

ARM公司微控制器系统级设计专家**Joseph Yiu**享誉业界的代表作品!

ARM公司ARM Cortex-M0产品经理**Dominic Pajak**作序!

全球首本系统论述Cortex-M0内核、体系结构、指令集、编译器、程序设计及软件移植的经典译作!



清华

开发者书库



The Definitive Guide to the ARM Cortex-M0

ARM Cortex-M0

权威指南

Joseph Yiu◎著

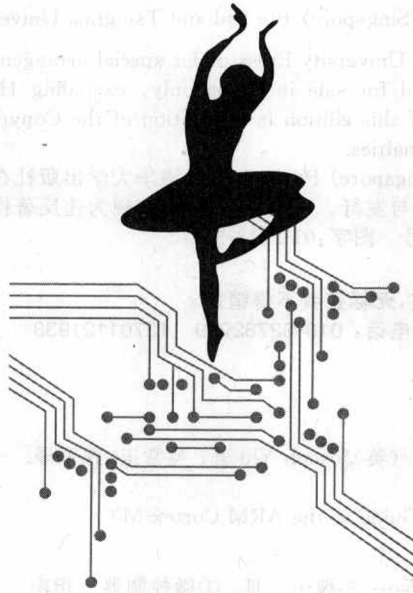
吴常玉 魏军◎译



清华大学出版社

清华

开发者书库



The Definitive Guide to the ARM Cortex-M0

ARM Cortex-M0

Joseph Yiu◎著
吴常玉 魏军◎译

清华大学出版社

北京

The Definitive Guide to the ARM Cortex-M0

Joseph Yiu

ISBN: 9780123854773

Copyright © 2011 by Elsevier. All rights reserved.

Authorized Simplified Chinese translation edition published by the Proprietor.

Copyright © 2013 by Elsevier (Singapore) Pte Ltd and Tsinghua University Press. All rights reserved.

Published in China by Tsinghua University Press under special arrangement with Elsevier (Singapore) Pte Ltd. This edition is authorized for sale in China only, excluding Hong Kong SAR, Macao SAR and Taiwan. Unauthorized export of this edition is a violation of the Copyright Act. Violation of this Law is subject to Civil and Criminal Penalties.

本书简体中文版由 Elsevier (Singapore) Pte Ltd. 授予清华大学出版社在中国大陆地区(不包括香港、澳门特别行政区以及台湾地区)出版与发行。未经许可之出口,视为违反著作权法,将受法律之制裁。

北京市版权局著作权合同登记号 图字:01-2013-3717

本书封底贴有 Elsevier 防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

ARM Cortex-M0 权威指南/(英)Joseph Yiu 著;吴常玉,魏军译. —北京:清华大学出版社,2013.8
(清华开发者书库)

书名原文: The Definitive Guide to the ARM Cortex-M0

ISBN 978-7-302-33004-2

I. ①A… II. ①J… ②吴… ③魏… III. ①微控制器—指南 IV. ①TP332.3-62

中国版本图书馆 CIP 数据核字(2013)第 149365 号

责任编辑:盛东亮

封面设计:李召霞

责任校对:梁毅

责任印制:何芊

出版发行:清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址:北京清华大学学研大厦 A 座 邮 编:100084

社总机:010-62770175 邮 购:010-62786544

投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

课 件 下 载: <http://www.tup.com.cn>, 010-62795954

印 刷 者:北京鑫丰华彩印有限公司

装 订 者:三河市新茂装订有限公司

经 销:全国新华书店

开 本:186mm×240mm

印 张:29

字 数:665千字

版 次:2013年8月第1版

印 次:2013年8月第1次印刷

印 数:1~3000

定 价:69.00元

前言

PREFACE

我是在大学时开始学习微控制器的,那时我使用的单板计算机有些还是 8 位微控制器,程序存储在外部的可擦除可编程的只读存储器中(EEPROM)。EEPROM 使用相对比较大的双列直插封装(DIP),它可以通过玻璃窗里的紫外线擦除。从那时起,微控制器技术发生了很大变化:外部 EEPROM 被片上 Flash 存储器代替,DIP 也变成了表面贴装,而且多数微控制器是在系统可重复编程的。越来越多的外设被加到微控制器中来,软件的复杂度也显著地提高了。

从 2004 年起,微控制器市场发生了很大的变化,之前,市场上的微控制器基本上都是 8 位和 16 位的,32 位微控制器受成本所限,主要用于高端产品。尽管多数 8 位和 16 位微控制器可以使用 C 编程,而试图将所有的所需功能整合到一个小的微控制器中却变得越来越困难。你可能需要 1 天时间来写 C 程序,然后发现由于微控制器的处理速度太慢,无法应对所需的处理任务,你还得花两天时间用汇编重写部分程序。

即便你在开发简单的应用程序,对微控制器的处理能力没有很高的要求,而由于项目的需要,你可能需要偶尔使用另外一种架构的微控制器,这可能会花费一定的工夫。你不但需要花钱购买新的工具,还得用上几周的时间学习使用开发工具,以及数月的时间熟悉新的架构。

2004 年 10 月,ARM7 微控制器的价格降到了 3 美元以内,对于那些需要开发复杂的嵌入式软件的用户来说,这是一个很好的消息。从那时起,随着 Cortex-M3 的推出,ARM 微控制器的价格进一步下降,现在你用 1 美元就足以买到一个 ARM 微控制器了。因此,ARM 微控制器为越来越多的人所接受。除了可以提供极佳的性能以外,现代的 ARM 微控制器具备极低的功耗,已经不再局限于高端应用了。

和许多好的想法一样,Cortex-M0 的理念开始于酒吧里工程师间的对话。一些 ARM 合作伙伴在寻找一种很小的 32 位处理器,这个想法很快就成为了一个成熟的项目(代号为 Swift)。2009 年,Cortex-M0 的设计完成,它很快就成为了最成功的 ARM 处理器产品之一。

通过本书中的例子,你会发现 Cortex-M0 微控制器非常易于使用。在有些方面,由于线性存储器架构的简化,简单却灵活的异常模型,易于理解的调试特性以及 ARM 公司、微控制器和软件方案供应商提供的各种软件程序,它们甚至比 8 位机还要简单。

由于 Cortex-M 处理器是非常 C 友好的,用汇编优化代码是没有必要的,而且编写中断

处理也不用许多特殊的 C 伪指令。对于有些嵌入式开发者来说,切换到 ARM 微控制器也就意味着在微控制器之间的切换将会更加简单,因为他们无须再购买新的开发工具和学习新的架构了。在网上你可以发现许多人已经开始使用 Cortex-M0 微控制器,并开发了許多有趣的项目。

在开发了多个 ARM 处理器的项目之后,我也逐渐获得了一些使用 Cortex-M 处理器的经验(可能还有些白发)。在得到许多朋友的鼓励和帮助后,我决定将这些经验写在一本书中,并且和那些使用 ARM Cortex-M 处理器的嵌入式开发者分享。在写第一本书的时候我学到了很多東西,那本书是关于 Cortex-M3 处理器的。在第一本书出版以后,我收到了很多读者非常有价值的反馈信息(有些甚至不是关于 ARM 的),我对这些读者表示感谢。我知道自己的作品并不完美,但欣慰的是许多读者发现我撰写的有关 Cortex-M3 的书^①的价值,并且期待《ARM Cortex-M0 权威指南》将会更好。

本书面向的读者包括学生、开发爱好者、嵌入式软件开发者、研究人员,甚至是半导体产品工程师;因此,本书涵盖的信息非常广泛,包括大多数嵌入式开发者都会觉得有用的大量高级技术细节。同时,书中还有许多实例,可供嵌入式软件开发新手使用。

希望本书对你有用,并希望你可以在下一个项目中找到使用 Cortex-M0 的乐趣!

Joseph Yiu

^① Joseph Yiu 撰写的经典图书《The Definitive Guide to the ARM Cortex-M3, Second Edition》和《The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors, Third Edition》。——编辑注

目录

CONTENTS

译者序	1
推荐序	3
前言	5
致谢	7
本书约定	9
缩写术语	11
第 1 章 绪论	1
1.1 为什么要选择 Cortex-M0	1
1.1.1 能耗效率	1
1.1.2 代码密度	1
1.1.3 易于使用	2
1.2 Cortex-M0 处理器的应用	2
1.3 ARM 和 ARM 处理器的背景	3
1.4 Cortex-M0 处理器说明和 ARM 体系结构	5
1.5 ARM 处理器和 ARM 生态系统	8
1.6 开始使用 Cortex-M0 处理器	9
1.7 本书的结构和资源	9
第 2 章 Cortex-M0 技术综述	10
2.1 Cortex-M0 处理器简介	10
2.2 ARM Cortex-M0 处理器的特性	11
2.2.1 系统特性	11
2.2.2 应用特性	12
2.2.3 调试特性	12
2.2.4 其他特性	13
2.3 Cortex-M0 处理器的优势	13
2.3.1 能耗效率	13

2.3.2	8位和16位架构的局限性	15
2.3.3	易于使用,软件可移植	16
2.3.4	选择多样化	16
2.4	低功耗应用	16
2.4.1	门数量低	17
2.4.2	高效率	17
2.4.3	低功耗特性	17
2.4.4	逻辑单元提升	17
2.5	Cortex-M0 的软件可移植性	18
第3章	体系结构	19
3.1	概述	19
3.2	系统模型	19
3.2.1	操作模式和状态	19
3.2.2	寄存器和特殊寄存器	20
3.2.3	R0-R12	21
3.2.4	R13,栈指针(SP)	21
3.2.5	R14,链接寄存器(LR)	21
3.2.6	R15,程序计数器(PC)	21
3.2.7	xPSR,组合程序状态寄存器	22
3.2.8	应用程序状态寄存器(APSR)的行为	23
3.2.9	PRIMASK: 中断屏蔽特殊寄存器	24
3.2.10	CONTROL: 特殊寄存器	24
3.3	存储器系统	25
3.4	栈空间操作	26
3.5	异常和中断	27
3.6	嵌套向量中断控制器(NVIC)	28
3.6.1	灵活的中断管理	28
3.6.2	支持嵌套中断	29
3.6.3	向量化的异常入口	29
3.6.4	中断屏蔽	29
3.7	系统控制块(SCB)	29
调试系统		29
3.8	程序映像和启动流程	30

第 4 章 Cortex-M0 编程入门	33
4.1 嵌入式系统编程入门	33
4.1.1 微控制器是如何启动的	33
4.1.2 嵌入式程序设计	34
4.2 输入和输出	37
4.3 开发流程	38
4.4 C 编程和汇编编程	41
4.5 什么是程序映像	42
4.5.1 向量表	42
4.5.2 C 启动代码	44
4.5.3 程序代码	44
4.5.4 C 库代码	45
4.5.5 RAM 中的数据	45
4.6 C 编程：数据类型	46
4.7 用 C 语言操作外设	47
4.8 Cortex 微控制器软件接口标准(CMSIS)	51
4.8.1 CMSIS 介绍	51
4.8.2 CMSIS 中有什么是标准化的	52
4.8.3 CMSIS 的组织结构	52
4.8.4 使用 CMSIS	53
4.9 CMSIS 的优势	55
第 5 章 指令集	57
5.1 ARM 和 Thumb 指令集的背景	57
5.2 汇编基础	59
5.2.1 汇编语法一览	59
5.2.2 后缀的使用	61
5.2.3 Thumb 代码和统一汇编语言(UAL)	62
5.2.4 指令列表	62
5.2.5 处理器内移动数据	63
5.2.6 存储器访问	65
5.2.7 栈空间访问	68
5.2.8 算术运算	69
5.2.9 逻辑运算	73
5.2.10 移位和循环操作	74

5.2.11	展开和顺序反转操作	76
5.2.12	程序流控制	78
5.2.13	存储器屏障指令	80
5.2.14	异常相关指令	81
5.2.15	休眠模式特性相关指令	82
5.2.16	其他指令	83
5.3	伪指令	84
第6章	指令集使用实例	85
6.1	概述	85
6.2	程序控制	85
6.2.1	If-Else	85
6.2.2	循环	86
6.2.3	进一步了解跳转指令	86
6.2.4	跳转条件的典型用法	87
6.2.5	函数调用和函数返回	88
6.2.6	跳转表	89
6.3	数据访问	90
6.3.1	简单数据访问	90
6.3.2	使用存储器访问指令的例子	91
6.4	数据类型转换	93
6.4.1	数据大小的转换	93
6.4.2	大小端转换	94
6.5	数据处理	94
6.5.1	64位/128位加法	94
6.5.2	64位/128位减法	95
6.5.3	整数除法	95
6.5.4	无符号整数开方根	97
6.5.5	位和位域运算	98
第7章	存储器系统	100
7.1	概述	100
7.2	存储器映射	101
7.3	程序存储器, Boot Loader 和存储器重映射	104
7.4	数据存储器	105
7.5	支持小端和大端	106

7.5.1	数据类型	107
7.5.2	硬件行为对编程的影响	108
7.5.3	数据对齐	108
7.5.4	访问非法地址	109
7.5.5	多寄存器加载和存储指令的使用	109
7.6	存储器属性	110
第 8 章	异常和中断	112
8.1	什么是异常和中断	112
8.2	Cortex-M0 处理器的异常类型	112
8.2.1	不可屏蔽中断(NMI)	113
8.2.2	硬件错误	113
8.2.3	SVC(请求管理调用)	113
8.2.4	PendSV(可挂起的系统调用)	113
8.2.5	系统节拍	114
8.2.6	中断	114
8.3	异常优先级定义	114
8.4	向量表	116
8.5	异常流程概述	117
8.5.1	接受异常请求	117
8.5.2	压栈和出栈	117
8.5.3	异常返回指令	118
8.5.4	末尾连锁	118
8.5.5	延迟到达	119
8.6	EXC_RETURN	119
8.7	异常入口流程的细节	121
8.7.1	压栈	121
8.7.2	取出向量并更新 PC	123
8.7.3	寄存器更新	123
8.8	异常退出流程的细节	123
8.8.1	寄存器出栈	123
8.8.2	从返回地址取值并执行	123
第 9 章	中断控制和系统控制	125
9.1	NVIC 和系统控制块特性	125
9.2	中断使能和清除使能	125

9.3	中断挂起和清除挂起	127
9.4	中断优先级	128
9.5	中断控制的通用汇编代码	130
9.5.1	使能和禁止中断	130
9.5.2	设置和清除中断挂起状态	131
9.5.3	设置中断优先级	132
9.6	异常屏蔽寄存器(PRIMASK)	133
9.7	中断输入和挂起行为	134
	简单的中断处理	134
9.8	中断等待	137
9.9	系统异常的控制寄存器	138
9.10	系统控制寄存器	140
9.10.1	CPU ID 基址寄存器	140
9.10.2	应用中断和复位控制寄存器	140
9.10.3	配置和控制寄存器	142
第 10 章	支持操作系统的特性	143
10.1	支持操作系统的特性概述	143
	为什么要使用嵌入式操作系统	143
10.2	SysTick 定时器	145
10.3	SysTick 寄存器	145
10.3.1	设置 SysTick	147
10.3.2	SysTick 用于时间测量	148
10.4	进程栈和进程栈指针	148
10.5	SVC	152
10.6	PendSV	153
第 11 章	低功耗特性	155
11.1	低功耗嵌入式系统概述	155
11.2	Cortex-M0 处理器的低功耗优势	156
11.3	低功耗特性概述	157
11.4	休眠模式	158
11.5	等待事件(WFE)和等待中断(WFI)	160
11.5.1	等待事件(WFE)	160
11.5.2	等待中断(WFI)	161
11.5.3	唤醒条件	162

11.6	退出休眠特性	164
11.7	唤醒中断控制器	165
第 12 章	错误处理	167
12.1	错误异常概述	167
	错误是怎么发生的	167
12.2	分析错误	168
12.3	意外切换至 ARM 状态	169
12.4	实际应用中的错误处理	170
12.5	锁定	172
	12.5.1 锁定的原因	172
	12.5.2 锁定期间发生了什么	173
12.6	防止锁定	173
第 13 章	调试特性	175
13.1	软件开发和调试特性	175
13.2	调试特性一览	176
13.3	调试接口	176
13.4	暂停模式和调试事件	179
13.5	调试系统	180
第 14 章	Keil MDK 入门	182
14.1	Keil MDK 介绍	182
14.2	使用 Keil MDK 的第一步	183
	14.2.1 创建 Blinky 工程	183
	14.2.2 创建工程代码	185
	14.2.3 工程设置	190
	14.2.4 编译和建立程序	194
	14.2.5 使用调试器	195
14.3	其他的工程配置	197
	14.3.1 目标,源文件组	197
	14.3.2 编译器和代码生成选项	198
	14.3.3 模拟器	199
	14.3.4 在 RAM 中运行	201
14.4	定制 Keil 中的启动代码	204
14.5	使用 Keil 中的分散加载特性	204

第 15 章 简单应用程序开发	206
15.1 使用 CMSIS	206
15.2 将 SysTick 用作单发定时器	209
15.3 UART 示例	211
15.3.1 简单的输入和输出	211
15.3.2 重定向	214
15.3.3 开发自己的输入和输出函数	219
15.4 简单中断编程	222
15.4.1 中断编程概述	222
15.4.2 度盘控制接口实例	223
15.4.3 中断控制函数	229
15.5 CMSIS 的不同版本	231
第 16 章 汇编工程和 C 与汇编混合工程	233
16.1 用汇编开发工程	233
16.2 汇编编程的建议规则	233
16.3 汇编函数的结构	235
16.4 简单的汇编工程实例	236
16.5 为变量分配数据空间	244
16.6 用汇编实现 UART	246
16.7 其他的文字输出函数	247
复杂的跳转处理	249
16.8 混合语言工程	249
16.8.1 在汇编中调用 C 函数	250
16.8.2 在 C 代码中调用汇编函数	250
16.9 嵌入汇编	251
16.10 使用特殊指令	252
16.11 习语识别	253
第 17 章 在编程中使用低功耗特性	255
17.1 概述	255
17.2 Cortex-M0 处理器的休眠模式回顾	255
17.3 在程序中使用 WFE 和 WFI	256
17.4 使用挂起发送事件特性	257
17.5 使用退出休眠特性	258

17.6	唤醒中断控制器(WIC)特性	259
17.7	事件通信接口	260
17.8	开发低功耗应用程序	262
17.9	LPC111x 的低功耗特性使用示例	263
第 18 章	使用 SVC、PendSV 和 Keil RTX Kernel	272
18.1	概述	272
18.2	使用 SVC 异常	272
18.3	使用 PendSV 异常	276
18.4	使用嵌入式 OS	278
18.5	Keil RTX 实时内核	278
18.6	OS 启动流程	280
18.6.1	简单的 OS 实例	281
18.6.2	任务间通信	284
18.6.3	事件通信	285
18.6.4	互斥体	287
18.6.5	信号量	288
18.6.6	信箱消息	289
18.6.7	周期时间间隔	292
18.6.8	其他的 RTX 特性	294
18.6.9	应用程序实例	294
第 19 章	ARM RealView 开发组件入门	299
19.1	概述	299
19.2	简单的应用程序实例	300
19.3	使用分散加载文件	303
19.4	用 C 实现的含有向量表的实例	304
19.5	在 RVDS 中使用 MicroLIB	308
19.6	在 RVDS 中使用汇编进行应用程序开发	309
19.7	Flash 编程	311
19.8	使用 RealView 调试器进行调试	312
19.9	使用 RealView 调试器的串行线调试	317
19.10	RVDS 中的重定向	319
第 20 章	GNU C 编译器入门	321
20.1	概述	321

20.2	典型的开发流程	322
20.3	简单的 C 程序开发	323
20.4	CodeSourcery 通用启动代码	326
20.5	使用用户定义的向量表	327
20.6	在 gcc 中使用 Printf	331
20.7	内联汇编	332
20.8	gcc 中的 SVC 实例	333
20.9	硬件错误异常实例	336
20.10	Flash 编程和调试	339
第 21 章	软件移植	340
21.1	概述	340
21.2	ARM 处理器	340
21.3	ARM7TDMI 和 Cortex-M0 之间的差异	341
21.3.1	操作模式	341
21.3.2	寄存器	341
21.3.3	指令集	343
21.3.4	中断	343
21.4	从 ARM7TDMI 向 Cortex-M0 移植软件	343
21.4.1	启动代码和向量表	343
21.4.2	中断	344
21.4.3	C 程序代码	344
21.4.4	汇编代码	345
21.4.5	原子操作	345
21.4.6	优化	345
21.5	Cortex-M1 和 Cortex-M0 之间的差异	346
21.5.1	指令集	346
21.5.2	NVIC	346
21.5.3	系统级特性	346
21.6	在 Cortex-M0 和 Cortex-M1 之间移植软件	346
21.7	Cortex-M3 和 Cortex-M0 之间的差异	347
21.7.1	系统模型	347
21.7.2	NVIC 和异常	347
21.7.3	指令集	349
21.7.4	系统级特性	349
21.7.5	调试特性	350

21.8	在 Cortex-M0 和 Cortex-M3 之间移植软件	350
21.9	在 Cortex-M0 和 Cortex-M4 处理器之间移植软件	351
21.10	从 8 位机/16 位机往 Cortex-M0 移植程序	353
21.10.1	通用改动	353
21.10.2	存储器需求	354
21.10.3	8 位机和 16 位机不再适用的优化	355
21.10.4	实例：从 8051 移植到 ARM Cortex-M0	355
第 22 章	Cortex-M0 产品	358
22.1	概述	358
22.2	微控制器产品和专用标准产品 (ASSP)	358
22.2.1	NXP Cortex-M0 微控制器	358
22.2.2	NuMicro 微控制器	359
22.2.3	Mocha-1 ARM Cortex-M0 可配置阵列	360
22.2.4	Melfas MCS-7000 系列触摸屏控制器	361
22.3	编译器和软件开发组件	361
22.3.1	Keil 微控制器开发套件 (MDK)	361
22.3.2	TASKING VX-Toolset for ARM	362
22.3.3	IAR Embedded Workbench for ARM	362
22.3.4	CrossWorks for ARM	362
22.3.5	Red Suite	362
22.3.6	LabVIEW C 代码生成器	363
22.4	开发板	365
22.4.1	LPCXpresso	365
22.4.2	IAR 的 LPC1114 入门套件	365
22.4.3	LPC1114 Cortex-M0 模块	365
22.4.4	Keil Cortex-M0 开发板	366
附录 A	Cortex-M0 指令集	367
附录 B	Cortex-M0 异常类型快速参考	371
B.1	异常类型	371
B.2	异常压栈后的栈内容	371
附录 C	软件接口标准 (CMSIS) 快速参考	373
C.1	数据类型	373

C.2	异常枚举	373
C.3	NVIC 操作函数	374
C.4	系统和 SysTick 操作函数	376
C.5	内核寄存器操作函数	376
C.6	特殊指令操作函数	377
附录 D NVIC、SCB 以及 SysTick 寄存器快速参考		378
D.1	NVIC 寄存器一览	378
D.2	中断设置使能寄存器(NVIC->ISER)	378
D.3	中断清除使能寄存器(NVIC->ICER)	378
D.4	中断设置挂起寄存器(NVIC->ISPR)	379
D.5	中断清除挂起寄存器(NVIC->ICPR)	379
D.6	中断优先级寄存器(NVIC->IRP[0]到 NVIC->IRP[7])	379
D.7	SCB 寄存器一览	380
D.8	CPU ID 基地址寄存器(SCB->CPUID)	380
D.9	中断控制状态寄存器(SCB->ICSR)	381
D.10	应用中断和控制状态寄存器(SCB->AICR)	381
D.11	系统控制寄存器(SCB->SCR)	382
D.12	配置控制寄存器(SCB->CCR)	382
D.13	系统处理优先级寄存器 2(SCB->SHR[0])	383
D.14	系统处理优先级寄存器 3(SCB->SHR[1])	383
D.15	系统处理控制和状态寄存器	383
D.16	SysTick 寄存器一览	383
D.17	SysTick 控制和状态寄存器(SysTick->CTRL)	384
D.18	SysTick 重装载值寄存器(SysTick->LOAD)	384
D.19	SysTick 当前值寄存器(SysTick->VAL)	384
D.20	SysTick 校准值寄存器(SysTick->CALIB)	385
附录 E 调试寄存器快速参考		386
E.1	概述	386
E.2	内核调试寄存器	386
E.3	断点单元	388
E.4	数据监视点单元	389
E.5	ROM 表寄存器	391