

<<项目驱动>>

图书基本信息

书名：<<项目驱动>>

13位ISBN编号：9787512404922

10位ISBN编号：7512404921

出版时间：2011-7

出版时间：北京航空航天大学

作者：周立功

页数：286

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>

<<项目驱动>>

内容概要

这本《项目驱动——单片机应用设计基础》由周立功主编陈明计等编著，全书以80C51单片机为主，通过项目驱动的方法融合相关知识点。

内容主要分两部分：第一部分为第1~4

章，以SDCC51编程语言为基础，深入浅出地介绍如何使用嵌入式C编程来控制单片机各种外设部件，并给出常用C编程算法。

第二部分为第5~7章，重点介绍TinyOS51嵌入式多任务操作系统的基本原理，及其在80C51单片机中的实现，并针对同一工程项目给出使用前后台程序和嵌入式多任务操作系统两种不同的编程方法。

通过这两种方法的比较，可使读者了解嵌入式多任务操作系统在项目编程中的优势。

本书注重在教学中强化学生的动手训练，强调理论与实践相结合。

读者通过本书的学习，可熟练掌握嵌入式C的编程方法，并初步掌握嵌入式多任务操作系统的编程知识。

《项目驱动——单片机应用设计基础》可作为大学本科、高职高专电子信息、自动化、机电一体化、计算机等专业的教材，也可作为电子爱好者的自学用书，还可作为从事单片机应用开发工程技术人员的参考资料。

<<项目驱动>>

书籍目录

第1章 深入理解嵌入式C

1.1 概述

1.1.1 特性

1.1.2 引脚排列与描述

1.1.3 特殊功能寄存器

1.2 单片机最小系统与开发工具

1.2.1 Tiny51核心模块

1.2.2 复位电路

1.2.3 晶体振荡电路

1.2.4 单片机在线仿真与编程

1.3 SDCC扩展

1.3.1 SDCC简介

1.3.2 应用示例

1.3.3 关键字与数据类型

1.4 存储器类语言

1.4.1 存储类型

1.4.2 存储模式

1.4.3 特殊功能寄存器数据类型

1.4.4 位数据类型

1.4.5 存储器绝对寻址

1.4.6 指针

1.5 函数

1.5.1 函数参数和局部变量

1.5.2 覆盖

1.5.3 使用专用寄存器组

1.6 深入理解嵌入式C

1.6.1 概述

1.6.2 方法

1.6.3 函数调用与参数传递

1.6.4 函数返回

1.6.5 局部变量存储

1.7 经典范例程序设计

1.7.1 LED流水灯范例

1.7.2 蜂鸣器驱动范例

1.7.3 数码管动态扫描显示驱动范例

1.7.4 键盘动态扫描驱动范例

第2章 特殊功能部件与外设

2.1 中断系统

2.1.1 中断概念

2.1.2 80C51的中断结构

2.1.3 相关寄存器

2.1.4 中断向量

2.1.5 中断操作

2.1.6 使能和禁止中断

2.2 定时/计数器

<<项目驱动>>

- 2.2.1 相关寄存器
- 2.2.2 定时/计数器模式
- 2.2.3 定时器查询延时
- 2.2.4 定时器中断延时
- 2.2.5 无源蜂鸣器驱动程序
- 2.2.6 数码管动态扫描演示程序
- 2.2.7 测量负脉冲
- 2.3 看门狗
 - 2.3.1 看门狗的作用
 - 2.3.2 看门狗的工作原理
 - 2.3.3 看门狗定时器的结构
 - 2.3.4 寄存器描述
 - 2.3.5 看门狗周期值设置
 - 2.3.6 应用示例
- 2.4 I2C总线及其驱动程序
 - 2.4.1 I2C简介
 - 2.4.2 决策
 - 2.4.3 软件接口
 - 2.4.4 基本时序代码
 - 2.4.5 外部接口代码
 - 2.4.6 E2PROM读/写范例
 - 2.4.7 CAT1024驱动程序
 - 2.4.8 温度的测量
- 2.5 串行口及其驱动程序
 - 2.5.1 硬件基础
 - 2.5.2 决策
 - 2.5.3 软件接口
 - 2.5.4 初始化
 - 2.5.5 发送数据
 - 2.5.6 接收数据
 - 2.5.7 测试用例
- 第3章 数据结构与计算方法初步
 - 3.1 简单阈值控制算法
 - 3.1.1 算法原理
 - 3.1.2 应用实例
 - 3.2 循环队列
 - 3.2.1 队列的逻辑结构和基本运算
 - 3.2.2 队列的存储结构
 - 3.2.3 循环队列的运算
 - 3.3 常用检错算法
 - 3.3.1 奇偶校验
 - 3.3.2 和校验
 - 3.3.3 循环冗余校验
 - 3.4 应用实例
 - 3.4.1 Hex文件
 - 3.4.2 通信编程
- 第4章 保险箱密码锁控制器(方案一)

<<项目驱动>>

4.1 概述

4.1.1 保险箱

4.1.2 锁芯机械结构

4.1.3 密码锁控制器

4.1.4 密码锁工作原理

4.2 准备工作

4.2.1 概述

4.2.2 使用说明

4.2.3 硬件概要设计

4.2.4 软件概要设计

4.3 硬件驱动设计

4.3.1 延时驱动

4.3.2 锁驱动

4.3.3 可复用的硬件驱动

4.4 虚拟驱动设计

4.4.1 虚拟锁驱动

4.4.2 虚拟键盘驱动

4.4.3 虚拟蜂鸣器驱动

4.4.4 虚拟显示器驱动

4.4.5 虚拟存储器驱动

4.5 主程序设计

4.5.1 准备工作

4.5.2 编写代码

4.6 直流电机及其功率接口

4.6.1 概述

4.6.2 直流电机的工作原理

4.6.3 直流电机的单向驱动

4.6.4 直流电机的双向驱动

第5章 TinyOS51嵌入式操作系统微小内核

5.1 基础知识

5.1.1 概述

5.1.2 头文件

5.1.3 变量命名规则

5.1.4 范例分析

5.1.5 setjmp与longjmp的实现

5.2 最简单的多任务模型

5.2.1 双任务切换模型

5.2.2 待解决的问题

5.2.3 setTaskJump()的实现

5.2.4 任务切换模型范例分析

5.3 协作式多任务操作系统

5.3.1 整体规划

5.3.2 任务控制块

5.3.3 内部变量初始化

5.3.4 创建任务

5.3.5 启动多任务环境

5.3.6 任务切换

<<项目驱动>>

5.3.7 删除任务

5.3.8 小结

5.4 时间片轮询多任务操作系统

5.4.1 概述

5.4.2 整体规划

5.4.3 任务控制块

5.4.4 内部变量初始化

5.4.5 创建任务

5.4.6 启动多任务环境

5.4.7 任务调度

5.4.8 时钟节拍中断

5.4.9 longjmpInIsr()

5.4.10 任务延时

5.4.11 删除任务

5.5 信号量

5.5.1 概述

5.5.2 整体规划

5.5.3 任务控制块

5.5.4 内部变量初始化

5.5.5 信号量定义

5.5.6 创建信号量

5.5.7 获得信号量

5.5.8 发送信号量

5.5.9 删除任务

5.6 消息邮箱

5.6.1 概述

5.6.2 整体规划

5.6.3 任务标志与消息邮箱

5.6.4 创建消息邮箱

5.6.5 获得消息

5.6.6 发送消息

第6章 程序设计基础

6.1 任务设计

6.1.1 任务的分类

6.1.2 任务的划分

6.2 系统函数使用概述

6.2.1 系统函数总览

6.2.2 中断服务程序调用函数的限制

6.2.3 系统函数的分类

6.3 系统函数的使用场合

6.3.1 时间管理

6.3.2 资源同步

6.3.3 行为同步

6.4 时间管理

6.5 临界区

6.6 信号量

6.6.1 简介

<<项目驱动>>

6.6.2 信号量的工作方式

6.6.3 任务同步中断服务程序

6.6.4 任务间同步

6.6.5 资源同步

6.7 消息邮箱

6.7.1 简介

6.7.2 消息邮箱的工作方式

6.7.3 中断服务程序与任务通信

6.7.4 任务间数据通信

第7章 保险箱密码锁控制器(方案二)

7.1 软件开发流程

7.2 决策

7.2.1 概述

7.2.2 总体目标

7.2.3 使用说明

7.2.4 限制条件

7.2.5 具体开发目标

7.2.6 其他决策内容

7.3 模块划分

7.3.1 概述

7.3.2 硬件层

7.3.3 设备驱动层

7.3.4 虚拟设备层

7.3.5 应用层

7.4 接口定义

7.4.1 密码的输出、存储与显示

7.4.2 应用层接口

7.4.3 虚拟设备层接口

7.4.4 设备驱动层接口

7.5 编写代码

7.5.1 概述

7.5.2 可复用的驱动

7.5.3 I2C驱动

7.5.4 CAT1024驱动

7.5.5 虚拟键盘驱动

7.5.6 虚拟蜂鸣器驱动

7.5.7 人机交互程序

7.5.8 主程序

7.6 测试、验收与小结

参考文献

<<项目驱动>>

章节摘录

版权页：插图： 当串行口接收到或发送完一个字符时，将中断标志RI / TI置位。

中断标志置位并不意味着CPU即刻响应中断，若要CPU产生中断响应还必须将中断使能寄存器IE（见表2.1）中的总中断使能位EA和相对应的中断使能位置位。

一个中断源产生中断请求（即该中断源的中断标志置位）后，若该中断使能（IE寄存器的总中断使能位EA和单独中断使能位均置位），且CPU没有处理同优先级中断或更高优先级的中断，则该中断将立即得到响应。

此时，正常的程序流程将被打断，程序计数器Pc将指向该中断源的中断向量地址来执行中断服务程序，中断服务程序执行完毕将执行中断返回指令RETI，返回中断前被打断的正常程序流程处继续执行下面的程序，如图2.2所示。

若CPU正在处理同优先级中断或更高优先级中断，则要等到同优先级中断或更高优先级中断处理完毕才能响应这个中断（得到响应）。

因中断可能发生在正常程序流程的任何地方，故在中断服务程序执行完毕，为了使程序能正确返回到发生中断的地方继续执行，在进入中断服务程序之前，CPU自动将当前的PC值压入堆栈。

当中断返回时，只要执行RETI指令，CPU就能将之前压入的返回地址弹出到PC，继续执行正常程序流程。

必须在中断退出之前清除中断源的中断请求标志，否则会引发重复中断。

在表2.3中所有的中断请求标志都是由硬件置位。

编号为0～3的中断源的中断请求标志在进入中断响应时由硬件清除，编号为4的串行口中断请求标志RI或TI必须要通过软件清除。

在图2.2中，正常程序流程可以是主程序中的程序，也可以是正在执行的较低优先级的中断服务程序。

在正常程序流程和中断服务程序中，可能都要用到一些CPU的公共资源，比如，累加器A、B、PSW、DPTR和R0～R7。

一旦CPU的硬件收到中断的信号，它就会自动地开始所有的工作，推进CPU的状态，读中断数，从RAM中提取向量，然后启动ISR。

此时，作为程序员必须将“断点”处的信息保存到堆栈中，当程序恢复运行时，将保存在堆栈中的“信息”再恢复到CPU的寄存器中，在“断点”处作为初始数据接着运行。

在用汇编语言编程时，执行中断服务程序之前，须将这些公共资源入栈（PUSH），这叫做“现场保护”；当中断服务程序执行完毕，在执行RETI指令之前，将这些公共资源出栈（POP），这叫做“现场恢复”。

R0～R7也可通过改变PSW寄存器中的RS0和RS1位来切换通用寄存器组而不必入栈。

若用C51语言编程，编译器可自动处理压栈和出栈而不用软件干预，通用寄存器组的切换也可用关键字using实现。

<<项目驱动>>

编辑推荐

<<项目驱动>>

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>