

<<嵌入式系统原理与实践>>

图书基本信息

书名：<<嵌入式系统原理与实践>>

13位ISBN编号：9787121158223

10位ISBN编号：7121158221

出版时间：2012-3

出版时间：电子工业出版社

作者：王宜怀

页数：415

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## <<嵌入式系统原理与实践>>

### 内容概要

《嵌入式系统原理与实践：ARM Cortex-M4

Kinetic微控制器》是国内第一本以ARM Cortex-M4内核的Kinetic微控制器为蓝本来讲述嵌入式系统的图书。

Kinetic系列微控制器将高效的ARM Cortex-M4内核与先进的低功耗设计技术相结合，是业内功耗最低的基于Cortex-M4的MCU解决方案。

全书共15章，其中前4章简要阐述了嵌入式系统的知识体系、学习误区、学习建议和基于硬件构件的嵌入式系统开发方法，给出了ARM Cortex-M4简介及K60硬件最小系统，示例了第一个样例程序及开发环境下的工程组织方法，完成了第一个K60工程的入门任务，并讲解了第一个带中断的实例，前4章囊括了学习一个新MCU完整要素的入门环节；第5章到14章分别给出了GPIO的应用实例（键盘、LED与LCD）、定时器、A/D、D/A、比较器、TSI、SPI、I2C、I2S、Flash、CAN、SDHC、USB、以太网及K60其他模块等；最后一章给出了进一步学习的指导。

《嵌入式系统原理与实践：ARM Cortex-M4

Kinetic微控制器》适用于有关高校嵌入式系统的教学或技术培训资料，也可供ARM Cortex-M4应用工程师作为技术研发参考。

## &lt;&lt;嵌入式系统原理与实践&gt;&gt;

## 书籍目录

## 第1章 概述

## 1.1 嵌入式系统定义、由来及特点

## 1.1.1 嵌入式系统的定义

## 1.1.2 嵌入式系统的由来及其与微控制器的关系

## 1.1.3 嵌入式系统的特点

## 1.2 嵌入式系统的知识体系、学习误区及学习建议

## 1.2.1 嵌入式系统的知识体系

## 1.2.2 嵌入式系统的学习误区

## 1.2.3 基础阶段的学习建议

## 1.2.4 嵌入式系统开发所遇到的若干问题

## 1.3 嵌入式硬件构件的基本思想与应用方法

## 1.4 基于硬件构件的嵌入式系统硬件电路设计

## 1.4.1 设计时需要考虑的基本问题

## 1.4.2 硬件构件化电路原理图绘制的简明规则

## 1.4.3 实验PCB板设计的简明规则

## 1.5 基于硬件构件的嵌入式底层软件构件的编程方法

## 1.5.1 嵌入式硬件构件和软件构件的层次模型

## 1.5.2 底层构件的实现方法与编程思想

## 1.5.3 硬件构件及底层软件构件的重用与移植方法

## 1.6 嵌入式系统的常用术语

## 1.6.1 与硬件相关的术语

## 1.6.2 与通信相关的术语

## 1.6.3 与功能模块及软件相关的术语

## 1.7 本章小结

## 习题

## 第2章 Kinetis微控制器概述与K60硬件最小系统

## 2.1 学习一个新MCU芯片的基本要素

## 2.2 ARM背景知识简介

## 2.2.1 ARM简介

## 2.2.2 Cortex-M4处理器特性简介

## 2.3 Kinetis系列微控制器概述及型号标识

## 2.3.1 Kinetis系列微控制器概述

## 2.3.2 Kinetis系列微控制器型号标识

## 2.4 K60系列微控制器的存储器映像与编程结构

## 2.4.1 K60系列MCU性能概述与内部结构简图

## 2.4.2 K60系列存储器映像

## 2.5 K60的引脚功能与硬件最小系统

## 2.5.1 K60的引脚功能

## 2.5.2 K60的硬件最小系统原理图

## 2.5.3 Kinetis写入器与K60核心板

## 2.5.4 硬件最小系统测试方法

## 2.6 ARM Cortex-M4的寄存器及指令简介

## 2.6.1 ARM Cortex-M4的寄存器简介

## 2.6.2 ARM Cortex-M4的指令系统简介

## 2.7 本章小结

## <<嵌入式系统原理与实践>>

### 习题

#### 第3章 第一个样例程序及工程组织

##### 3.1 GPIO模块的驱动构件设计

###### 3.1.1 GPIO的基础知识

###### 3.1.2 GPIO模块概要与编程要点

###### 3.1.3 GPIO驱动构件设计

##### 3.2 CodeWarrior开发环境简介

##### 3.3 嵌入式设计编码基本规范

###### 3.3.1 硬件驱动构件文件

###### 3.3.2 数据类型

###### 3.3.3 函数

###### 3.3.4 源码文件夹结构

##### 3.4 第一个C语言工程：控制小灯闪烁

###### 3.4.1 Light构件设计

###### 3.4.2 Light构件测试工程主程序

###### 3.4.3 在CW环境下导入样例工程

##### 3.5 理解第一个C工程

###### 3.5.1 CW开发环境下工程文件组织框架

###### 3.5.2 文件说明

###### 3.5.3 芯片上电启动执行过程

##### 3.6 在CW环境下创建一个新的工程

##### 3.7 本章小结

### 习题

#### 第4章 异步串行通信

##### 4.1 异步串行通信的基础知识

###### 4.1.1 基本概念

###### 4.1.2 RS-232C总线标准

###### 4.1.3 电平转换电路原理

##### 4.2 UART模块功能概述

##### 4.3 UART模块的编程结构

##### 4.4 UART模块的底层驱动构件设计

###### 4.4.1 UART构件的函数原型设计

###### 4.4.2 UART构件的头文件 ( bw uan.h )

###### 4.4.3 UART构件的源程序文件 ( hw uart.c )

##### 4.5 以查询方式接收的UART模块测试实例

##### 4.6 以中断方式接收的UART模块测试实例

##### 4.7 本章小结

### 习题

#### 第5章 GPIO的应用实例：键盘、LED与LCD

##### 5.1 键盘模块的驱动构件设计

###### 5.1.1 键盘模型及接口

###### 5.1.2 键盘驱动构件设计I

###### 5.1.3 键盘驱动构件测试实例

##### 5.2 LED模块的驱动构件设计

###### 5.2.1 LED的基础知识

###### 5.2.2 LED驱动构件设计

###### 5.2.3 LED驱动构件测试实例

## <<嵌入式系统原理与实践>>

### 5.3 LCD模块的驱动构件设计

#### 5.3.1 LCD的基础知识

#### 5.3.2 LCD驱动构件设计

#### 5.3.3 LCD驱动构件测试实例

### 5.4 本章小结

#### 习题

### 第6章 定时器相关模块

#### 6.1 计数器 / 定时器的基本工作原理

#### 6.2 可编程延迟模块PDB

##### 6.2.1 PDB的基础知识

##### 6.2.2 PDB模块概要与编程要点

##### 6.2.3 PDB构件设计及测试实例

#### 6.3 Flex定时器FTM

##### 6.3.1 FTM的基础知识

##### 6.3.2 FTM模块概要与编程要点

##### 6.3.3 FTM构件设计及测试实例

#### 6.4 周期中断定时器PIT

##### 6.4.1 PIT的基础知识

##### 6.4.2 PIT模块概要与编程要点

##### 6.4.3 PIT构件设计及测试实例

#### 6.5 低功耗定时器LPTMR

##### 6.5.1 LPTMR的基础知识

##### 6.5.2 LPTMR模块概要与编程要点

##### 6.5.3 LPTMR构件设计及测试实例

#### 6.6 载波调制发射器 (CMT)

##### 6.6.1 CMT的基础知识

##### 6.6.2 CMT模块概要与编程要点

##### 6.6.3 CMT构件设计及测试实例

#### 6.7 实时时钟

##### 6.7.1 RTC基础知识

##### 6.7.2 RTC模块概要与编程要点

##### 6.7.3 RTC构件设计及测试实例

### 6.8 本章小结

#### 习题

### 第7章 A/D、D/A、CMP和TSI模块

### 第8章 SPI、I2C与I2S

### 第9章 Flash在线编程

### 第10章 K60的CAN总线开发方法

### 第11章 大容量SD存储卡SDHC

### 第12章 USB2.0编程

### 第13章 基于K60的嵌入式以太网

### 第14章 系统时钟与其他功能模块

### 第15章 进一步学习指导

### 参考文献

## 章节摘录

版权页：插图：远程帧不是必需的，例如应用层协议DeviceNet中未用远程帧，但并未影响DeviceNet在可靠运行、通信效率方面的性能，且DeviceNet也是国际流行的CAN应用层协议。

3. 错误帧 错误帧由CAN控制器的硬件进行处理，与用户编程无关。

下面概要说明发送错误帧的工作机制。

CAN节点通过发送引脚发送报文时，接收引脚也在同步接收报文，当发送报文的ACK场为“1”时，接收到的应答间隙（ACK Slot）一定要是“0”才代表发送成功。

在CAN总线网络中只要有一个节点正确接收到了报文，并将发送节点的应答间隙写为“0”，则发送节点就认为发送数据成功。

在报文的应答过程中，若某一节点检测到错误，则它会立刻发送错误帧，一般是发送连续的6个0或1。由CAN的位填充原理可知，当有五个连续的0或1出现时，为了传送中的同步，必须插入一个反型位作为填充。

因此如果连续出现6个或6个以上的0或1，则此次传送错误，报文将被丢弃。

此时当发送节点收到这个错误帧后，便知道发送出错，并试图重发报文。

任何节点检测到总线错误都会发送错误帧。

错误帧由两个不同的场组成。

第一个场是由不同节点提供的错误标志（FLAG）的叠加；第二个场是错误界定符。

错误帧的组成如图10—10所示。

错误标志有两种形式：主动错误（Error Active）标志和被动错误（Error Passive）标志。

主动错误标志由6个连续的0位组成，而被动错误标志由6个连续的1位组成。

检测到错误条件的“主动错误”的节点通过发送主动错误标志指示错误。

错误标志的形式破坏了从帧起始到CRC界定符的位填充的规则，或者破坏了ACK场或帧结束的固定形式。

所有其他的节点由此检测到错误条件并与此同时开始发送错误标志。

因此，6个连续“0”的序列导致一个结果，这个结果就是把个别节点发送的不同的错误标志叠加在一起。

这个序列的总长度最小为6位，最大为12位。

检测到错误条件的“错误被动”的节点试图通过发送被动错误标志指示错误。

“被动错误”的节点等待6个相同极性的连续位（这6个位处于被动错误标志的开始）。

当这6个相同的位被检测到时，被动错误标志的发送就完成了。

## <<嵌入式系统原理与实践>>

### 编辑推荐

《嵌入式系统原理与实践:ARM Cortex-M4 Kinetis微控制器》适用于有关高校嵌入式系统的教学或技术培训资料，也可供ARM Cortex—M4应用工程师作为技术研发参考。

<<嵌入式系统原理与实践>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>