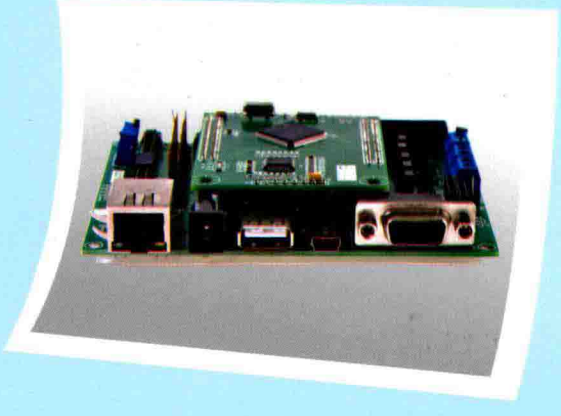
普通高等教育“十二五”规划教材

基于STM32的嵌入式系统设计

JIYU STM32 DE QIANRUSHI XITONG SHEJI

刘　一　主编



中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

普通高等教育“十二五”规划教材

基于STM32的嵌入式系统设计

刘　一　主　编

范君闯　白　娜　副主编

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书介绍了以 ARM Cortex-M3 为内核的 STM32F103 增强型微控制器，深入讲解其硬件和软件设计方法。全书内容包括：ARM Cortex-M3 内核结构，开发环境与最小系统，STM32 固件库，时钟控制系统，向量中断控制器，系统定时器（SysTick），通用、复用及重映射 I/O，外部中断输入，USART，SPI，通用定时器，ADC，看门狗等，并通过综合实例，详细讲解了嵌入式以太网串口服务器的设计。

本书适合作为普通高等院校嵌入式系统设计课程的教材，也可作为高校师生课程设计、毕业设计以及电子设计竞赛的指导教材。

图书在版编目（CIP）数据

基于 STM32 的嵌入式系统设计 / 刘一主编. — 北京：中国铁道出版社，2015.9
普通高等教育“十二五”规划教材
ISBN 978-7-113-20544-7

Ⅰ. ①基… Ⅱ. ①刘… Ⅲ. ①微控制器－系统设计－高等学校－教材 Ⅳ. ①TP332.3

中国版本图书馆 CIP 数据核字(2015)第 173669 号

书　　名：基于 STM32 的嵌入式系统设计
作　　者：刘　一　主编

策　　划：潘星泉　　读者热线：400-668-0820
责任编辑：潘星泉　彭立辉
封面设计：付　巍
封面制作：白　雪
责任校对：汤淑梅
责任印制：李　佳

出版发行：中国铁道出版社（100054，北京市西城区右安门西街 8 号）
网　　址：http://www.51eds.com
印　　刷：河北新华第二印刷有限责任公司
版　　次：2015 年 9 月第 1 版　　2015 年 9 月第 1 次印刷
开　　本：787mm×1 092mm　1/16　印张：16.75　字数：408 千
书　　号：ISBN 978-7-113-20544-7
定　　价：33.00 元

FOREWORD

前 言

STM32 系列微控制器是近年来迅速兴起的基于 ARM Cortex-M3 内核的高端 32 位微控制器的代表。其中，STM32F103 微控制器，工作频率为 72 MHz，内置高速存储器、丰富的增强型 I/O 端口和连接到两条 APB 总线的外设。优秀的性能、丰富的外设以及低廉的价格等优点，使其在工业控制、消费电子、汽车电子、安防监控等众多领域得到了广泛应用。本书就是基于 STM32F103 微控制器介绍嵌入式系统开发的。

对于初学者而言，特别是只有少数 8 位单片机开发经验的人来说，跨入 STM32 这扇大门，开发方式发生了较大的改变。这里的“改变”包括：开发环境的改变、开发工具的改变、工程结构的改变和调试手段的改变。学习 STM32 时，建议按如下几步进行：第一步，收集阅读资料，资料包括 STM32 书籍、官方的芯片文档和库函数文档。但是，不管看书籍还是文档，不要奢求一下都能理解、记住，只需理解基本内容，对复杂的内容有一个初步的印象，以后碰到问题的时候，知道需要哪方面的知识，然后查阅资料或上网找答案。第二步，选购一个例程比较丰富的开发板，不用买最贵的，只要有比较丰富的配套例程即可。按照例程，把开发板上的相关测试、操作步骤，都动手做一遍，以便熟悉开发软件的使用。先看例程的效果，再去读例程代码，理解为何这样写，不理解的地方查书、查资料。然后，参照资料开始改动例程，编译下载，查看效果是否达到自己的设想。STM32 的外设模块特别多，学习要有先后顺序与侧重点。GPIO、USART、TIMER、NVIC 和 ADC 是最常用的功能模块，要非常熟悉。其他的如 USB、DMA 等较难理解的模块可以在以后用到的时候再深入学习。

关于是学习基于寄存器编程还是基于库函数编程这个问题，笔者的看法是先学习基于库函数的编程更容易。可能有很多同学停留在对 8 位单片机的认识上，认为代码里看不到对寄存器的直接设置，很不安心。这个观念要转变，其实大家当初在学习 C 语言的时候，哪里看到寄存器了？我们要理解掌握 C 语言的精髓，包括结构体、枚举和函数调用等，这些是官方库函数的基础，其中定义了大量的结构体数据类型和枚举数据类型，提供了大量具有某个功能的函数。我们要知道这个函数的功能，如何调用，参数是什么，返回值是什么。

基于以上认识，我们编写本书主要是给大家提供一本入门的参考资料。本书提供了大部分功能模块的寄存器说明、主要的库函数说明和使用此功能模块的配置步骤；每个模块都提供了一个基于库函数的简单实例和一个基于寄存器的实例。这些实例都是两个复杂应用——基于STM32的智能充电器和基于STM32的智能家居的部分功能。最后一章提供了几个复杂的应用实例。我们的设想是：通过对配置步骤和实例的学习，使读者快速掌握使用库函数编写代码的方法；通过查阅本书能够找到大部分寄存器和库函数的使用说明。本书的读者需要具有一定的C语言、单片机基础。

本书由刘一（广东技术师范学院）任主编，范君闯、白娜（哈尔滨石油学院）任副主编。具体编写分工如下：刘一编写第5章～第8章、第10章～第14章；范君闯编写第1章～第3章；白娜编写第4章和第9章。

本书适合作为普通高等院校嵌入式系统设计课程的教材，也可作为高校师生课程设计、毕业设计以及电子设计竞赛的培训和指导教材，还可作为嵌入式开发人员的参考书。

由于时间仓促，编者水平有限，书中疏漏与不妥之处在所难免，恳请专家和读者批评指正。

编　者

2015年4月

目 录

CONTENTS

第 1 章 ARM Cortex-M3 内核结构

本章简单介绍了 ARM Cortex-M3 的内核，Cortex-M3（简称 CM3）处理器的结构，存储器系统存储器映射。

1.1 ARM Cortex-M3 内核简介

ARM 内核自推出以后，就迅速占领了全球相当大的市场份额，和以往传统的单片机内核相比，ARM 内核的以下特点使其在单片机内核领域得以迅速发展：

（1）Thumb-2 指令集架构（ISA）的子集，包含所有基本的 16 位和 32 位 Thumb-2 指令。

（2）哈佛处理器架构，在加载/存储数据的同时能够执行指令取指，周期大大缩短，执行效率更加高效。

（3）带分支预测的三级流水线。

（4）运算能力强，具有单时钟周期硬件 32 位乘法、除法。

（5）硬件除法。

（6）Thumb 状态和调试状态。

（7）处理模式和线程模式。

（8）ISR 的低延迟进入和退出。

（9）可中断-可继续（Interruptible-Continued）的 LDM/STM、PUSH/POP。

（10）支持 ARMv6 类型 BE8/LE。

（11）支持 ARMv6 非对齐访问。

为了增强和扩展指令系统的能力，ARM 内核同时支持 ARM 指令和 Thumb 指令，ARM 指令状态下，指令长度是 32 位；Thumb 指令状态下指令长度是 16 位；而且，两种指令状态可以动态切换；Thumb 指令是 ARM 指令的一个子集，使用 Thumb 指令对于缩减代码容量起到了很好的作用。

1.2 处理器的组件

图 1.1 所示为 Cortex-M3 处理器的结构，在 Cortex-M3 内核的单片机的开发应用中，各种外围设备（简称外设）都需要进行时钟配置，时钟配置会涉及 APB 总线的问题，在一些对时间要求比较高的算法中，也会涉及指令总线和数据总线速度的问题，总线时钟频率越高，系统处理速度越快。下面解析一下各种总线及其作用。

首先，先简单介绍一下 AHB 总线和 APB 总线：AHB（Advanced High Performance Bus）即高性能高级总线，一般速度最快；APB（Advanced Peripheral Bus）即高级外围总线，一般通过转换桥桥挂在 AHB 总线上，速度一般低于 AHB 总线。处理器的取指。运算等一般都是挂在 AHB 总线上，保证速度最快；GPIO、ADC 等外设一般挂在 APB 总线上。

I-Code 总线：I 指的就是指令 Instruction，I-Code 就是 32 位指令总线，每次取 32 位指令，所以在 Thumb 状态下，一次可以取 2 条指令。

D-Code 总线：D 指的就是数据 Data，D-Code 就是数据总线，同样也是 32 位宽度；尽管在开发中可以使用非 4 字节对齐方式，但是在数据总线取数据时一定是 4 字节对齐，因此，32 位的数据变量在处理过程中更加高效。

系统总线：负责指令和数据的传送，也是 32 位宽度。外部私有外设总线：这是一条基于 APB 总线的 32 位宽度总线，负责外设之间的访问。

调试端口总线：作为调试端口，方便用户开发调试。

在实际开发应用中，我们比较关心的是 AHB 和 APB，各种总线接口基本都直接或者间接挂接在这两条总线上。通俗地讲，AHB 总线的时钟速率决定了处理器指令处理速度，APB 决定了外设交互的速度。

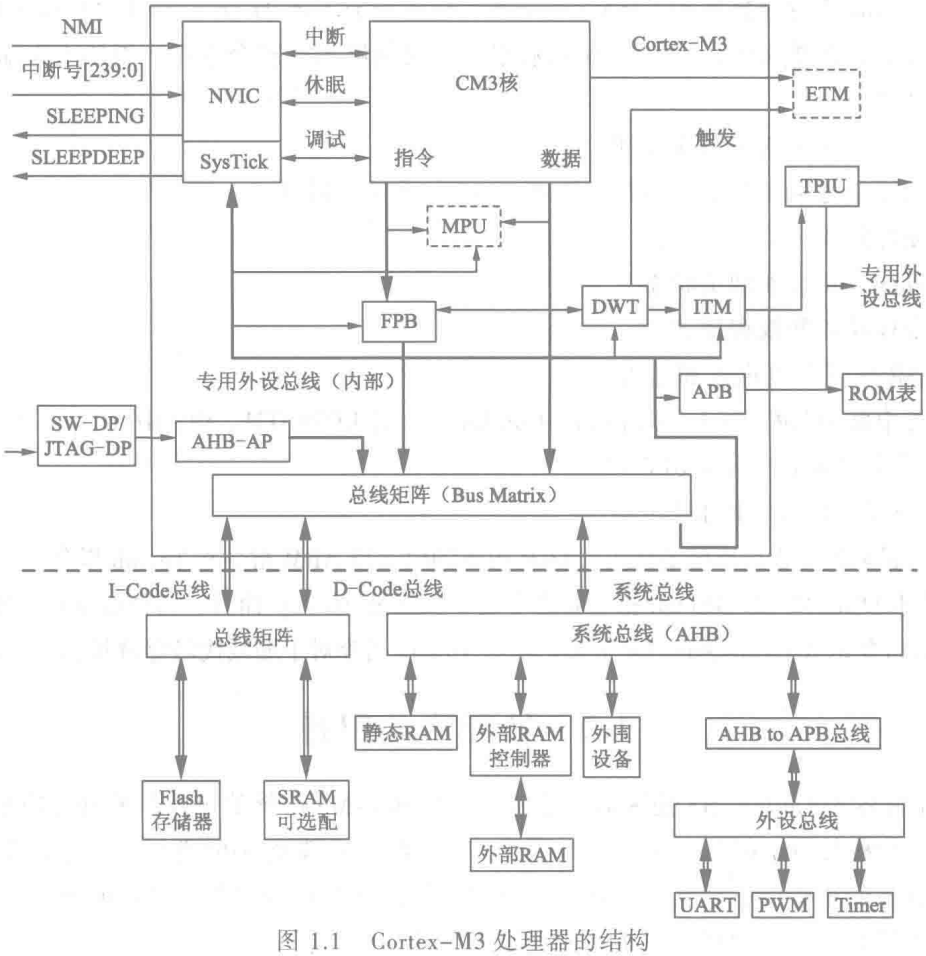


图 1.1　Cortex-M3 处理器的结构

MPU 和 ETM 并不是在所有的 M3 内核中都存在，这两个不是必需组件。图 1.1 中各项缩略语的说明如表 1.1 所示。

表 1.1　缩略语说明

缩　写	含　义
NVIC	嵌套向量中断控制器
SysTick Timer	系统时钟，用于提供时基，多为操作系统所使用
MPU	存储器保护单元（可选）
Bus Matrix	内部的 AHB 总线矩阵
AHB to APB 总线	AHB 到 APB 的总线桥
SW - DP/JTAG - DP	JTAG 调试端口
AHB - AP	AHB 访问端口，串行线/SWJ 接口的命令转换成 AHB 数据传送
ETM	嵌入式跟踪宏单元（可选），调试用。用于处理指令跟踪
DWT	数据观察点及跟踪单元，调试用。这是一个处理数据观察点功能的模块
ITM	指令跟踪宏单元
TPIU	跟踪单元的接口单元。所有跟踪单元发出的调试信息都要先送给它，再转发给外部跟踪捕获硬件
FPB	Flash 地址重载及断点单元
ROM 表	配置信息表

1.3　存储器系统

Cortex-M3 采用了固定的存储映射结构，如图 1.2 所示。Cortex-M3 的地址空间是 4 GB，程序可以在代码区，内部 SRAM 区以及外部 RAM 区中执行。但是因为指令总线与数据总线是分开的，最理想的是把程序放到代码区，从而使取指和数据访问各自使用自己的总线。

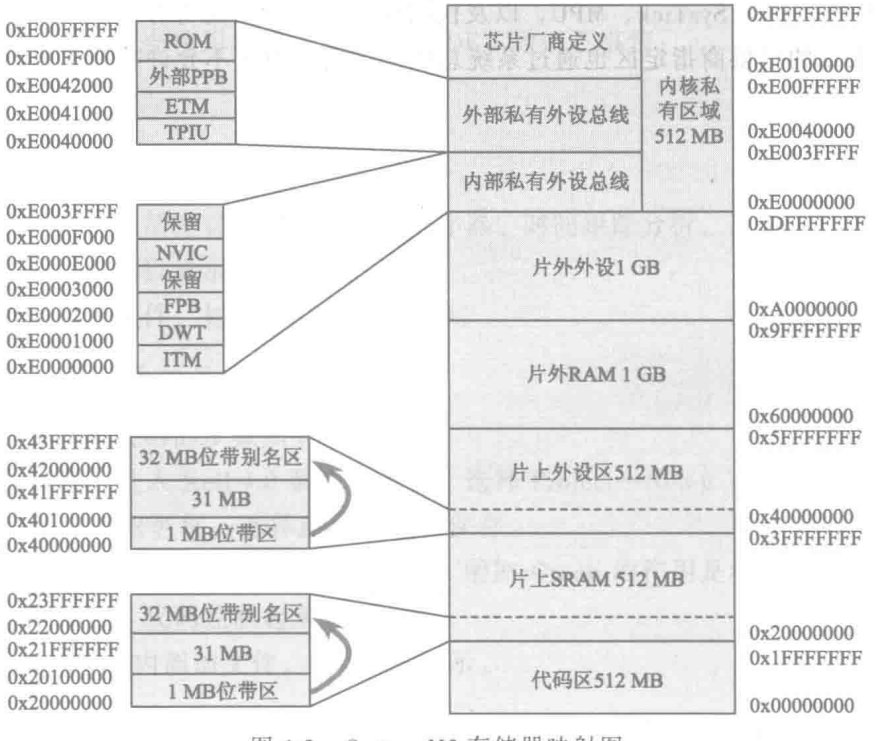


图 1.2　Cortex-M3 存储器映射图

内部 SRAM 区的大小是 512MB，用于让芯片制造商连接片上的 SRAM，这个区通过系统总线来访问。在这个区的下部，有一个 1 MB 的区间，被称为“位带区”。该位带区还有一个对应的 32 MB 的“位带别名区”，容纳了 8×2^{20} 个“位变量”（8051 的只有 128 个位变量），位带区对应的是最低的 1 MB 地址范围，而位带别名区中的每个字对应位带区的一个比特。位带操作只适用于数据访问，不适用于取指。通过位带的功能，可以把多个布尔型数据打包在单一的字中，且依然可以从位带别名区中，像访问普通内存一样使用它们。位带别名区中的访问操作是原子的，消灭了传统的“读—改—写”三步。

片上外设区对应 512 MB 的空间，芯片上所有与外围设备相关的寄存器都位于该区域。这个区中也有一条 32 MB 的位带别名区，以便于快捷地访问外设寄存器，用法与内部 SRAM 区中的位带相同。例如，可以方便地访问各种控制位和状态位。要注意的是，片上外设区内不允许执行指令。通常半导体厂商就是修改此区域的片上外设，来达到各具特色的、个性化的设备。

还有两个 1 GB 的范围，分别用于连接外部 RAM 和外围设备，它们之中没有位带。两者的区别在于外部 RAM 区允许执行指令，而外围设备区则不允许。最后，还剩下 512 MB 的区域，包括了系统级组件、内部私有外设总线、外部私有外设总线，以及由厂商定义的系统外设。

私有外设总线有两条：AHB 私有外设总线，只用于 CM3 内部的 AHB 外设，它们是：NVIC、FPB、DWT 和 ITM。APB 私有外设总线，既用于 CM3 内部的 APB 设备，也用于外围设备（这里的“外围”是对内核而言）。CM3 允许器件制造商再添加一些片上 APB 外设到 APB 私有总线上，它们通过 APB 接口来访问。NVIC 所处的区域叫作“系统控制空间（SCS）”，在 SCS 中除了 NVIC 外，还有 SysTick、MPU，以及代码调试控制所用的寄存器。

最后，未用的提供商指定区也通过系统总线来访问，但是不允许在其中执行指令。

第 2 章　开发环境与最小系统

本章首先介绍开发 CM3 芯片需要的系统平台和编译软件，编写、编译和调试的方法与相关设置。然后，介绍 STM32 芯片工作所需要的最小硬件电路组成。

2.1　开 发 环 境

系统平台及开发软件如下：

（1）系统平台：Windows XP、Windows 7。

（2）开发软件：MDK-ARM（Keil 4.0 以上）。

Keil 是 ARM 公司旗下的编译器，其软件操作方式简单，功能齐全，后续的软件升级有保障，而且有 Keil C51 开发经历的读者朋友可以更快上手；它支持 CM3 系列所有的芯片，同时也支持众多厂商的其他集成电路（IC），在单片机编程方面可以说是领军性的编译器；其他的诸如 IAR 等也是十分成熟的单片机编译器。

一般情况下，我们会使用集成开发环境（IDE）做以下事情：

（1）编写程序代码。

（2）编译程序。

（3）烧写程序。

（4）调试程序，包括查看变量、内存、寄存器、时间跟踪分析，甚至可以调用虚拟打印窗和虚拟逻辑分析仪用以显示程序输出。

（5）输出需要的文件，如 Hex、Bin、Lib 等。

2.1.1　新建工程和添加源代码

新建工程和增加代码的步骤如下：

（1）创建工程。进入 Keil 4.0 系统界面后，选择 Project→New μ Vision Project 命令（见图 2.1），然后输入工程名称，选择路径，进行保存。

（2）增加一个组（Group），如图 2.2 所示。增加 Group 的作用是将工程中不同功能模块的代码文件分类排放，代码控制起来更加方便。

（3）向 app Group 内添加文件，如图 2.3 所示。

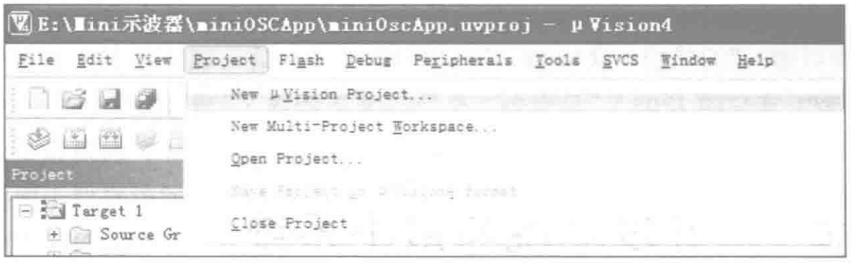


图 2.1 Keil 下新建工程

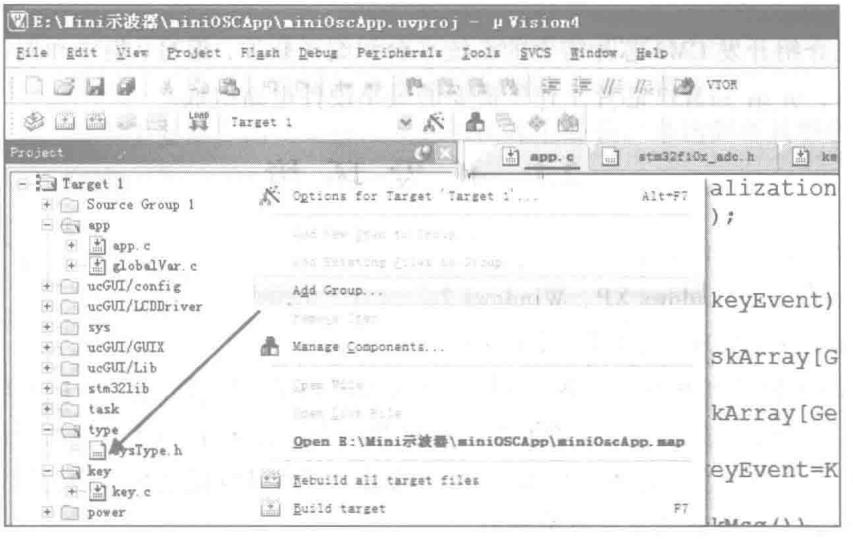


图 2.2 增加 Group

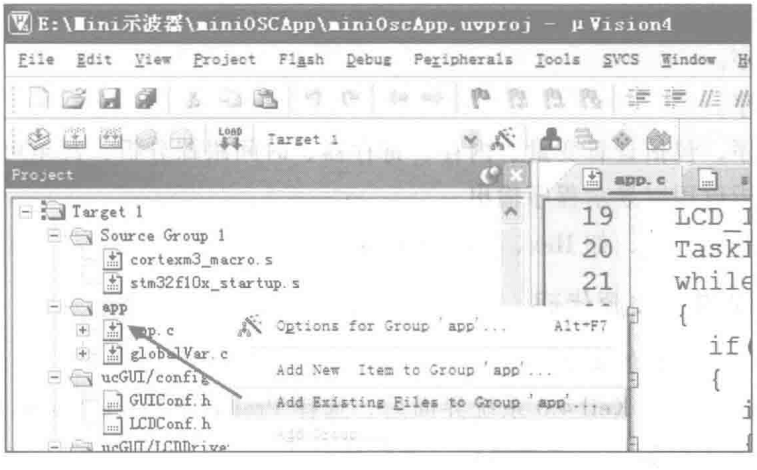


图 2.3 向 Group 内添加文件

2.1.2 设置工程

设置工程的步骤如下：

（1）单击图 2.4 所示按钮，或者从 Project 菜单进入设置界面。

图 2.4　进入工程设置

（2）在 Device 选项卡下选择芯片，如图 2.5 所示。

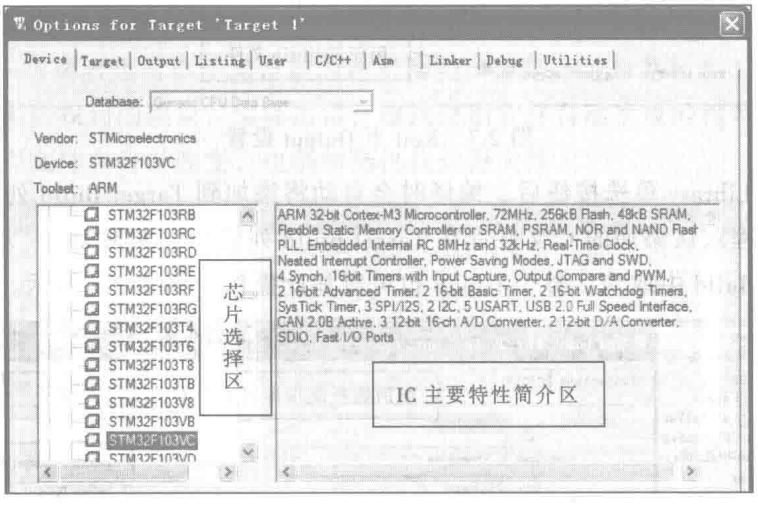


图 2.5　设置 Device 选项卡

（3）设置 Target 选项卡中相应选项，如图 2.6 所示。

图 2.6　设置 Target 选项卡

程序占用 ROM 是指程序编译后的代码容量；程序占用 RAM 是指程序运行期间各种变量所使用的“内存”。

（4）设置 Output 中相关选项，如图 2.7 所示。

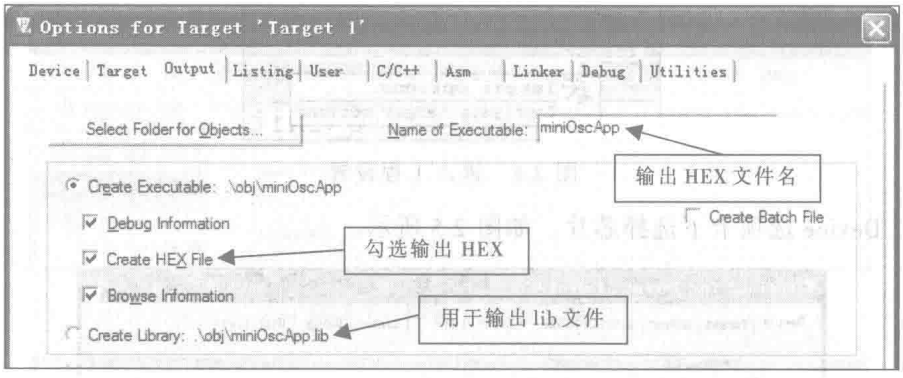


图 2.7 Keil 下 Output 设置

选中 Create Library 单选按钮后，编译时会自动将添加到 Target Build 列表的 C 文件编译到一个库中；工程默认添加的 C 文件都在 Target Build 列表中；如果 app.c 无须编译到库内，可将其从 Target Build 中取消，整个 Group 也可以如此操作，如图 2.8 所示。

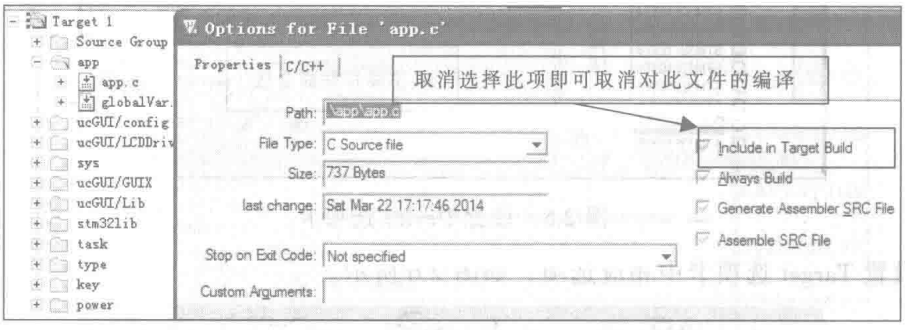


图 2.8 Keil 下文件属性

（5）设置 User 选项卡，编译输出 Bin 文件，如图 2.9 所示。

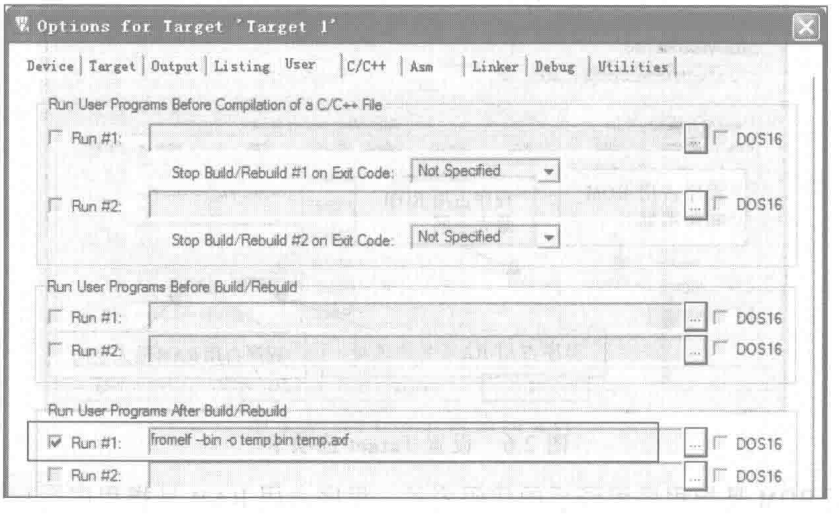


图 2.9 设置输出 Bin 文件

有时，在开发过程中需要编译出 Bin 文件而不是 Hex 文件，要编译出 Bin 文件就要手动

写入指令来控制编译器执行生成 Bin。指令格式如下：

```
Fromelf -bin -o ×××.bin ×××.axf
```

其中，×××.axf 对应工程中此文件的实际名称；×××.bin 为要生成的 bin 文件名称；同时要选中 Run #1 复选框。

（6）如图 2.10、图 2.11 所示，添加文件夹路径后，写程序时就可以直接写入#include "key.h"，编译器会自动在已经包含的文件夹内搜索文件，不用再指出其绝对路径或者相对路径，程序书写更加方便；添加文件夹以后，Keil 默认使用文件夹在工程内的相对路径，如.\GUI\LCDDriver，读者也可以手动设定为绝对路径；但是，绝对路径在工程文件夹目录改变后会出现文件无法找到的现象，编译出错；建议使用软件自动生成的相对路径，使用相对路径时，即使工程文件夹目录改变，也仍然能再找到源文件。

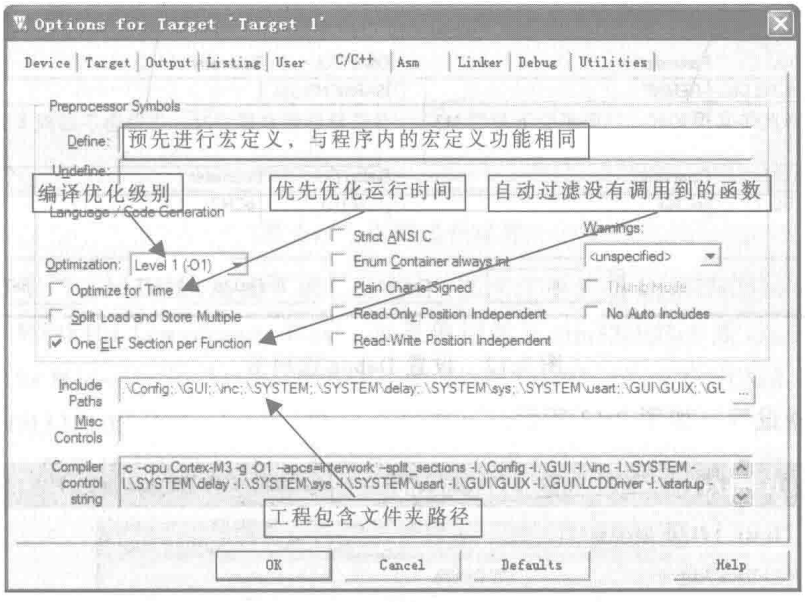


图 2.10　设置 C/C++选项卡

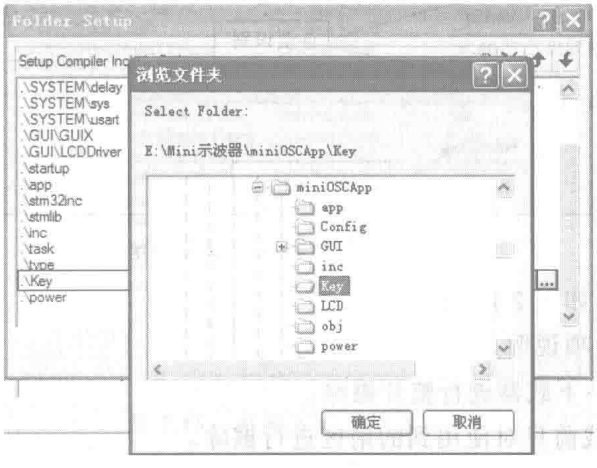


图 2.11　C/C++路径选择

（7）设置 Debug 选项卡，如图 2.12 所示。

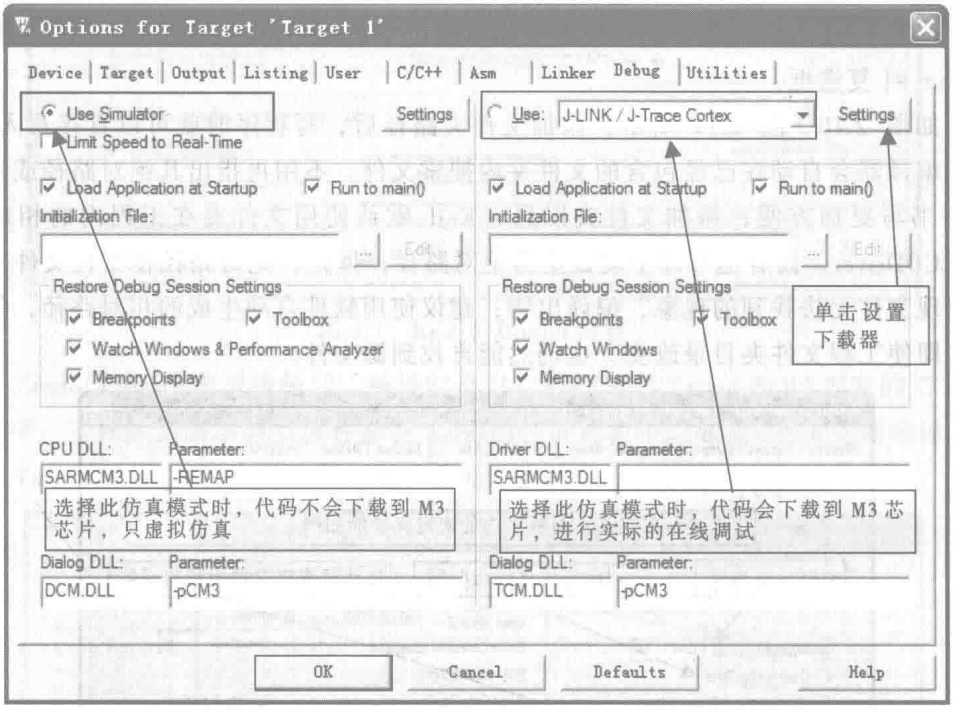


图 2.12　设置 Debug 选项卡

（8）下载器设置，如图 2.13 所示。

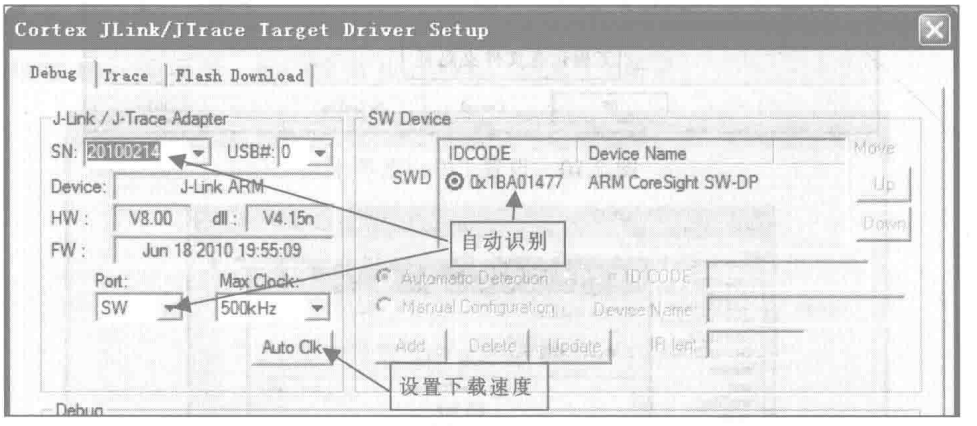


图 2.13　下载器设置之 Debug

（9）下载过程设置如图 2.14 所示。

下载过程涉及的选项说明如下：

Erase Full Chip：在下载器进行整片擦除。

Erase Sectors：下载前只对使用到的扇区进行擦除。

Do not Erase：下载前不擦除。