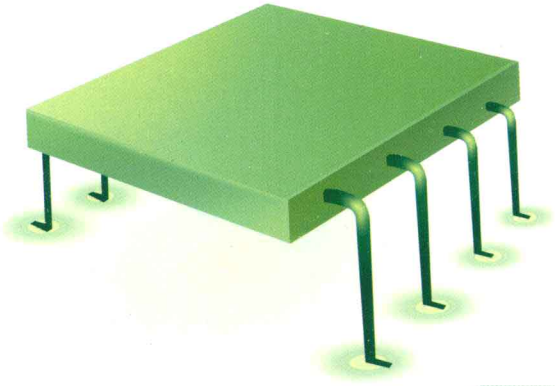
# ARM 嵌入式应用技术

## ——基于Proteus虚拟仿真

徐爱钧　徐　阳　编著



北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

CD-ROM INCLUDED

# ARM 嵌入式应用技术
# ——基于 Proteus 虚拟仿真

徐爱钧　徐　阳　编著



北京航空航天大学出版社

## 内 容 简 介

本书以 NXP 公司的 LPC2100 系列 ARM 处理器为原型，以 Keil MDK for ARM 为软件平台，阐述了基于 Proteus 虚拟仿真的 ARM 嵌入式应用技术，分析了与 ARM 处理器架构相关的技术要点，详细介绍 LPC2138 ARM 处理器片内功能资源和外部扩展应用技术，并给出了大量 Proteus 虚拟仿真实例。

本书配光盘一张，其中包含 Proteus DEMO 版软件包和书中全部实例。

本书适合于从事 ARM 嵌入式系统开发设计的工程技术人员阅读，也可作为大专院校相关专业嵌入式系统课程的教学用书。

**图书在版编目(CIP)数据**

ARM 嵌入式应用技术 ：基于 Proteus 虚拟仿真 / 徐爱钧，徐阳编著. -- 北京 ：北京航空航天大学出版社，2012.7

ISBN 978-7-5124-0818-0

Ⅰ. ①A… Ⅱ. ①徐… ②徐… Ⅲ. ①微处理器—系统设计 Ⅳ. ①TP332

中国版本图书馆 CIP 数据核字(2012)第 099066 号



**ARM 嵌入式应用技术**

**——基于 Proteus 虚拟仿真**

徐爱钧 徐 阳 编著

责任编辑 张冀青

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) http://www.buaapress.com.cn

发行部电话：(010)82317024 传真：(010)82328026

读者信箱：emsbook@gmail.com 邮购电话：(010)82316936

涿州市新华印刷有限公司印装 各地书店经销

*

开本：710×1000 1/16 印张：21.75 字数：476 千字

2012 年 7 月第 1 版 2012 年 7 月第 1 次印刷 印数：4 000 册

ISBN 978-7-5124-0818-0 定价：45.00 元(含光盘 1 张)

# 前　言

随着 ARM 处理器在嵌入式系统中的应用日益广泛，各种 ARM 开发技术层出不穷。英国 Labcenter 公司推出的 Proteus 软件是一款极好的 ARM 处理器开发平台，它以其特有的虚拟仿真技术很好地解决了 ARM 处理器与其外围电路的设计和协同仿真问题，可以在没有 ARM 处理器实际硬件的条件下，利用 PC 进行虚拟仿真，实现 ARM 应用系统的软、硬件协同设计。采用 Proteus 虚拟仿真技术，可以在原理图设计阶段对系统性能进行评估，验证所设计的电路是否达到技术指标要求，使设计过程变得简单容易。

Proteus 软件已有 20 多年的历史，涵盖了 PIC、AVR、MCS8051、68HC11、ARM 等微处理器模型，以及多种常用电子元器件，包括 74 系列、CMOS4000 系列集成电路、A/D 和 D/A 转换器、键盘、LCD 显示器及 LED 显示器，还提供示波器、逻辑分析仪、通信终端、电压/电流表、$I^2C$/SPI 终端等各种虚拟仪表，这些都可以直接用于虚拟仿真，极大地提高了应用系统设计效率。

利用 Proteus 软件平台来学习 ARM 嵌入式系统应用开发技术，可以使学习过程变得直观形象。基于原理图的虚拟模型仿真，可实现源代码的程序调试，还能看到程序运行后的输入、输出效果。在 PC 上修改原理电路图要比修改实际硬件电路容易得多，成功进行虚拟仿真并获得期望结果之后，再制作硬件电路板进行在线调试，可以获得事半功倍的效果。

本书在构思及选材上，以易学易用为原则，详细介绍了 NXP 公司的 LPC2138 ARM 处理器片内功能资源和外部扩展应用技术，并给出了大量 Proteus 虚拟仿真实例。全书共分 6 章，各章主要内容如下：

第 1 章 ARM 体系结构基础。介绍 ARM 支持的数据类型和存储器结构、ARM 处理器的工作状态和运行模式、寻址方式、指令集及 ARM 汇编语言规范，给出了用汇编语言编写的启动程序和其他范例。

第 2 章 Proteus for ARM7 虚拟仿真。介绍 Proteus 支持的 ARM 模型、ELF/SWARF 装载器，以及在 ISIS 集成环境中绘制原理电路图，进行源代码仿真调试，还有与 Keil 环境联机仿真的方法。

第 3 章 LPC213x ARM 处理器。介绍 ARM 处理器特性与存储器结构、系统控制模块、向量中断控制器 VIC、引脚功能配置等。

第4章 LPC2138片内集成功能应用技术。介绍通用输入/输出端口GPIO、通用异步接收发送器UART、$I^2C$接口、SPI接口、SSP接口、定时器/计数器、脉宽调制器PWM、A/D和D/A转换器、实时时钟和看门狗定时器，并给出了具体应用实例。

第5章 LPC2138片外扩展功能应用技术。介绍点阵字符型和点阵图形液晶显示模块接口技术、用DS18B20和LPC2138实现的数字温度计、用DS1302和LPC2138实现的万年历、SD卡与LPC2138的接口及应用、网络芯片ENC28J60与LPC2138的接口及应用。

第6章 μC/OS-II在LPC2138上的移植与应用。介绍μC/OS-II在LPC2138上的移植、μC/OS-II的任务管理、任务的同步与通信，以及编写μC/OS-II应用程序的方法。

每章都给出了大量应用实例，所有实例均在Proteus平台上仿真通过，并随本书配套光盘提供给读者，对加深理解LPC2138 ARM处理器基本原理和提高应用设计能力具有极大帮助。本书在编写过程中得到了广州风标电子技术有限公司(http://www.windway.cn)匡载华总经理的大力支持和热情帮助，还得到彭秀华、吴子平、杨振、范林、肖恩凯、杨佳恒、郑传波、郑勇、李致平、罗杰、杨新浩、牛宣云等的协助，在此一并表示感谢。

由于作者水平有限，书中难免会有错误和不妥之处，恳请广大读者批评指正。有兴趣的朋友可发送邮件到ajxu@tom.com和ajxu41@sohu.com，直接与作者联系；也可发送邮件到bhcbslx@sina.com，与本书策划编辑进行交流。

Proteus的DEMO软件可到官方网站http://www.labcenter.co.uk下载，或者与国内代理商广州风标电子技术有限公司联系购买正版软件。

徐爱钧　徐　阳

2012年2月于长江大学

# 目 录

**第 1 章 ARM 体系结构基础** …… 1

1.1 ARM 支持的数据类型和存储器结构 …… 1
1.2 ARM 处理器的工作状态和运行模式 …… 3
1.3 寄存器组织 …… 4
1.3.1 寄存器分类 …… 4
1.3.2 通用寄存器 …… 6
1.3.3 程序状态寄存器 …… 7
1.4 异 常 …… 9
1.4.1 ARM 体系结构所支持的异常类型 …… 9
1.4.2 各类异常的具体描述 …… 10
1.4.3 对异常的响应和返回 …… 12
1.5 ARM 指令集 …… 13
1.5.1 ARM 指令的功能与格式 …… 13
1.5.2 指令的条件码 …… 14
1.6 ARM 指令的寻址方式 …… 15
1.6.1 寄存器寻址 …… 15
1.6.2 立即寻址 …… 15
1.6.3 寄存器移位寻址 …… 16
1.6.4 寄存器间接寻址 …… 16
1.6.5 基址寻址 …… 17
1.6.6 相对寻址 …… 17
1.6.7 多寄存器寻址 …… 17
1.6.8 堆栈寻址 …… 18
1.6.9 块复制寻址 …… 18
1.6.10 ARM 伪指令 …… 19
1.7 Thumb 指令集 …… 20
1.8 ARM 汇编语言编程 …… 21
1.8.1 ARM 汇编语言规范 …… 21

1.8.2　汇编伪指令 …… 21
1.8.3　程序设计举例 …… 22
1.8.4　汇编语言与 C/C++混合编程 …… 24
1.9　启动代码 …… 24
1.9.1　Startup.s 文件 …… 25
1.9.2　IRQ.s 文件 …… 28
1.9.3　Target.c 文件 …… 29
1.9.4　Target.h 文件 …… 30
1.9.5　config.h 文件 …… 31
1.9.6　分散加载文件 …… 31

**第 2 章　Proteus for ARM7 虚拟仿真** …… 34

2.1　Proteus for ARM 简介 …… 34
2.1.1　Proteus 支持的 ARM 模型 …… 34
2.1.2　Proteus ELF/DWARF 装载器 …… 36
2.1.3　Proteus LPC2000 的调试窗口 …… 37
2.2　集成环境 ISIS …… 41
2.3　原理图绘制与源代码仿真调试 …… 43
2.4　原理图与 Keil 环境联机仿真调试 …… 48

**第 3 章　LPC213x ARM 处理器** …… 55

3.1　LPC213x 处理器特性与存储器结构 …… 55
3.1.1　主要特性 …… 55
3.1.2　存储器结构 …… 56
3.1.3　存储器重映射和 Boot Block …… 58
3.1.4　存储器加速模块 …… 61
3.2　系统控制模块 …… 63
3.2.1　时钟频率控制 …… 63
3.2.2　VPB 分频器 …… 67
3.2.3　功率控制 …… 68
3.2.4　复　位 …… 70
3.2.5　唤醒定时器 …… 71
3.3　外部中断输入 …… 72
3.4　向量中断控制器 VIC …… 75
3.4.1　VIC 寄存器 …… 76
3.4.2　中断源 …… 79
3.4.3　VIC 使用注意事项 …… 81
3.5　引脚功能配置 …… 82

3.5.1　引脚选择寄存器 …… 82
3.5.2　引脚配置示例 …… 84
3.6　系统控制应用举例 …… 84
3.6.1　存储器映射 …… 84
3.6.2　锁相环 …… 88
3.6.3　存储器加速模块 …… 92
3.7　VIC 中断应用举例 …… 93
3.7.1　外部中断 …… 94
3.7.2　向量中断 …… 96
3.7.3　嵌套中断 …… 97
3.7.4　快速中断 …… 99
3.7.5　软件中断 …… 100

**第 4 章　LPC2138 片内集成功能应用技术** …… 103

4.1　通用输入/输出端口 GPIO …… 103
4.1.1　主要特性 …… 103
4.1.2　寄存器描述 …… 103
4.1.3　应用举例 …… 105
4.2　通用异步接收发送器 UART …… 110
4.2.1　主要特性 …… 110
4.2.2　寄存器描述 …… 111
4.2.3　应用举例 …… 119
4.3　$I^2C$ 接口 …… 125
4.3.1　主要特性 …… 125
4.3.2　操作模式 …… 126
4.3.3　寄存器描述 …… 129
4.3.4　应用举例 …… 132
4.4　SPI 接口 …… 136
4.4.1　主要特性 …… 136
4.4.2　SPI 数据传输 …… 136
4.4.3　寄存器描述 …… 138
4.4.4　应用举例 …… 140
4.5　SSP 接口 …… 145
4.5.1　主要特性 …… 145
4.5.2　寄存器描述 …… 145
4.5.3　应用举例 …… 148
4.6　定时器/计数器 …… 151
4.6.1　主要特性 …… 151

4.6.2　寄存器描述 …… 152
4.6.3　应用举例 …… 157
4.7　脉宽调制器 PWM …… 165
4.7.1　主要特性 …… 165
4.7.2　寄存器描述 …… 167
4.7.3　应用举例 …… 173
4.8　A/D 转换器 …… 182
4.8.1　主要特性 …… 182
4.8.2　寄存器描述 …… 182
4.8.3　应用举例 …… 186
4.9　D/A 转换器 …… 192
4.9.1　主要特性 …… 192
4.9.2　寄存器描述 …… 192
4.9.3　应用举例 …… 193
4.10　实时时钟 …… 198
4.10.1　主要特性 …… 198
4.10.2　寄存器描述 …… 198
4.10.3　应用举例 …… 203
4.11　看门狗定时器 …… 208
4.11.1　主要特性 …… 208
4.11.2　寄存器描述 …… 209
4.11.3　应用举例 …… 211

**第 5 章　LPC2138 片外扩展功能应用技术** …… 216

5.1　液晶显示器 LCD 接口技术 …… 216
5.1.1　点阵字符型液晶显示模块接口技术 …… 216
5.1.2　12864 点阵图形液晶显示模块接口技术 …… 226
5.1.3　T6963C 点阵图形液晶显示模块接口技术 …… 235
5.2　用 DS18B20 和 LPC2138 实现的数字温度计 …… 244
5.3　用 DS1302 和 LPC2138 实现的万年历 …… 252
5.4　SD 卡与 LPC2138 的接口及应用 …… 265
5.4.1　SD 卡简介 …… 265
5.4.2　FAT16 文件系统 …… 267
5.4.3　SD 卡接口应用举例 …… 270
5.5　网络芯片 ENC28J60 与 LPC2138 的接口及应用 …… 284
5.5.1　ENC28J60 的功能模块与引脚分布 …… 284
5.5.2　ENC28J60 主要特性 …… 286
5.5.3　ENC28J60 应用举例 …… 288

**第 6 章　μC/OS－II 在 LPC2138 上的移植与应用** …… 305

6.1　μC/OS－II 简介 …… 305
6.2　μC/OS－II 在 LPC2138 上的移植 …… 306
6.2.1　编写 OS_CPU.H 文件 …… 308
6.2.2　编写 OS_CPU_C.C 文件 …… 309
6.2.3　编写 OS_CPU_A.S 文件 …… 314
6.2.4　关于中断 …… 317
6.2.5　挂接 SWI 软件中断 …… 318
6.2.6　中断服务程序 …… 319
6.3　编写 μC/OS－II 应用程序 …… 319
6.3.1　任务管理 …… 319
6.3.2　任务管理应用编程举例 …… 322
6.4　任务的同步与通信 …… 326
6.4.1　信号量 …… 326
6.4.2　信号量应用编程举例 …… 328
6.4.3　消息邮箱 …… 331
6.4.4　消息邮箱应用编程举例 …… 334

**参考文献** …… 338

# 第1章

# ARM体系结构基础

ARM是Advanced RISC Machines的缩写，ARM公司专注于设计，本身不生产芯片，靠转让设计许可由合作伙伴公司来生产基于ARM核的处理器芯片。目前，ARM公司在世界范围内的合作伙伴已超过100个，生产的ARM核芯片各具特色，在移动通信、手持计算、多媒体应用以及工业测量控制等领域得到广泛应用。ARM的32位RISC体系结构目前被公认为业界领先，所有ARM核处理器共享这一体系结构，从而确保应用人员在软件开发上可以获得最大的回报。ARM体系结构采用指令流水线，分别为取指、译码和执行；而ARM9体系结构则采用5级指令流水线，分别为取指、译码、执行、缓存和写回等，利用流水线重叠技术使系统性能大为提高。

## 1.1 ARM支持的数据类型和存储器结构

ARM体系结构支持三种数据类型：字(Word)，长度为32位；半字(Halfword)，长度为16位；字节(Byte)，长度为8位。

在ARM存储器组织中，半字必须与2字节边界对齐，字必须与4字节边界对齐，即半字必须开始于偶数地址，字必须开始于4的倍数的字节地址，如图1.1所示。

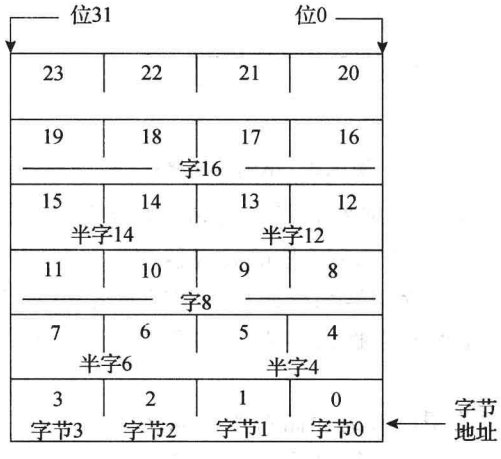



图1.1 ARM数据类型存储图

ARM 处理器采用冯·诺依曼(Von Neumann)形式的存储器结构，使用 $2^{32}$ 个字节长度为 8 位(即 1 字节)的单一、线性地址空间。从 0 字节开始到第 3 字节存放第 1 个字数据，从第 4 字节到第 7 字节存放第 2 个字数据，依次类推。指令和数据共用一条 32 位数据总线，只有加载、保存和交换指令可以访问存储器中的数据。作为 32 位的微处理器，ARM 体系结构所支持的最大地址空间为 4 GB($2^{32}$ 字节)。

ARM7 处理器采用 3 级指令流水线，在执行当前指令的同时对下一条指令进行译码，并对再下一条指令进行指令预取。如果预取操作溢出了地址空间顶端，则不会产生执行动作并导致不可预知的结果。

ARM 体系结构可以用两种方法存储字数据，分别称为大端格式(Big Endian)和小端格式(Little Endian)。在大端格式中，字数据的高字节存储在低地址中，而字数据的低字节存放在高地址中，如图 1.2 所示。

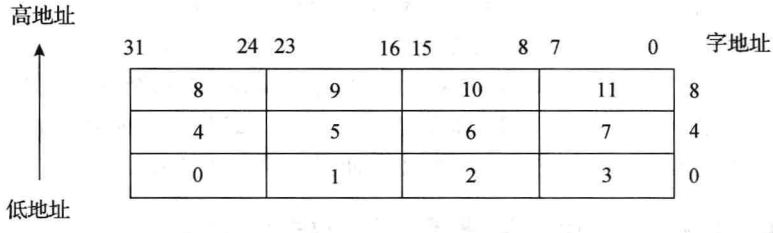



**图 1.2　大端存储格式**

与大端存储格式相反，在小端存储格式中，低地址中存放的是字数据的低字节，高地址中存放的是字数据的高字节，如图 1.3 所示。

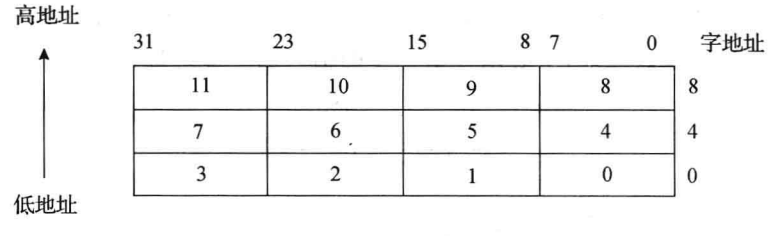



**图 1.3　小端存储格式**

多数厂家生产的 ARM 核处理器芯片都能够支持大端和小端两种存储格式，默认为小端存储格式。如果一个基于 ARM 核的处理器芯片将存储器系统配置为其中一种格式(如小端)，而实际连接的存储器系统配置为相反的格式(如大端)，那么只有以“字”为单位的指令取指、数据加载和数据保存能够可靠地实现，其他的存储器访问将出现不可预知的结果。

ARM 体系结构通常希望所有的存储器访问能适当地对齐。特别是用于字访问的地址通常要求字对齐，而半字访问的地址通常要求半字对齐。未按这种方式对齐的存储器访问称为非对齐的存储器访问。

ARM 体系结构完成 I/O 功能的标准方法是使用存储器映射 I/O。这种方法采用

特定的存储器地址，当从这些地址加载或向这些地址存储时，它们提供I/O功能。典型情况下，从存储器映射I/O地址加载用于输入，而向存储器地址I/O存储则用于输出。

## 1.2 ARM处理器的工作状态和运行模式

ARM处理器有两种工作状态：当执行32位的ARM指令时工作在ARM状态，当执行16位的Thumb指令时工作在Thumb状态。指令执行过程中处理器可以随时在两种工作状态之间切换，并且工作状态的改变不会影响处理器工作模式和相应寄存器中的内容。ARM指令集和Thumb指令集均有切换处理器状态的指令。

由ARM状态进入Thumb状态的方法：当操作数寄存器的状态位（位0）为1时，执行BX指令可以使处理器从ARM状态切换到Thumb状态。此外，如果处理器在Thumb状态发生异常（如IRQ、FIQ、Undef、Abort、SWI等），则当异常处理返回时，自动切换到Thumb状态。

由Thumb状态进入ARM状态的方法：当操作数寄存器的状态位（位0）为0时，执行BX指令可以使处理器从Thumb状态切换到ARM状态。此外，当处理器进行异常处理（如IRQ、FIQ、Undef、Abort、SWI等）时，将程序计数器PC指针放入异常模式链接寄存器中，并从异常向量地址开始执行程序，也可以使处理器切换到ARM状态。

ARM处理器支持如表1.1所列的7种运行模式。

**表1.1 ARM处理器的运行模式**

| 处理器运行模式 | 说 明 |
|---|---|
| 用户模式(usr) | ARM处理器正常的程序执行状态 |
| 系统模式(sys) | 运行具有特权的操作系统任务 |
| 快中断模式(fiq) | 支持高速数据传输或通道处理 |
| 管理模式(svc) | 操作系统保护模式 |
| 数据访问中止模式(abt) | 用于虚拟存储器及存储器保护 |
| 中断模式(irq) | 用于通用的中断处理 |
| 未定义指令中止模式(und) | 支持硬件协处理器的软件仿真 |

ARM处理器的运行模式可以通过软件改变，也可以通过外部中断或异常处理改变。大多数应用程序运行在用户模式下。当处理器运行在用户模式下时，某些被保护的系统资源是不能被访问的。

除了用户模式和系统模式之外的5种模式又称为异常模式（Exception Modes）。异常模式通常用于处理中断或异常，以及需要访问被保护的系统资源等情况。为了确保从用户模式进入异常模式的可靠性，每种模式都有一些特定的附加寄存器。

系统模式不允许有任何异常进入，它与用户模式有完全相同的寄存器，但不受用户

模式的限制。系统模式供需要访问系统资源的操作系统任务使用，且不使用与异常模式有关的附加寄存器，从而保证了当任何异常出现时操作系统任务的状态都是可靠的。

## 1.3　寄存器组织

### 1.3.1　寄存器分类

ARM 处理器共有 37 个 32 位寄存器，分为如下两类：

- 31 个通用寄存器，包括程序计数器 PC。这些寄存器是 32 位的。
- 6 个状态寄存器，也是 32 位的，但只使用了其中的 12 位。

这些寄存器不能被同时访问，具体有哪些寄存器是可以访问的，取决于处理器的工作状态和运行模式。寄存器被安排成部分重叠的组，每种处理器模式使用不同的寄存器组。

ARM 状态下的寄存器组织如图 1.4 所示。在任何时候，15 个通用寄存器 R0～R14、程序计数器 PC 和程序状态寄存器 CPSR 都是可以访问的。

| 用户 | 系统 | 快中断 | 管理 | 中止 | 中断 | 未定义 |
|---|---|---|---|---|---|---|
| R0 | R0 | R0 | R0 | R0 | R0 | R0 |
| R1 | R1 | R1 | R1 | R1 | R1 | R1 |
| R2 | R2 | R2 | R2 | R2 | R2 | R2 |
| R3 | R3 | R3 | R3 | R3 | R3 | R3 |
| R4 | R4 | R4 | R4 | R4 | R4 | R4 |
| R5 | R5 | R5 | R5 | R5 | R5 | R5 |
| R6 | R6 | R6 | R6 | R6 | R6 | R6 |
| R7 | R7 | R7 | R7 | R7 | R7 | R7 |
| R8 | R8 | R8_fiq | R8 | R8 | R8 | R8 |
| R9 | R9 | R9_fiq | R9 | R9 | R9 | R9 |
| R10 | R10 | R10_fiq | R10 | R10 | R10 | R10 |
| R11 | R11 | R11_fiq | R11 | R11 | R11 | R11 |
| R12 | R12 | R12_fiq | R12 | R12 | R12 | R12 |
| R13 | R13 | R13_fiq | R13_svc | R13_abt | R13_irq | R13_und |
| R14 | R14 | R14_fiq | R14_svc | R14_abt | R14_irq | R14_und |
| R15(PC) | R15(PC) | R15(PC) | R15(PC) | R15(PC) | R15(PC) | R15(PC) |

(a)ARM状态下的通用寄存器与程序计数器

| | | | | | | |
|---|---|---|---|---|---|---|
| CPSR | CPSR | CPSR | CPSR | CPSR | CPSR | CPSR |
| | | SPSR_fiq | SPSR_svc | SPSR_abt | SPSR_irq | SPSR_und |

分组寄存器

(b)ARM状态下的程序状态寄存器

**图 1.4　ARM 状态下的寄存器组织**

Thumb 状态下的寄存器集是 ARM 状态下寄存器集的一个子集，程序可以直接访问 8 个通用寄存器 R7～R0、程序计数器 PC、堆栈指针 SP、链接寄存器 LR 和程序状态寄存器 CPSR。每一种特权模式下都有一组 SP、LR 和 SPSR。图 1.5 所示为 Thumb

状态下的寄存器组织。

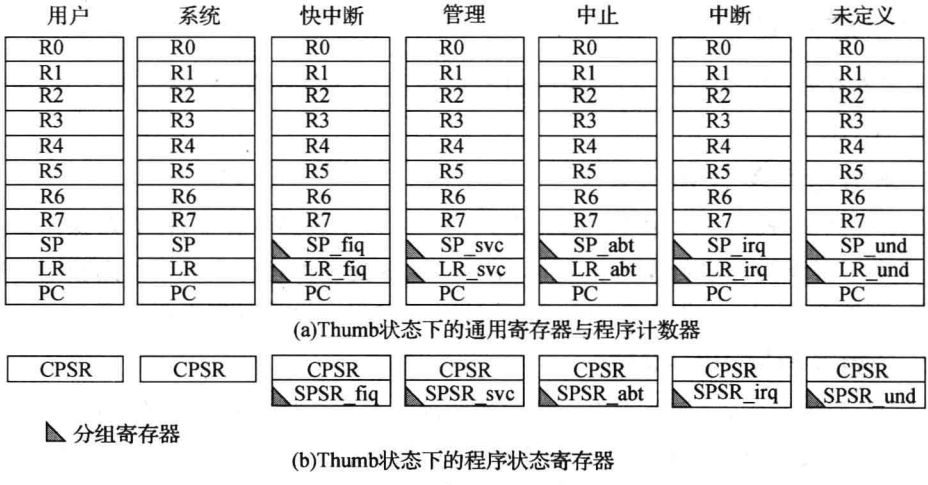



**图 1.5　Thumb 状态下的寄存器组织**

Thumb 状态下的寄存器组织与 ARM 状态下的寄存器组织的映射关系如图 1.6 所示。其中，Thumb 状态下的 R0～R7 与 ARM 状态下的 R0～R7 相同；Thumb 状态下的 CPSR 和 SPSR 与 ARM 状态下的 CPSR 和 SPSR 相同；Thumb 状态下的 SP 映射到 ARM 状态下的 R13；Thumb 状态下的 LR 映射到 ARM 状态下的 R14；Thumb 状态下的程序计数器 PC 映射到 ARM 状态下的 R15。

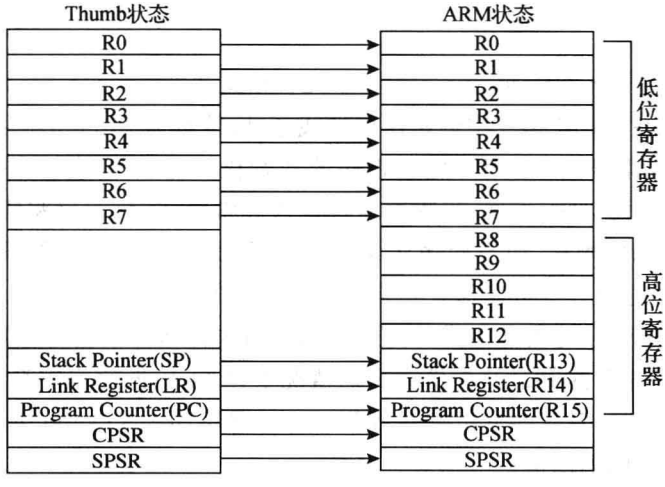



**图 1.6　Thumb 状态寄存器映射到 ARM 状态寄存器**

在 Thumb 状态下，高位寄存器 R8～R15 并不是标准寄存器集的一部分，但可使用汇编语言程序受限制地访问这些寄存器，将其用做快速的暂存器。使用带特殊变量的 MOV 指令，数据可以在低位寄存器和高位寄存器之间进行传送；高位寄存器的值可以

使用 CMP 和 ADD 指令进行比较，或加上低位寄存器中的值。

## 1.3.2　通用寄存器

通用寄存器(R0～R15)包括：未分组寄存器 R0～R7、分组寄存器 R8～14 和程序计数器 PC(R15)。

### 1. 未分组寄存器 R0～R7

在所有的运行模式下，未分组寄存器都指向同一个物理寄存器，没有特殊用途。因此，在中断或异常处理进行模式切换时，由于不同的处理器运行模式均使用相同的物理寄存器，可能会造成寄存器中数据的破坏，这一点在程序设计时应引起注意。

### 2. 分组寄存器 R8～R14

对于分组寄存器，它们每一次所访问的物理寄存器与处理器当前的运行模式有关。若要访问特定的物理寄存器而不依赖于当前的处理器模式，则要使用规定的名字。

对于 R8～R12 来说，每个寄存器对应两个不同的物理寄存器，当使用 fiq 模式时，访问寄存器 R8_fiq～R12_fiq；当使用除 fiq 模式以外的其他模式时，访问寄存器 R8_usr～R12_usr。

对于 R13 和 R14 来说，每个寄存器对应 6 个不同的物理寄存器，其中一个由用户模式和系统模式共用，另外 5 个物理寄存器对应于其他 5 种不同的运行模式。

采用以下的记号来区分不同的物理寄存器：

```
R13_<mode>
R14_<mode>
```

其中，mode 为 usr、fiq、irq、svc、abt、und 几种模式之一。

寄存器 R13 在 ARM 指令中通常用做堆栈指针，简称为 SP。每种运行模式均有自己独立的物理寄存器 R13，在用户应用程序的初始化部分，一般都要将 R13 初始化为指向其运行模式的栈空间，这样，当程序运行进入异常模式时，可以将需要保护的寄存器放入 R13 所指向的堆栈，而当程序从异常模式返回时，则从对应的堆栈中恢复被保护的寄存器，采用这种方式可以保证异常发生后程序的正常执行。

R14 也称作子程序链接寄存器，简称为 LR。当执行 BL(子程序调用)指令时，R14 中得到 R15(程序计数器 PC)的备份。其他情况下，R14 用做通用寄存器。与之类似，当发生中断或异常时，对应的分组寄存器 R14_svc、R14_irq、R14_fiq、R14_abt 和 R14_und 用来保存 R15 的返回值。

在任何一种运行模式下，都可以用 R14 保存子程序的返回地址。当使用指令 BL 或 BLX 调用子程序时，将 PC 的当前值复制给 R14，执行完子程序后，又将 R14 的值复制回 PC，即可完成子程序的调用返回。此外，当异常出现时，相应异常模式的 R14 也可被设置成异常返回地址。异常返回可以与子程序返回类似的方法实现，但使用的指令稍有不同。

### 3. 程序计数器 R15

寄存器 R15 通常用做程序计数器 PC。在 ARM 状态下，R15 的位[1:0]为 0，位[31:2]用于保存 PC 的值；在 Thumb 状态下，R15 的位[0]为 0，位[31:1]用于保存 PC 的值。

R15 虽然也可用做通用寄存器，但一般不这么使用，因为对 R15 的使用有一些特殊限制。当违反了这些限制时，程序的执行结果是未知的。

由于 ARM7 体系结构采用了两级流水线技术，所以对于 ARM 指令集而言，PC 总是指向当前指令的下两条指令的地址，即 PC 的值为当前指令的地址值加 8 字节。

在 ARM 状态下，任一时刻可以访问以上所讨论的 16 个通用寄存器和 1～2 个状态寄存器。在非用户模式(特权模式)下，则可访问特定模式分组寄存器。

## 1.3.3　程序状态寄存器

ARM 处理器包含 1 个当前程序状态寄存器 CPSR 和 5 个程序状态保存寄存器 SPSR。所有运行模式下都可以访问当前程序状态寄存器。每种异常模式都有对应的程序状态保存寄存器，用户模式和系统模式不属于异常，因而没有程序状态保存寄存器。程序状态寄存器的定义如图 1.7 所示，其主要功能包括：

- 保存算术逻辑单元 ALU 中的当前操作信息；
- 控制允许和禁止中断；
- 设置处理器的运行模式。

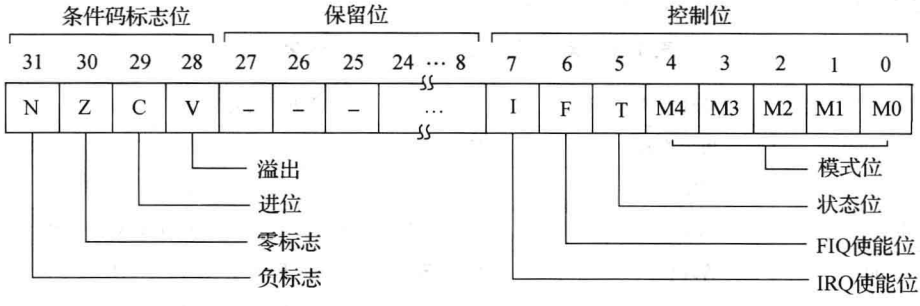



图 1.7　程序状态寄存器

### 1. 条件码标志

图 1.7 中的 N、Z、C、V 均为条件码标志位，它们的内容可以因算术或逻辑运算的结果而改变，并且可以决定某条指令是否被执行。在 ARM 状态下，绝大多数的指令都是有条件执行的。在 Thumb 状态下，仅有分支指令是有条件执行的。

条件码标志位的具体含义如表 1.2 所列。