过 KSEG0 或 KSEG1 访问引导闪存存储器 （BFM）、程序闪存存储器 （PFM）、数据 RAM 存储器 （DRM）和外设 SFR。

固定映射转换 （FMT）单元可以将存储器地址段转换为相应的物理地址区域。图 3-1 显示了 PIC32MX 内核在虚拟地址空间和物理地址空间之间实现的固定映射方案。虚拟存储器地址段也可以进行高速缓存，前提是器件上提供了高速缓存模块。请注意， KSEG1 存储器地址段是不可高速缓存的，而 KSEG0 和 USEG/KUSEG 是可高速缓存的。

存储器地址段的映射取决于 CPU 错误级别 （通过 CPU 状态寄存器中的 ERL 位设置）。错误级别由 CPU 在发生复位、软复位或 NMI 时设置 （ERL = 1）。在该模式下，处理器运行于内核模式， USEG/KUSEG 被视为非映射和非高速缓存区，图 3-1 中的映射不适用。该模式可与使用基于 TLB 的 MMU 的其他 MIPS 处理器内核保持兼容。 C 语言启动代码会将 ERL 位清零，从而当应用程序软件启动时，会看到虚拟存储器到物理存储器的正确映射，如图 3-1 所示。

地址段 KSEG0 和 KSEG1 总是转换为物理地址 0x0。通过这种转换安排， CPU 可以通过两个独立的虚拟地址访问相同的物理地址：一个是通过 KSEG0，另一个是通过 KSEG1。因此，应用程序可以选择以高速缓存或非高速缓存的方式执行同一段代码。更多详细信息，请参见**第 4 章 “预取高速缓存模块”**（DS61119）。只有通过 KSEG1 段才可访问片上外设（非高速缓存访问）。

## 3.4 PIC32MX 地址映射

程序闪存存储器分为内核分区和用户分区。内核程序闪存空间从物理地址 0x1D000000 处开始，而用户程序闪存空间则从物理地址 0xBD000000 + BMXPUDBA 寄存器值处开始。类似地，内部 RAM 也分为内核分区和用户分区。内核 RAM 空间从物理地址 0x00000000 处开始，而用户 RAM 空间则从物理地址 0xBF000000 + BMXDUDBA 寄存器值处开始。默认情况下，全部闪存和 RAM 仅映射到内核模式应用程序。

请注意，BMXxxxBA 寄存器设置必须与目标软件应用程序的存储器模型匹配。如果链接的代码与寄存器值不匹配，则程序可能无法运行，并可能在启动时产生总线错误异常。

| | |
|---|---|
| **注：** | 程序闪存存储器不能通过其地址映射进行写操作。对 PFM 地址范围的写操作会导致总线错误异常。 |

### 3.4.1 虚拟地址与物理地址的转换计算

要将内核地址 （KSEG0 或 KSEG1）转换为物理地址，需要使用 0x1FFFFFFF 对虚拟地址执行“按位与”运算：

- 物理地址 = 虚拟地址 & 0x1FFFFFFF

要将物理地址转换为 KSEG0 虚拟地址，需要使用 0x80000000 对物理地址执行“按位或”运算：

- KSEG0 虚拟地址 = 物理地址 | 0x80000000

要将物理地址转换为 KSEG1 虚拟地址，需要使用 0xA0000000 对物理地址执行“按位或”运算：

- KSEG1 虚拟地址 = 物理地址 | 0xA0000000

要从 KSEG0 虚拟地址转换为 KSEG1 虚拟地址，需要使用 0x20000000 对 KSEG0 虚拟地址执行“按位或”运算：

- KSEG1 虚拟地址 = KSEG0 虚拟地址 | 0x20000000

表 3-2： PIC32MX 地址映射

| 存储器类型 | | 虚拟地址 | | 物理地址 | | 大小（以字节为单位） |
|---|---|---|---|---|---|---|
| | | 起始地址 | 结束地址 | 起始地址 | 结束地址 | 计算 |
| 内核地址空间 | 引导闪存 | 0xBFC00000 | 0xBFC02FFF | 0x1FC00000 | 0x1FC02FFF | 12 KB |
| | 外设 | 0xBF800000 | 0xBF8FFFFF | 0x1F800000 | 0x1F8FFFFF | 4 KB |
| | **KSEG1** 程序闪存 (1,3) | 0xBD000000 | 0xBD000000 + BMXPUPBA - 1 | 0x1D000000 | 0x1D00000 + BMXPUPBA - 1 | BMXPUPBA |
| | **KSEG1** 程序 **RAM**(6) | 0xA0000000 + BMXDKPBA | 0xA0000000 + BMXDUDBA - 1 | 0x00000000 + BMXDKPBA | 0x00000000 + BMXDUDBA - 1 | BMXDUDBA - BMXDKPBA |
| | **KSEG1** 数据 **RAM**(6) | 0xA0000000 | 0xA0000000 + BMXDKPBA - 1 | 0x00000000 | 0x00000000 + BMXDKPBA - 1 | BMXPUPBA |
| | **KSEG0** 程序闪存 (2,5) | 0x9D000000 | 0x9D000000 + BMXPUPBA - 1 | 0x1D000000 | 0x1D000000 + BMXPUPBA - 1 | BMXPUPBA |
| | **KSEG0** 程序 **RAM**(6) | 0x80000000 + BMXDKPBA | 0x80000000 + BMXDUDBA - 1 | 0x00000000 + BMXDKPBA | 0x00000000 + BMXDUDBA - 1 | BMXDUDBA - BMXDKPBA |
| | **KSEG0** 数据 **RAM**(6) | 0x80000000 | 0x80000000 + BMXDKPBA - 1 | 0x00000000 | 0x00000000 + BMXDKPBA - 1 | BMXDKPBA |
| 存储器类型 | | 虚拟地址 | | 物理地址 | | 大小（以字节为单位） |
| | | 起始地址 | 结束地址 | 起始地址 | 结束地址 | 计算 |
| 用户地址空间 | **USEG/KSEG** 程序 **RAM**(6) | 0x7F000000 + BMXDUPBA | 0x7F000000 + BMXDRMSZ -1(3) | 0xBF000000 + BMXDUPBA | 0xBF000000 + BMXDRMSZ(3) - 1 | BMXDRMSZ(3) - BMXDUPBA |
| | **USEG/KSEG** 数据 **RAM**(6) | 0x7F000000 + BMXDUDBA | 0x7F000000 + BMXDUPBA - 1 | 0xBF000000 + BMXDUDBA | 0xBF000000 + BMXDUPBA - 1 | BMXDUPBA - BMXDUDBA |
| | **USEG/KSEG** 程序闪存 (6) | 0x7D000000 + BMXPUPBA | 0x7D000000 + BMXPFMSZ(4) - 1 | 0xBD000000 + BMXPUPBA | 0xBF000000 + BMXPFMSZ(4) - 1 | BMXPFMSZ(4) - BMXPUPBA |

注 **1：** 不可高速缓存范围中的程序闪存虚拟地址 （KSEG1）。

**2：** 可高速缓存和可预取范围中的程序闪存虚拟地址 （KSEG0）。

**3：** RAM 大小会因 PIC32MX 器件型号而异。

**4：** 闪存大小会因 PIC32MX 器件型号而异。

**5：** 当 BMXPUPBA 寄存器为 0 时，所有程序闪存均分配给内核模式的程序使用。这是复位时的默认值。

**6：** 当 BMXDUDBA、BMXDUPBA 或 BMXDKPBA 寄存器为 0 时，所有 RAM 均分配给内核模式程序使用。这是复位时的默认值。

### 3.4.2 程序闪存存储器分区

程序闪存存储器可以划分为用户和内核模式程序，如图 3-1 所示。

在复位时，用户模式分区不存在 （BMXPUPBA 初始化为 0）。整个程序闪存存储器映射到从虚拟地址 KSEG1：0xBD000000 （或 KSEG0：0x9D000000）开始的内核模式程序空间。要为用户模式程序设置分区，请按以下方式初始化 BMXPUPBA：

- BMXPUPBA = BMXPFMSZ – USER_FLASH_PGM_SZ

USER_FLASH_PGM_SZ 是用户模式程序的分区大小。BMXPFMSZ 是包含程序闪存存储器总容量的矩阵寄存器。

示例：

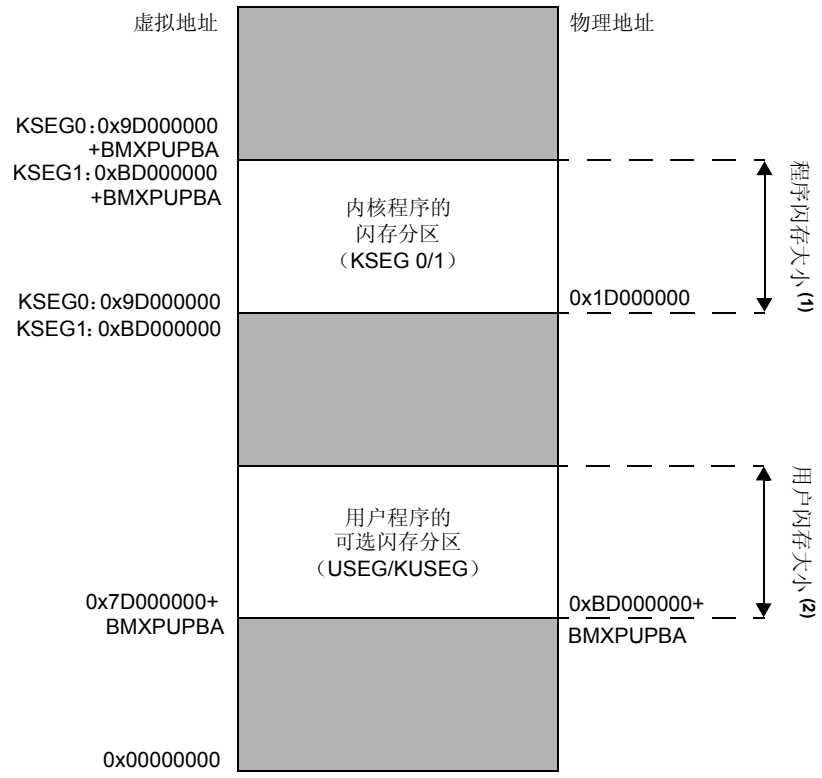
- 假设 PIC32MX 器件具有 512 KB 的闪存， BMXPFMSZ 将包含 0x00080000。
- 要创建 20 KB （0x5000）的用户闪存程序分区：
- BMXPUPBA = 0x80000 - 0x5000 = 0x7B000

用户闪存的大小将为 20K，留给内核闪存的大小将为 512 KB – 20 KB = 492 KB。

用户闪存分区将从 0x7D07B000 扩展至 0x7D07FFFF （虚拟地址）。

内核模式分区总是从 KSEG1：0xBD000000 或 KSEG0：0x9D000000 处开始。在以上示例中，内核分区将从 0xBD000000 扩展至 0xBD07AFFF （大小为 492 KB）。

**图 3-3：　　闪存分区**




**注　1：** 内核闪存大小 = BMXPUPBA

**2：** 用户闪存大小 = BMXPFMSZ-BMXPUPBA

**3：** 如果 BMXPUPBA 为 0，那么：
内核闪存大小 = BMXPFMSZ （即，全部闪存）
用户闪存大小 = 0

### 3.4.3　RAM 分区

RAM 存储器可以分为 4 个分区。它们是：

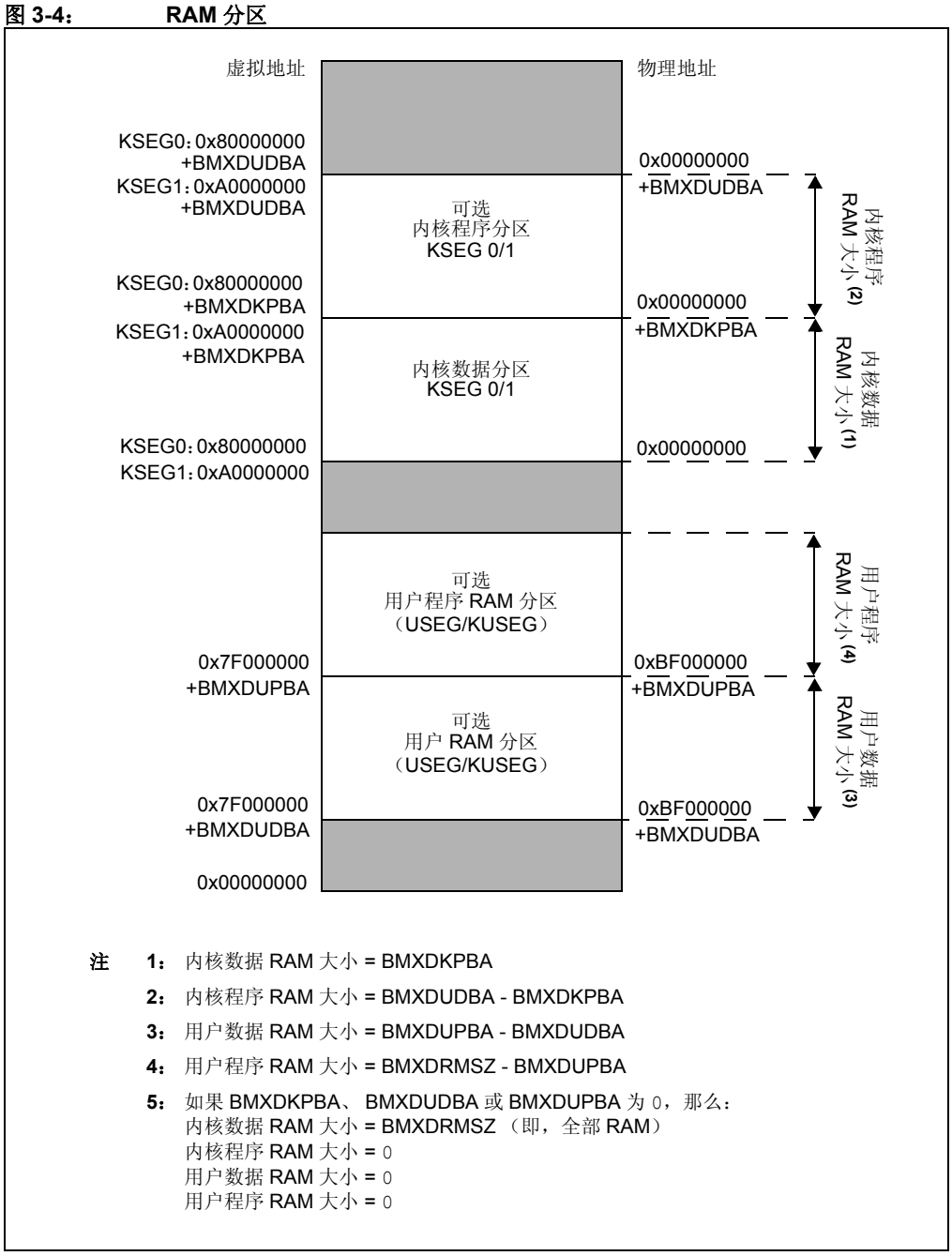
- 内核数据
- 内核程序
- 用户数据
- 用户程序

为了从数据 RAM 中执行代码，必须定义内核或用户程序分区。在上电复位 （Power-on Reset，POR）时，整个数据 RAM 分配给内核数据分区。该分区总是从数据 RAM 基址处开始。详情请参见图 3-4。

**注　1：** 为了对 RAM 进行正确分区，必须设定以下所有寄存器：BMXDKPBA、BMXDUDBA 和 BMXDUPBA。

**2：** 可用 RAM 的大小由 BMXDRMSZ 寄存器指定。

**图 3-4：　　RAM 分区**




**注　1：** 内核数据 RAM 大小 = BMXDKPBA

**2：** 内核程序 RAM 大小 = BMXDUDBA - BMXDKPBA

**3：** 用户数据 RAM 大小 = BMXDUPBA - BMXDUDBA

**4：** 用户程序 RAM 大小 = BMXDRMSZ - BMXDUPBA

**5：** 如果 BMXDKPBA、 BMXDUDBA 或 BMXDUPBA 为 0，那么：
内核数据 RAM 大小 = BMXDRMSZ （即，全部 RAM）
内核程序 RAM 大小 = 0
用户数据 RAM 大小 = 0
用户程序 RAM 大小 = 0

#### 3.4.3.1 内核数据 RAM 分区

内核数据 RAM 分区位于虚拟地址 KSEG0：0x80000000 和 KSEG1：0xA0000000 处。它总为可用，不能禁止。

请注意，如果 BMXDKPBA、BMXDUDBA 或 BMXDUPBA 寄存器中的任一寄存器为 0，那么整个 RAM 将分配给内核数据 RAM （即，内核数据 RAM 分区的大小由 BMXDRMSZ 寄存器值指定；请参见图 3-5）。否则，内核数据 RAM 分区的大小由 BMXDKPBA 寄存器值指定 （见图 3-6）。

内核数据 RAM 分区在复位时就存在，占用所有可用 RAM，因为 BMXDKPBA、 BMXDUDBA 和 BMXDUPBA 寄存器在每次复位时总是默认设为 0。

**图 3-5： 当 BMXDKPBA、 BMXDUDBA 或 BMXDUPBA = 0 时的 RAM 分区**

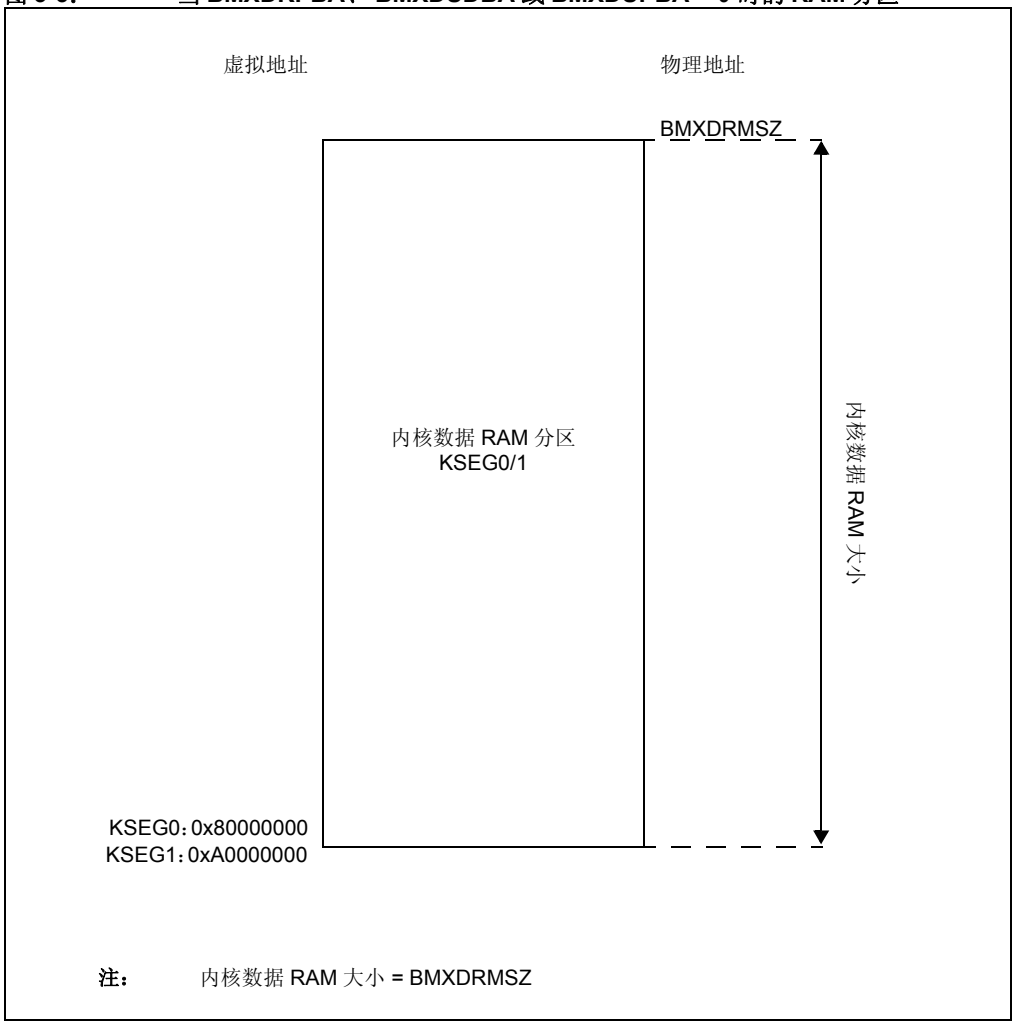
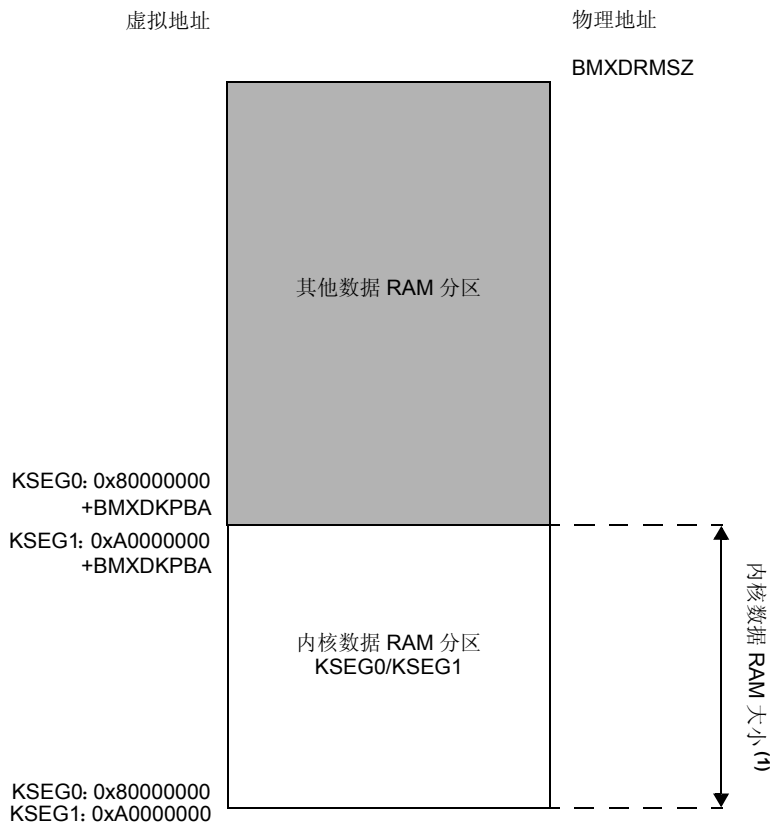


**图 3-6：　　内核数据 RAM 分区**




**注　1：** 内核数据 RAM 大小 = BMXDKPBA。

**2：** 寄存器 BMXDKPBA、 BMXDUDBA 或 BMXDUPBA 均不为 0。

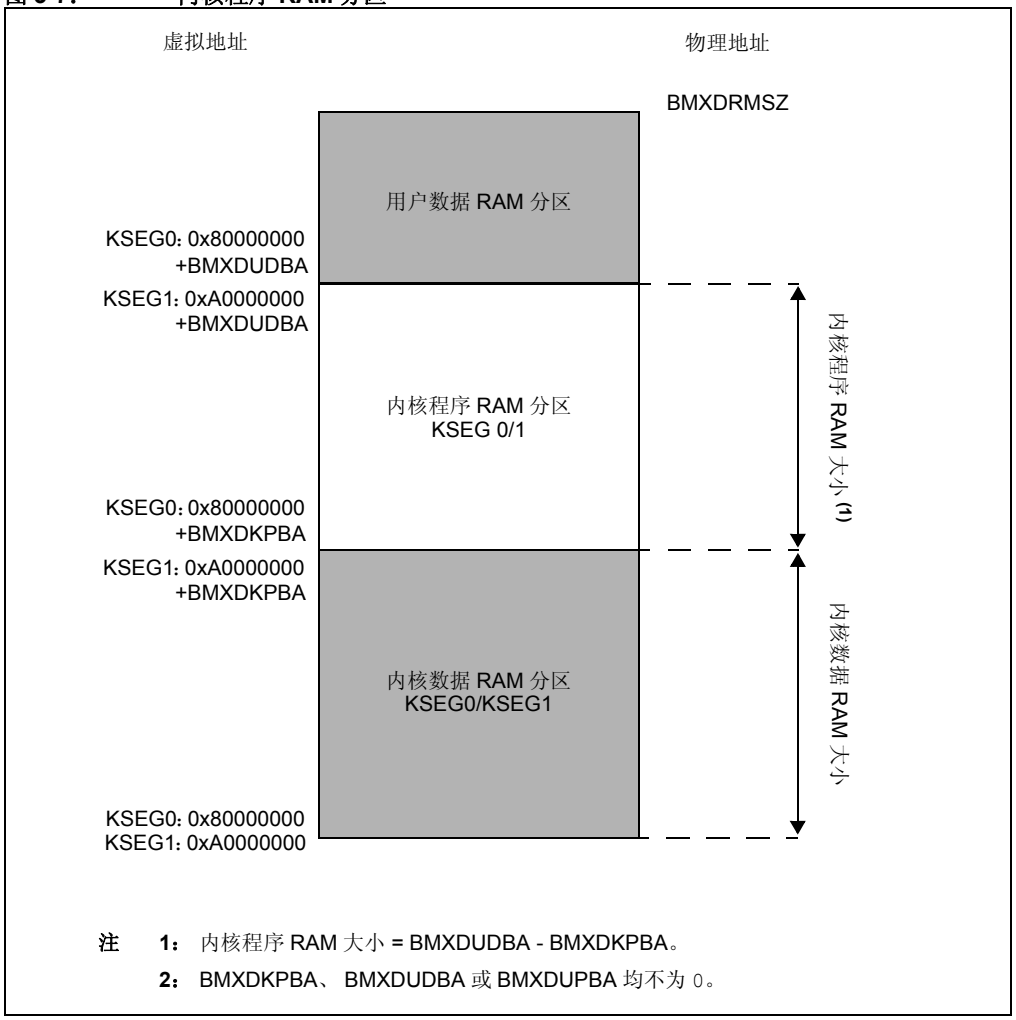
#### 3.4.3.2 内核程序 RAM 分区

如果需要在内核模式下从数据 RAM 中执行代码，则需要内核程序 RAM 分区。

该分区从 KSEG0：0x80000000 + BMXDKPBA （KSEG1：0xA0000000 + BMXDKPBA）处开始，其大小由 BMXDUDBA - BMXDKPBA 指定。请参见图 3-7。

内核程序 RAM 分区在复位时不存在，因为 BMXDKPBA 和 BMXDUDBA 寄存器在复位时默认设为 0。

**图 3-7： 内核程序 RAM 分区**




**注 1：** 内核程序 RAM 大小 = BMXDUDBA - BMXDKPBA。

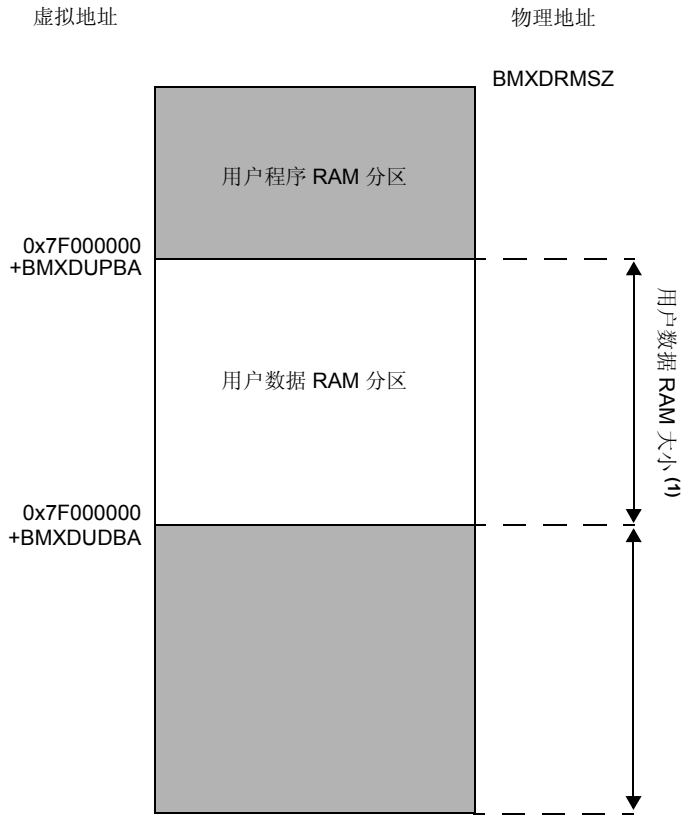
**2：** BMXDKPBA、 BMXDUDBA 或 BMXDUPBA 均不为 0。

#### 3.4.3.3 用户数据 RAM 分区

对于用户模式应用程序，需要处于 RAM 中的用户模式数据分区。该分区从地址 0x7F000000 + BMXDUDBA 处开始，其大小由 BMXDUPBA - BMXDUDBA 指定 （见图 3-8）。

用户数据 RAM 分区在复位时不存在，因为 BMXDUDBA 和 BMXDUPBA 寄存器在复位时默认设为 0。

**图 3-8： 用户数据 RAM 分区**




**注 1：** 用户数据 RAM 大小 = BMXDUPBA - BMXDUDBA。

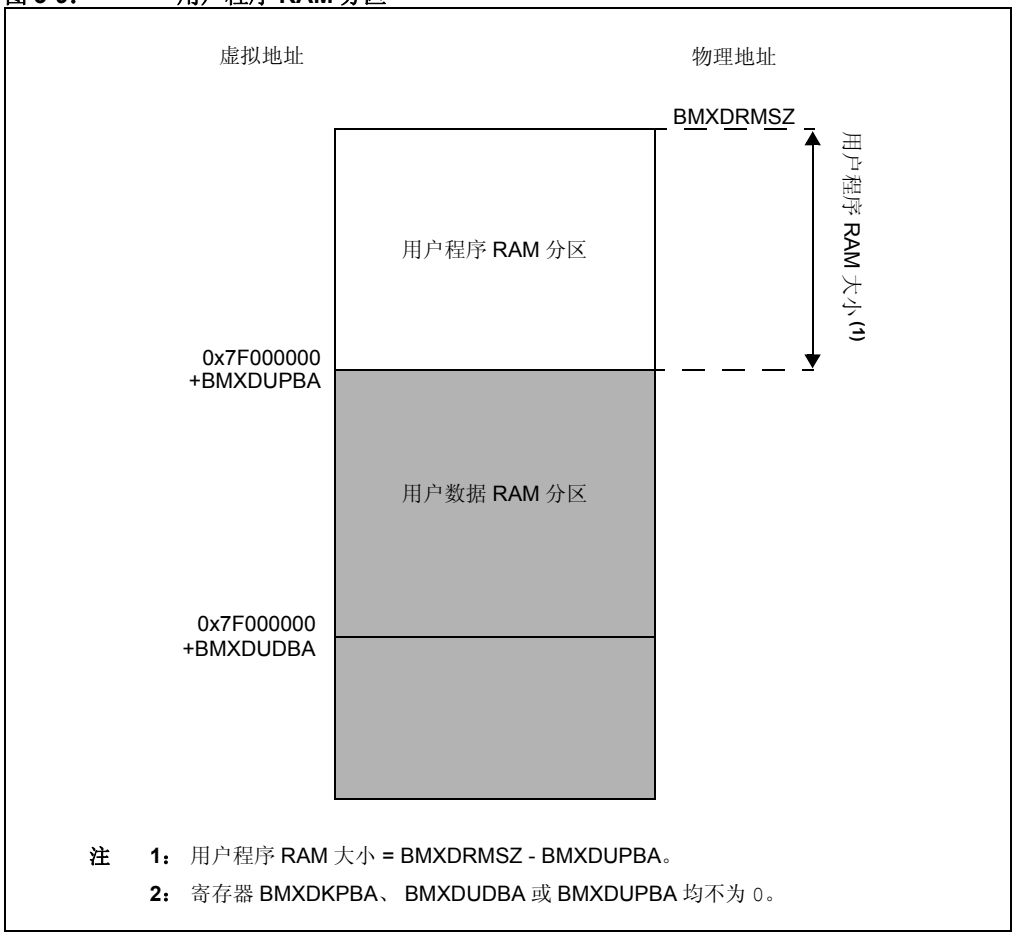
**2：** 寄存器 BMXDKPBA、 BMXDUDBA 或 BMXDUPBA 均不为 0。

#### 3.4.3.4 用户程序 RAM 分区

如果需要在用户模式下从数据 RAM 中执行代码，则需要处于数据 RAM 中的用户程序分区。该分区从地址 0x7F000000 + BMXDUPBA 处开始，其大小由 BMXDRMSZ - BMXDUPBA 指定。请参见图 3-9。

用户程序 RAM 分区在复位时不存在，因为 BMXDUPBA 寄存器在复位时默认设为 0。

**图 3-9： 用户程序 RAM 分区**




**注 1：** 用户程序 RAM 大小 = BMXDRMSZ - BMXDUPBA。

**2：** 寄存器 BMXDKPBA、 BMXDUDBA 或 BMXDUPBA 均不为 0。

#### 3.4.3.5 RAM 分区示例

本节提供了以下 RAM 分区实例：

- RAM 划分为内核数据分区
- RAM 划分为内核数据和内核程序分区
- RAM 划分为内核数据和用户数据分区
- RAM 划分为内核数据、内核程序和用户数据分区
- RAM 划分为内核数据、内核程序、用户数据和用户程序分区

**例 1. RAM 划分为内核数据分区**

整个 RAM 在复位后会被划分为内核数据 RAM 分区。无需任何其他编程操作。 BMXDKPBA、BMXDUDBA 或 BMXDUPBA 寄存器设置为 0 时，可以将整个 RAM 空间划分为内核数据分区（见图 3-5）。

**例 2. RAM 划分为内核数据和内核程序分区**

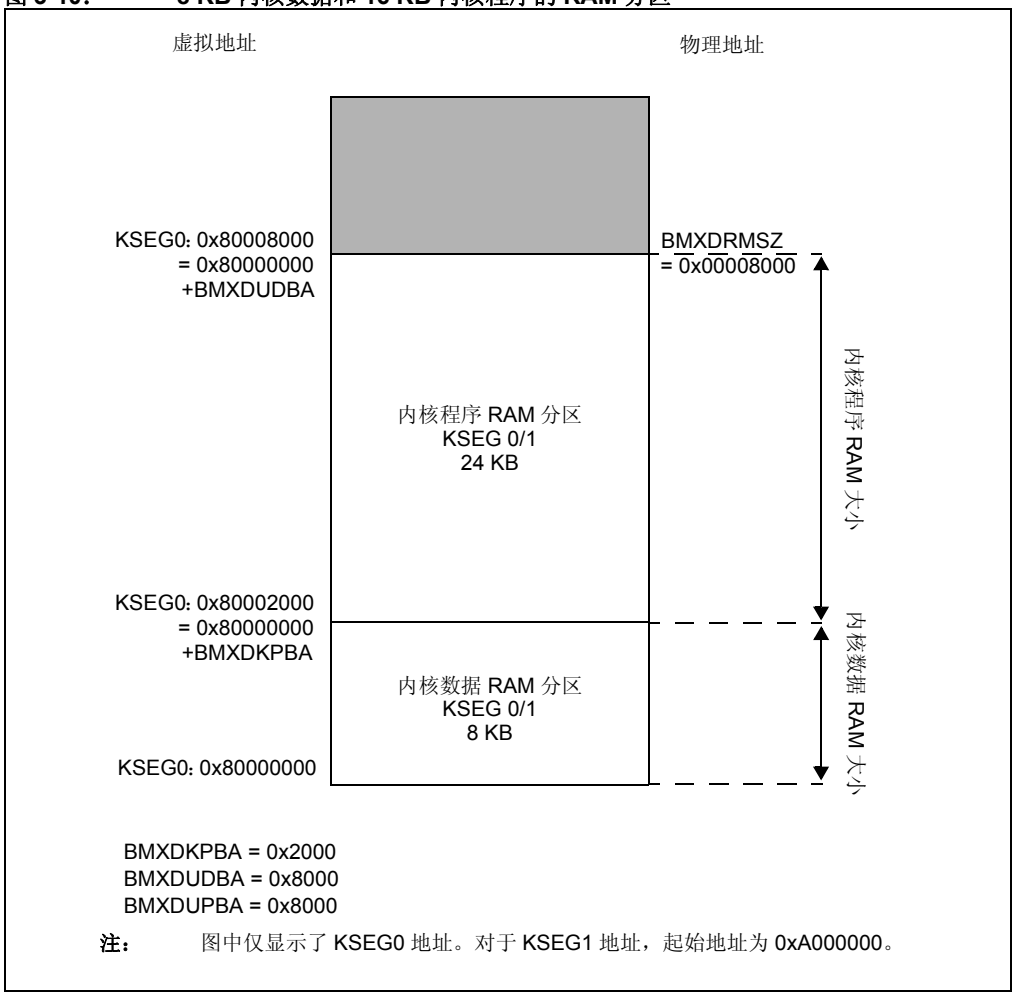
对于该示例，假设 PIC32MX 器件上的可用 RAM 为 32 KB ；其中，需要 8 KB 内核数据 RAM 和 24 KB 内核程序 RAM。在该示例中，用户数据 RAM 和用户程序 RAM 的大小设置为 0。

请注意，内核数据 RAM 分区总是必需的。详情请参见图 3-10。

寄存器的值如下：

- BMXDRMSZ = 0x00008000 （只读值）
- BMXDKPBA = 0x00002000 （即， 8 KB 内核数据）
- BMXDUDBA = 0x00008000 （即， 0x6000 内核程序）
- BMXDUPBA = 0x00008000 （即，用户数据大小 = 0，用户程序大小 = 0）

**图 3-10： 8 KB 内核数据和 16 KB 内核程序的 RAM 分区**




**注：** 图中仅显示了 KSEG0 地址。对于 KSEG1 地址，起始地址为 0xA000000。

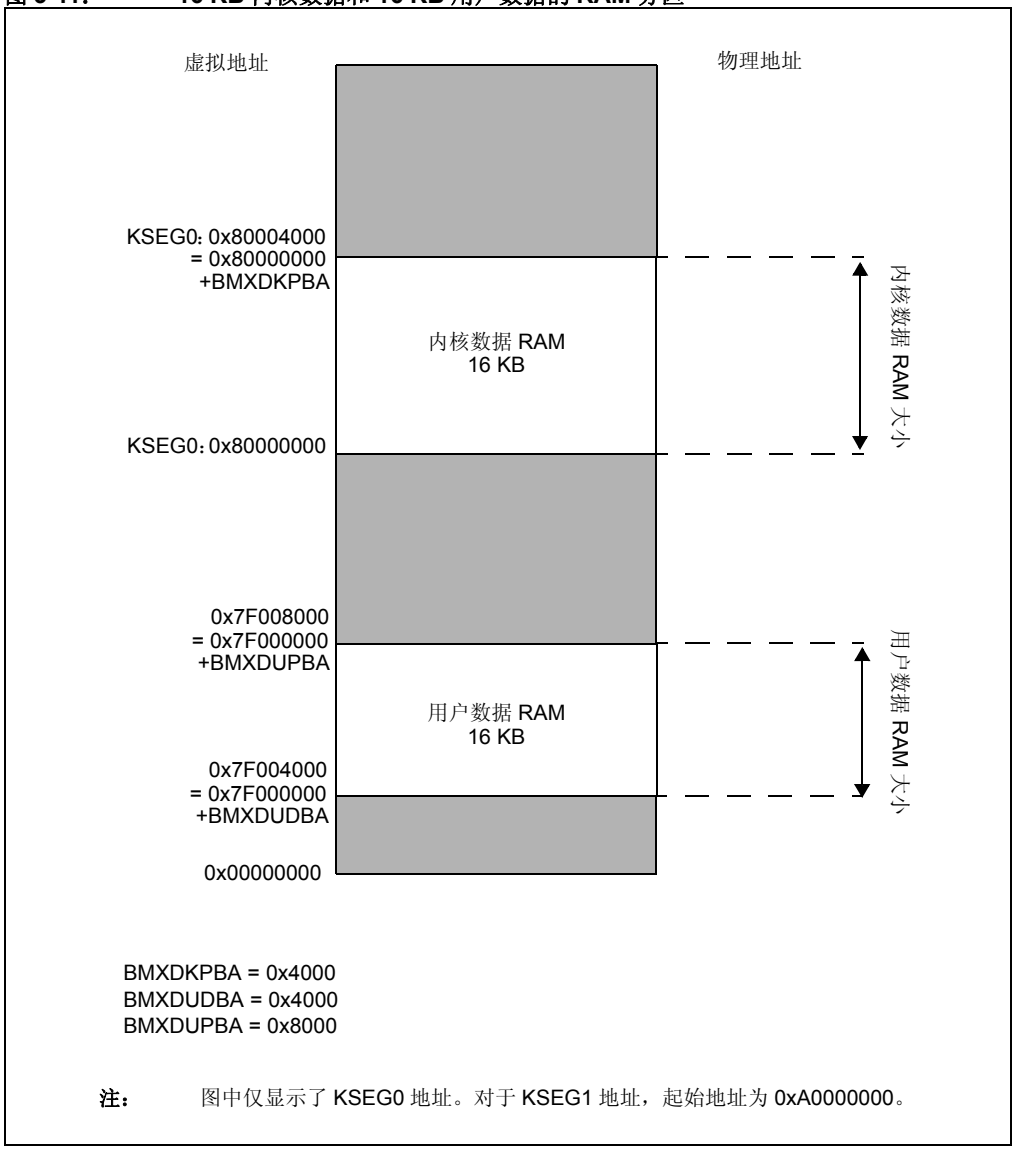
**例 3. RAM 划分为内核数据和用户数据分区**

对于该示例，假设 PIC32MX 器件上的可用 RAM 为 32 KB ；其中，需要 16 KB 内核数据 RAM 和 16 KB 用户数据 RAM。在该示例中，内核程序 RAM 和用户程序 RAM 的大小设置为 0。详情请参见图 3-11。

寄存器的值如下：

- BMXDRMSZ = 0x00008000 （只读值）
- BMXDKPBA = 0x00004000 （即， 16 KB 内核数据）
- BMXDUDBA = 0x00004000 （即， 0 内核程序）
- BMXDUPBA = 0x00008000 （即，用户数据大小 = 16 KB，用户程序大小 = 0）

**图 3-11： 16 KB 内核数据和 16 KB 用户数据的 RAM 分区**




**注：** 图中仅显示了 KSEG0 地址。对于 KSEG1 地址，起始地址为 0xA0000000。

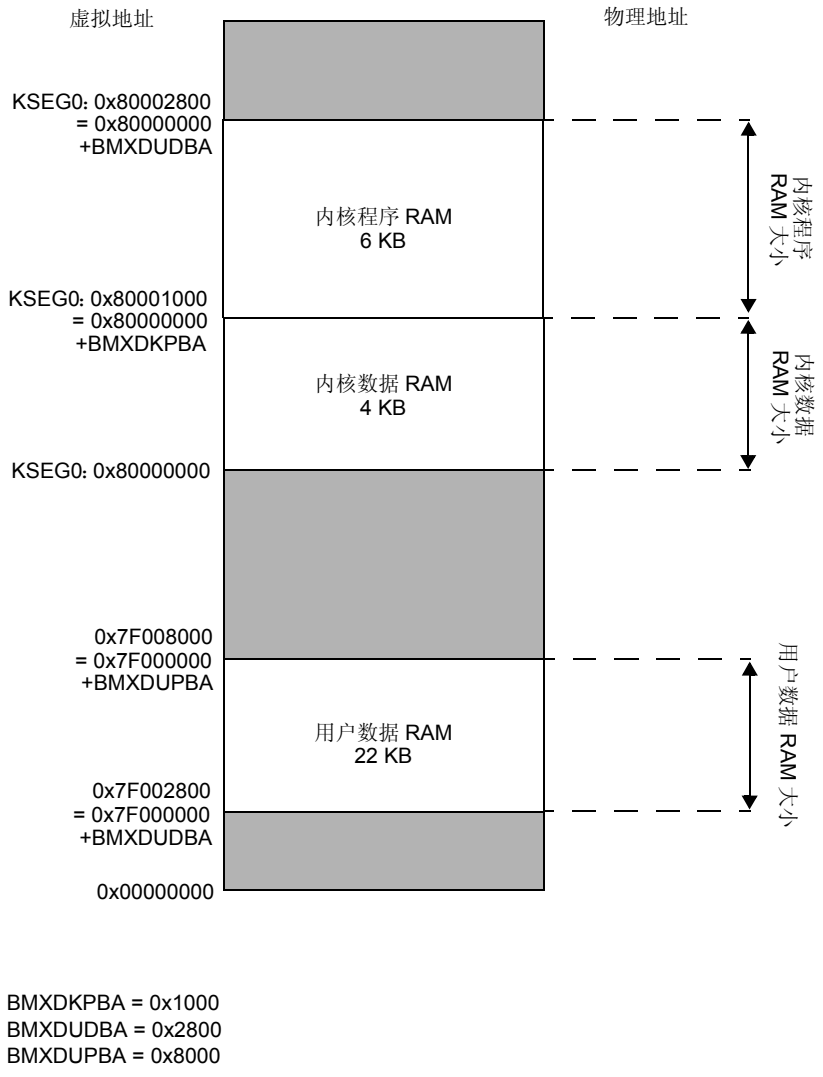
**例 4. RAM 划分为内核数据、内核程序和用户数据分区**

对于该示例，假设 PIC32MX 器件上的可用 RAM 为 32 KB；其中，需要 4 KB 内核数据 RAM、6 KB 内核程序 RAM 和 22 KB 用户数据 RAM。在该示例中，用户程序 RAM 的大小设置为 0。详情请参见图 3-12。

寄存器的值如下：

- BMXDRMSZ = 0x00008000（只读值）
- BMXDKPBA = 0x00001000（即， 4 KB 内核数据）
- BMXDUDBA = 0x00002800（即， 6 KB 内核程序）
- BMXDUPBA = 0x00008000（即，用户数据大小 = 22 KB，用户程序大小 = 0）

**图 3-12： 4 KB 内核数据、 6 KB 内核程序和 22 KB 用户数据的 RAM 分区**




**注：** 图中仅显示了 KSEG0 地址。对于 KSEG1 地址，起始地址为 0xA0000000。

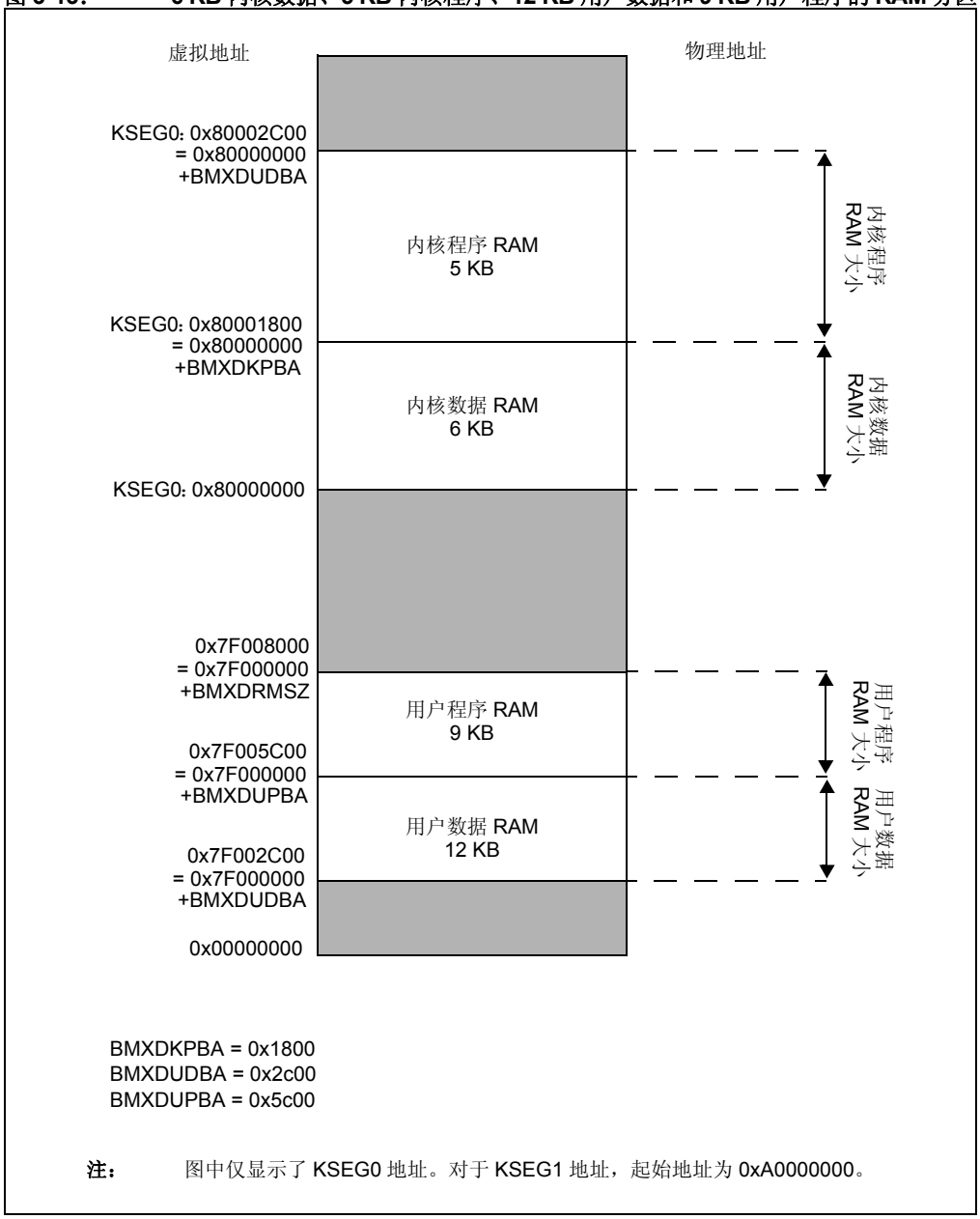
**例 5. RAM 划分为内核数据、内核程序、用户数据和用户程序分区**

对于该示例，假设 PIC32MX 器件上的可用 RAM 为 32 KB；其中，需要 6 KB 内核数据 RAM、5 KB 内核程序 RAM、 12 KB 用户数据 RAM 和 9 KB 用户程序 RAM。详情请参见图 3-13。

寄存器的值如下：

- BMXDRMSZ = 0x00008000 （只读值）
- BMXDKPBA = 0x00001800 （即， 6 KB 内核数据）
- BMXDUDBA = 0x00002C00 （即， 5 KB 内核程序）
- BMXDUPBA = 0x00005C00 （即，用户数据大小 = 12 KB，用户程序大小 = 9 KB）

**图 3-13： 6 KB 内核数据、5 KB 内核程序、12 KB 用户数据和 9 KB 用户程序的 RAM 分区**




**注：** 图中仅显示了 KSEG0 地址。对于 KSEG1 地址，起始地址为 0xA0000000。

## 3.5 总线矩阵

处理器支持两种工作模式，即内核模式和用户模式。总线矩阵控制每种模式的存储器分配。它还控制给定地址空间区域的访问类型，即程序或数据访问。

总线矩阵将主器件（统称为发起器）与从器件（统称为目标）相连接。在主总线结构上，PIC32MX 产品系列最多可以有 5 个发起器和 3 个目标 （例如，闪存和 RAM 等）。

在 5 个可能的发起器中， CPU 指令总线 （CPU IS）、 CPU 数据总线 （CPU DS）、在线调试（ICD）和 DMA 控制器（DMA）是默认的一组发起器，总是存在。PIC32MX 还包含了发起器扩展接口 （Initiator Expansion Interface， IXI），用以支持未来扩展的附加发起器。

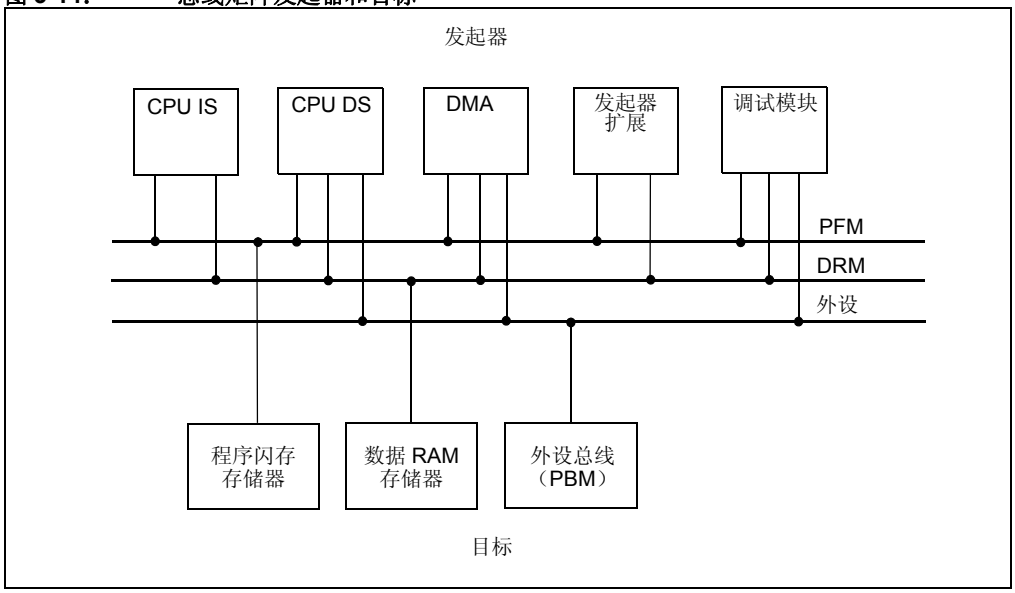
总线矩阵可以对映射到目标的通用地址范围进行解码。目标（存储器或外设）可能会提供附加的地址，具体取决于它的功能。

表 3-3 列出了发起器可以访问的目标。

**表 3-3： 发起器访问映射**

| 发起器 | 目标 | | |
|---|---|---|---|
| | 闪存 | RAM | 外设总线 |
| **CPU IS** | 有 | 有 | 无 |
| **CPU DS** | 有 | 有 | 有 |
| **DMA** | 有 | 有 | 有 |
| **IXI** | 有 | 有 | 无 |
| **ICD** | 有 | 有 | 有 |

**图 3-14： 总线矩阵发起器和目标**



### 3.5.1 发起器仲裁模式

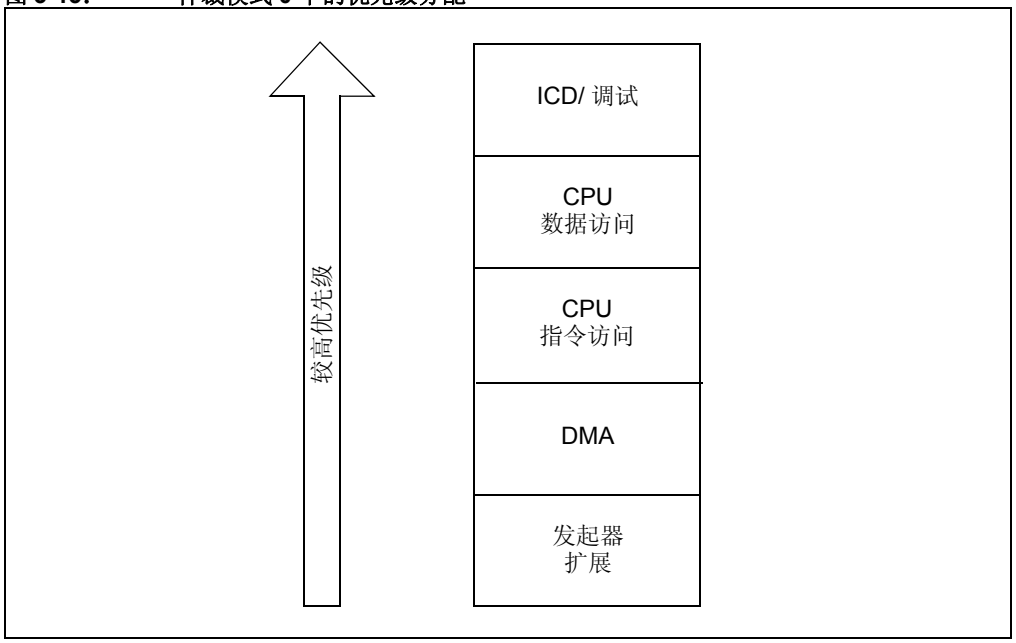
由于可能会有多个发起器尝试访问同一目标，所以必须使用仲裁方案来控制对于目标的访问。仲裁模式为所有发起器分配优先级。进行目标访问时，优先级较高的发起器总是优先于优先级较低的发起器。

#### 3.5.1.1 仲裁模式 0

图 3-15 显示了仲裁模式 0 中的固定优先级方案。CPU 数据和指令访问的优先级高于 DMA 访问。该模式会使 DMA “挨饿”，所以请在不使用 DMA 时选择该模式。

如图 3-15 所示，每个发起器都分配了固定优先级。寄存器字段 BMXARB （BMXCON<2:0>）编程为 0 时，将选择模式 0 工作。

**图 3-15： 仲裁模式 0 中的优先级分配**

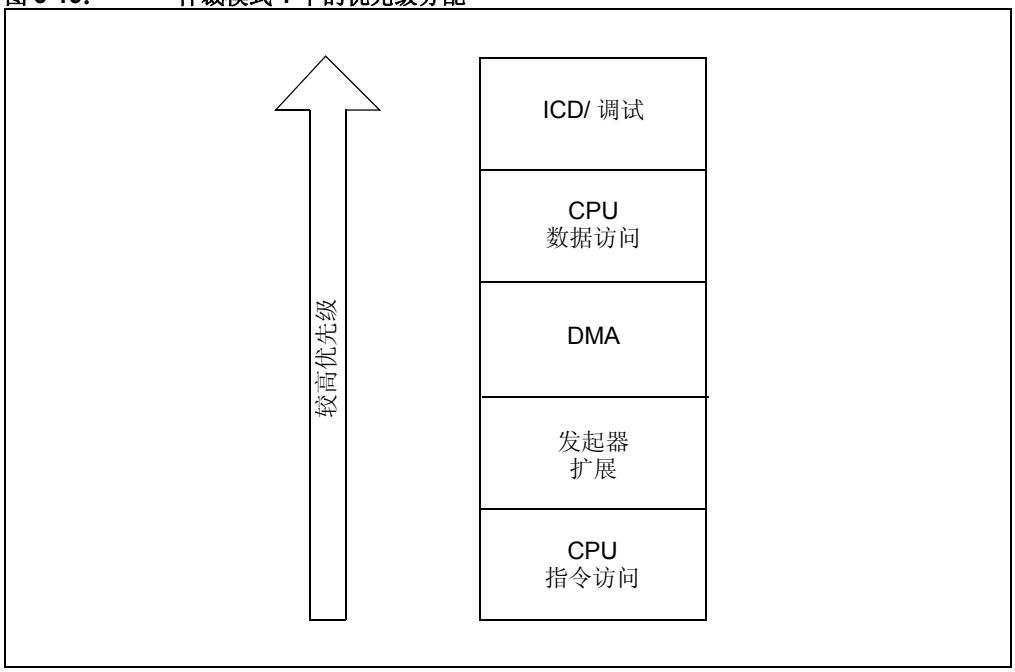


#### 3.5.1.2 仲裁模式 1

仲裁模式 1 是类似于模式 0 的固定优先级方案；但是，CPU IS 的优先级总是最低。图 3-16 显示了模式 1 中的优先级方案。模式 1 仲裁是默认模式。

寄存器字段 BMXARB （BMXCON<2:0>）编程为 1 时，将选择模式 1 工作。

**图 3-16： 仲裁模式 1 中的优先级分配**

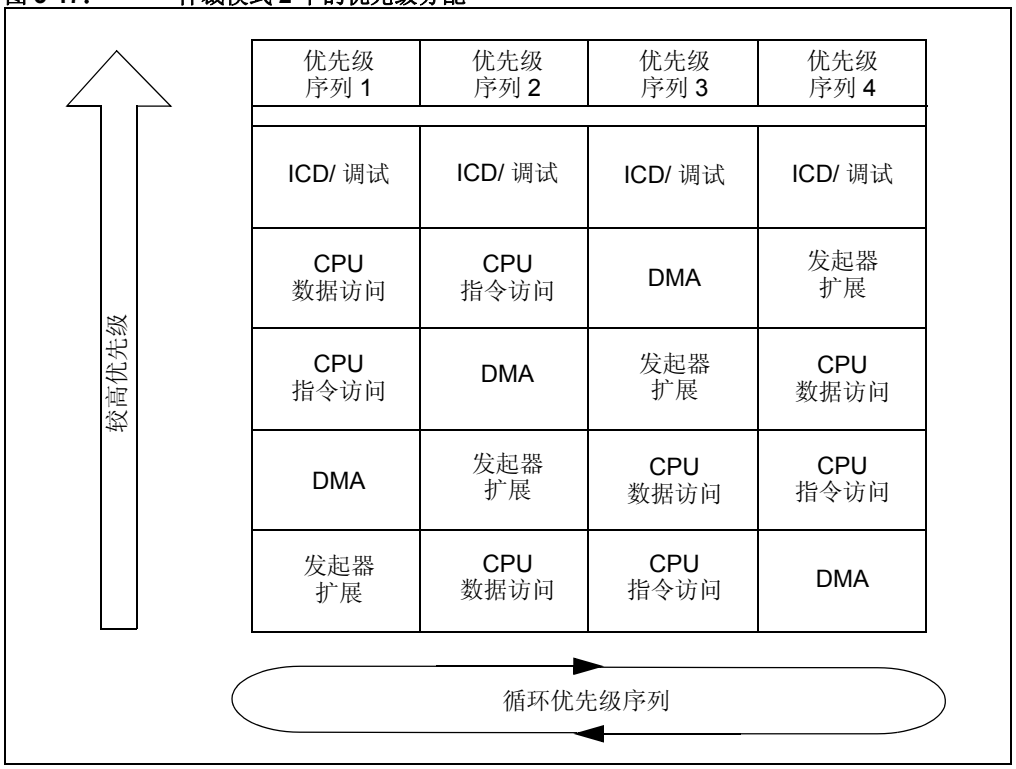


#### 3.5.1.3 仲裁模式 2

模式 2 仲裁支持为所有发起器分配循环优先级。每个发起器不分配固定优先级，而是以循环方式分配最高优先级。在该模式下，除以下例外情况，将应用循环优先级：

1. CPU 数据总是优先于 CPU 指令。
2. ICD 优先级总是最高。
3. 当 CPU 处理异常 （EXL = 1）或错误 （ERL = 1）时，仲裁器会临时恢复为模式 0。

**图 3-17： 仲裁模式 2 中的优先级分配**




请注意，如果存在待处理的 CPU 数据访问，则在循环优先级方案中不会选择优先级序列 2。在这种情况下，当数据访问完成时，会立即选择序列 2。

寄存器字段 BMXARB （BMXCON<2:0>）编程为 2 时，将选择模式 2 工作。

### 3.5.2 总线错误异常

发生以下情况时，总线矩阵会产生总线错误异常：

- 尝试访问未实现的存储器
- 尝试访问非法目标
- 尝试写入程序闪存存储器

总线错误异常可以通过清零BMXCON寄存器中的BMXERRxxx位而临时禁止，但不建议这么做。

处于 DEBUG （调试）模式时，对于来自 CPU IS 和 CPU DS 的访问，总线矩阵会禁止总线错误异常。

### 3.5.3 精确中断断点支持

PIC32MX 通过在数据 RAM 访问中插入 1 个等待状态来支持精确中断断点。利用该方法， CPU 可以恰好在断点地址指令之前停止执行。这对于被中断的存储指令非常有用。不使用等待状态时，仍然会在存储指令处发生中断，但是，DRM 单元会被更新为存储值。如果使能了等待状态，则 DRM 不会被更新为存储值。

## 3.6 I/O 引脚控制

没有任何引脚与该模块关联。

## 3.7 节能和调试模式下的操作

### 3.7.1 上电或欠压复位 （Brown-out Reset， BOR）时的存储器操作：

- 数据 RAM 的内容是未定义的
- BMXxxxBA 寄存器复位为 0
- CPU 切换为内核模式

### 3.7.2 复位时的存储器操作：

- 数据 RAM 内容保持不变。如果器件受代码保护，则 RAM 内容会被清零
- BMX 基址寄存器 （BMXxxxBA）设置为 0
- CPU 切换为内核模式

### 3.7.3 从器件休眠或空闲模式唤醒时的存储器操作：

- RAM 内容保持不变
- BMX 基址寄存器 （BMXxxxBA）内容不变
- CPU 模式不变

## 3.8 代码示例

**例 3-1： 在程序闪存中创建 12K 的用户模式分区**

```
BMXPUPBA = BMXPFMSZ - (12*1024);  // User Mode Flash 12K,
                                  // Kernel Mode Flash 500K (512K-12K)
```

**例 3-2： 创建 16K 的内核模式数据 RAM 分区；剩余的 RAM 用作内核程序分区**

```
BMXDKPBA = 16*1024;
BMXDUDBA = BMXDRMSZ;
BMXDUPBA = BMXDRMSZ;
```

例 3-3 可用于在 RAM 中创建以下分区：

- 内核模式数据 = 12K
- 内核模式程序 = 6K
- 用户模式数据 = 8K
- 用户模式程序 = 6K

**例 3-3： 创建 RAM 分区**

```
BMXDKPBA = 12*1024;               // Kernel Data Partition of 12K.
                                  // Start offset of Kernel Program Partition
BMXDUDBA = BMXDKPBA + (6*1024);   // Kernel Program Partition of 6K
                                  // Start offset of User Data Partition
BMXDUPBA = BMXDUDBA + (8*1024);   // User Data Partition of 8K
                                  // Start offset of User Program Partition.
                                  // This partition will go up to the size of
                                  // RAM (32K).So the partition size will be
                                  // 6K (32K - 8K - 6K - 12K)
```

## 3.9 设计技巧

**问 1：** ***在复位时， CPU 运行于哪种模式？***

**答 1：** CPU 以内核模式启动。整个 RAM 映射到 KSEG0 和 KSEG1 中的内核数据段。闪存映射到 KSEG0 和 KSEG1 中的内核程序段。此外， ERL = 1，应将它复位为 0（通常是在 C 启动代码中）。

**问 2：** ***我是否需要初始化 BMX 寄存器？***

**答 2：** 通常不需要。可以将 BMX 寄存器保留为默认值，这样内核模式应用程序可以使用最大的 RAM 和闪存。如果要从 RAM 中运行代码或设置用户模式分区，则需要配置 BMX 寄存器。

**问 3：** ***CPU 复位向量地址是什么？***

**答 3：** CPU 复位地址为 0xBFC00000。

**问 4：** ***什么是总线错误异常？***

**答 4：** 当 CPU 尝试访问未实现地址时，会发生总线错误异常。此外，当 CPU 尝试从 RAM 中执行程序，但未定义 RAM 程序分区时，也会产生总线错误异常。

## 3.10 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 PIC32MX 器件系列而编写的，但其概念是相近的，通过适当修改并受到一定限制即可使用。当前与 PIC32MX 系列的存储器构成相关的应用笔记包括：

| 标题 | 应用笔记编号 |
|---|---|
| 目前没有相关的应用笔记。 | N/A |

**注：** 如需获取更多 PIC32MX 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

## 3.11 版本历史

### 版本 A （2007 年 8 月）

这是本文档的初始版本。

### 版本 B （2007 年 10 月）

更新了文档 （删除了 “机密” 状态）。

### 版本 C （2008 年 4 月）

将状态修改为 “初稿”； 将 U-0 修改为 r-x。

### 版本 D （2008 年 6 月）

将保留位从 “保持为” 更改为 “写入”。

### 版本 E （2009 年 7 月）

该版本包括以下更新：

- 对整篇文档的文字和格式进行了少量更新
- 在以下内容中增加了 “注 1”、“注 2” 和 “注 3”（它们分别介绍清零、置 1 和取反寄存器）：
  - 表 3-1：存储器构成 SFR 汇总
  - 寄存器 3-1：BMXCON：总线矩阵配置寄存器
  - 寄存器 3-2：BMXDKPBA：数据 RAM 内核程序基址寄存器
  - 寄存器 3-3：BMXDUDBA：数据 RAM 用户数据基址寄存器
  - 寄存器 3-4：BMXDUPBA：数据 RAM 用户程序基址寄存器
  - 寄存器 3-6：BMXPUPBA：程序闪存 （PFM）用户程序基址寄存器
- 删除了所有清零、置 1 和取反寄存器的说明
- 在 BMXDRMSZ：数据 RAM 大小寄存器中增加了附加的位值定义 （0x0001000）（见寄存器 3-5）

### 版本 F （2010 年 7 月）

该版本包括以下更新：

- 对整篇文档的文字和格式进行了少量更新
- 对下列寄存器添加注 4 和 5：
  - BMXDKPBA：数据 RAM 内核程序基址寄存器 （见寄存器 3-2）
  - BMXDUDBA：数据 RAM 用户数据基址寄存器 （见寄存器 3-3）
  - BMXDUPBA：数据 RAM 用户程序基址寄存器 （见寄存器 3-4）
  - BMXPUPBA：程序闪存 （PFM）用户程序基址寄存器 （见寄存器 3-6）
- 更新 PIC32MX 地址映射 （见表 3-2）

注：

**请注意以下有关 Microchip 器件代码保护功能的要点：**

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。



ISBN: 978-1-60932-658-6

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

*Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2002 认证。公司在 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。*

# 全球销售及服务网点

## 美洲

**公司总部 Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

**亚特兰大 Atlanta**
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

**波士顿 Boston**
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

**芝加哥 Chicago**
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

**克里夫兰 Cleveland**
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

**达拉斯 Dallas**
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

**底特律 Detroit**
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

**科科莫 Kokomo**
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

**洛杉矶 Los Angeles**
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

**圣克拉拉 Santa Clara**
Santa Clara, CA
Tel: 1-408-961-6444
Fax: 1-408-961-6445

**加拿大多伦多 Toronto**
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

## 亚太地区

**亚太总部 Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

**中国 – 北京**
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

**中国 – 成都**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**中国 – 重庆**
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

**中国 – 香港特别行政区**
Tel: 852-2401-1200
Fax: 852-2401-3431

**中国 – 南京**
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

**中国 – 青岛**
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

**中国 – 上海**
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

**中国 – 沈阳**
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

**中国 – 深圳**
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

**中国 – 武汉**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**中国 – 西安**
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

**中国 – 厦门**
Tel: 86-592-238-8138
Fax: 86-592-238-8130

**中国 – 珠海**
Tel: 86-756-321-0040
Fax: 86-756-321-0049

**台湾地区 – 高雄**
Tel: 886-7-213-7830
Fax: 886-7-330-9305

**台湾地区 – 台北**
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

## 亚太地区

**台湾地区 – 新竹**
Tel: 886-3-6578-300
Fax: 886-3-6578-370

**澳大利亚 Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**印度 India - Bangalore**
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

**印度 India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**印度 India - Pune**
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

**日本 Japan - Yokohama**
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

**韩国 Korea - Daegu**
Tel: 82-53-744-4301
Fax: 82-53-744-4302

**韩国 Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

**马来西亚 Malaysia - Kuala Lumpur**
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

**马来西亚 Malaysia - Penang**
Tel: 60-4-227-8870
Fax: 60-4-227-4068

**菲律宾 Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**新加坡 Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**泰国 Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350

## 欧洲

**奥地利 Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**丹麦 Denmark-Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**法国 France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**德国 Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**意大利 Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**荷兰 Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**西班牙 Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**英国 UK - Wokingham**
Tel: 44-118-921-5869
Fax: 44-118-921-5820

07/15/10